

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**КРОССПЛАТФОРМЕННОЕ КЛИЕНТ-
СЕРВЕРНОЕ ПРИЛОЖЕНИЕ**

Студент

31.05.2022 РМ
(подпись, дата)

Щербакова Кристина Александровна

Руководитель

31.05.22 Ю
(подпись, дата)

Юрова Ольга Викторовна

Нормоконтролёр

31.05.2022 А
(подпись, дата)

Бычков Андрей Станиславович

Работа допущена к защите (протокол зас. кафедры от 01.06.2022 г. № 13)

Заведующий кафедрой

М. А.
(подпись)

М. А. Митрохин

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

15.06.2022 № 13)

Секретарь ГЭК

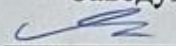
Е
(подпись)

Е.И. Гурин

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«26» апреля 2022 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Щербакова Кристина Александровна гр. 18ВВ1 факультета вычислительной техники.

2. Тема работы «Кроссплатформенное клиент-серверное приложение»

Тема утверждена приказом ПГУ №357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О. В.

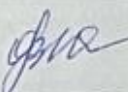
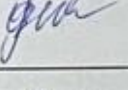
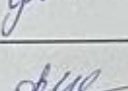
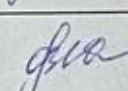
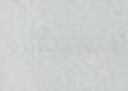
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки – кроссплатформенное клиент-серверное приложение. Проект должен иметь клиент-серверную архитектуру и обладать всеми особенностями и признаками кроссплатформенных приложений, содержать интуитивно понятный интерфейс, иметь систему взаимодействия с игровыми объектами.

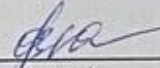
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, определить архитектуру приложения, выбрать способ взаимодействия между клиентом и сервером, изучить особенности кроссплатформенной разработки и возможности кроссплатформенных приложений, разработать протокол взаимодействия, выбрать технологии, применяемые при проектировании, выбрать языки программирования и среду разработки, разработать кроссплатформенное приложение, создать пользовательский и графический интерфейсы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10 %	26.04.22 - 29.04.22	
2.	Выбор архитектуры приложения и технологий, применяемых при проектировании и разработке	15 %	30.04.22 - 04.05.22	
3.	Разработка графического интерфейса	20 %	05.05.22 - 09.05.22	
4.	Разработка функционала приложения	25 %	10.05.22 - 21.05.22	
5.	Тестирование и отладка	15 %	22.05.22 - 24.05.22	
6.	Выполнение задания по БЖД	5 %	25.05.22 - 27.05.22	
7.	Оформление выпускной работы	10 %	28.05.22 - 31.05.22	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы 26.04.2022  Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент 26.04.2022  Щербакова К.А.
(подпись, дата)

Работу к защите допустить
Декан факультета 14.06.2022  Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 119 листов, 32 рисунка, 10 таблиц, 6 источников, 8 приложений.

КЛИЕНТ – СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, САПЁР, JAVA, КРОССПЛАТФОРМЕННОЕ ПРИЛОЖЕНИЕ, NETBEANS, КЛИЕНТ, СЕРВЕР

Цель работы – разработка кроссплатформенного клиент – серверного приложения «Сапёр». Данное приложение должно соответствовать классическим правилам, содержать интуитивно понятный интерфейс, выдавать информацию о своем текущем состоянии. Объектом разработки является клиент – серверное приложение, состоящее из клиентской программы с графическим интерфейсом и сервера с игровым полем.

Результатом разработки клиент – серверного кроссплатформенного приложения является проект, удовлетворяющий поставленной задаче. А именно, разработано, реализовано и протестировано приложение для игры, поддерживающее как одиночную игру, так и многопользовательскую игру.

Разработка производилась с использованием большого стекатехнологий, к которым относятся: язык программирования Java, интегрированная среда разработки NetBeans.

					ПГУ 1.090301.45.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Кроссплатформенное клиент-серверное приложение Пояснительная записка	Лит.	Лист	Листов
Разраб.		Щербакова К.А.					2	124
Провер.		Юрова О.В.						
Н. Контр.		Бычков А. С.						
Утверд.		Митрохин М.А.						
						ФВТ, каф. ВТ гр. 18ВВ1		

СОДЕРЖАНИЕ

Введение	4
1 Кроссплатформенное клиент-серверное приложение.....	6
1.1 Анализ предметной области	6
1.1.1 Постановка задачи	8
1.1.2 Анализ аналогов	8
1.1.3 Анализ инструментов разработки.....	13
1.1.4 Анализ языков программирования	17
1.2 Проектирование и разработка.....	20
1.2.1 Выбор решения	20
1.2.2 Описание работы клиент – серверного приложения	20
1.2.3 Описание способа организации пользовательского интерфейса	33
1.3 Тестирование результатов работы.....	36
2 Безопасность жизнедеятельности.....	42
2.1 Характеристика потенциальных опасностей и вредностей.	43
2.2 Классификация и категорирование помещения	44
2.3 Санитарно-гигиеническая характеристика помещения	46
2.3.1 Требования к помещениям для работы с ПЭВМ	46
2.3.2 Организация рабочих мест	47
2.3.3 Микроклимат рабочего помещения.....	49
2.3.4 Шум и вибрация.....	50
2.3.5 Ионизирующие и неионизирующие излучения	51
2.3.6 Освещение рабочего места.....	51
2.4 Электробезопасность.....	52

2.5 Пожарная безопасность.....	54
2.6 Расчет общего искусственного освещения помещения	56
Заключение	60
Список использованных источников	61
Приложение А – <i>UML</i> -диаграмма классов клиента	62
Приложение Б – <i>UML</i> -диаграмма классов сервера	64
Приложение В – <i>UML</i> -диаграмма прецедентов.....	66
Приложение Г – <i>UML</i> -диаграмма развертывание	68
Приложение Д – <i>UML</i> -диаграмма последовательности.....	70
Приложение Е – <i>UML</i> -диаграмма деятельности	72
Приложение Ж – Листинг программы	74
Приложение З – Презентация	118

Введение

Кроссплатформенные приложения – приложения, которые разрабатываются и затем мгновенно могут работать на многих платформах. Суть разработки в том, что исходный код приложения переводится в нативный, то есть очевидный определенному устройству. В результате программа сможет взаимодействовать с установленной на нём операционной системой.

На протяжении многих лет кроссплатформенная разработка завоевала репутацию одного из самых известных течений разработки программного обеспечения. Кроссплатформенный подход разрешает создавать приложения для всевозможных платформ с одной кодовой базой, что экономит время и деньги и освобождает от лишних усилий.

Кроссплатформенная разработка имеет очевидные плюсы, выделяющие ее среди остальных отраслей компьютерных технологий.

Расширение пользовательской базы за счёт появления приложения одновременно в нескольких магазинах – это и дальнейшие преимущества располагают большую значимость непосредственно для бизнес-целей. Один исходный код избавляет компанию от надобности нанимать нескольких разработчиков для каждой платформы. Также важным преимуществом является повышение скорости разработки. Для разных проектов показатели сокращения сроков запуска будут различными в зависимости от функционала приложения, насколько оно сложное, но в среднем экономия времени сможет достигать 20%. И это уменьшает time-to-market – сроки выхода продукта на рынок. 75% кодовой базы кроссплатформенного приложения можно использовать повторно, адаптируя её для новых проектов, это помогает избавиться от написания уникального кода для различных платформ, тем самым экономить время и усилия. Снижаются затраты на исправление ошибок и добавление нового функционала. Важным плюсом представляется простая облачная интеграция такого рода

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

приложений. Единый исходный код также позволяет легко интегрировать кроссплатформенное приложение с облачной платформой. Доступны плагины корпоративного уровня для интеграции ваших бизнес-приложений с облаком, что делает их универсально совместимыми.

Но, как и у всего кроссплатформенная разработка имеет свои недостатки. Одними из них является неидеальный интерфейс с точки зрения адаптации, ограниченность среди инструментов мультиплатформенной разработки, а именно каждый год платформы *Android* и *iOS* выпускают обновленные версии с расширенными возможностями и функциональностью. Хотя технологии процветают, кроссплатформенным разработчикам трудно поддерживать совместимость приложений с ограниченными инструментами. Кроссплатформенные приложения менее гибкие, чем нативные приложения. Нативные приложения популярны благодаря богатому пользовательскому опыту, хорошей работе и привлекательному интерфейсу. Теперь кроссплатформенные приложения также могут иметь элегантный интерфейс, но иногда они не соответствуют ожиданиям пользователя. Низкая скорость и сбои, связанные с производительностью, могут испортить впечатление пользователя [1].

Сегодня бизнес сталкивается с огромной конкуренцией — предприниматели ориентируются на огромную аудиторию, которая состоит из пользователей смартфонов. Кроссплатформенное приложение является экономически эффективным решением.

Кроссплатформенные инструменты разработки приложений постоянно развиваются, и разработчики могут с легкостью создавать приложения корпоративного уровня.

1 Кроссплатформенное клиент-серверное приложение

1.1 Анализ предметной области

Кроссплатформенная игра — вариант онлайн-игр, которые поддерживают геймплей между игроками с разными устройствами. Простой пример — вы и трое ваших товарищей решаете провести вечер в боях отрядов в Fortnite. Вы заходите в игру с компьютера, а ваши партнеры применяют две разные консоли и, например, смартфон с большим экраном.

Кроссплатформенный — это термин, используемый для обозначения части программного обеспечения, совместимого с более чем одной системой. Например, известный медиаплеер *VLC* соединим с тремя главными настольными операционными системами: *Microsoft*, *Mac OS* и *Linux*. Кроссплатформенная поддержка также может распространяться на мобильные устройства, при этом значительные приложения доступны как в *Apple App Store*, так и в *Google Play Store*.

В зависимости от того, как запрограммировано определенное программное обеспечение, разработчику может понадобиться полностью перекодировать и переупаковать программное обеспечение. Некоторые фреймворки позволяют разработчикам более плавно включать кроссплатформенную поддержку своего программного обеспечения.

Есть множество причин, по которым пользователям удобно применять кроссплатформенное программное обеспечение. Например, при использовании программного обеспечения одним из наиболее важных причин является совместимость файлов. Если вы применяете формат файла, доступный только в одной операционной системе, вы можете подумать о преобразовании его в формат с помощью средства чтения, которое функционирует во многих системах.

Кроссплатформенная игра объединена с понятием кроссплатформенной разработки, которая применяет программные языки и инструменты для развертывания программного обеспечения на нескольких

платформах. Кроссплатформенная игра — это еще хорошая концепция от возможности разрешить игроку играть в игру на разных аппаратных платформах, зачастую имея лишь покупку названия ради одной системы, чтобы обладать доступом к ней на иных системах, и сохраняя свой прогресс в игре с помощью облачного хранилища или аналогичного приема [2].

Когда дело доходит до игр, кросс-совместимость — огромная тема. Во времена консольных войн с 1980-х до начала 2010-х кроссплатформенная совместимость была огромной проблемой. Ключевые производители игр, такие как *Sony*, *Microsoft* и *Nintendo*, создавали и финансировали игры. Эти игры зачастую разрабатывались в их собственных студиях либо игровыми студиями, с которыми у них имелось стратегическое партнерство.

Отчего многие игры были «эксклюзивными», их возможно было приобрести и играть исключительно на одном устройстве. Когда покупатели решают, какую консоль покупать, они должны учитывать, какие игры доступны для конкретных консолей. Например, *Halo* всегда будет на *Xbox*, *Uncharted* будет на *Playstation*, а Марио будет на консолях *Nintendo*.

В то время как *Nintendo* в основном придерживалась эксклюзивных предложений, *Microsoft* и *Sony* приступили представлять более кроссплатформенную поддержку. Из-за роста популярности компьютерных игр в последнее десятилетие значительные крупные эксклюзивные игры начали появляться на ПК и консолях. Из-за более высокого потолка производительности на настольных компьютерах разработчики также приступили разрабатывать свои игры с учетом потребностей компьютерных геймеров и снижать производительность при переходе на консоли.

Кросс-игра также существенно повышает доступную базу пользователей определенной игры, что хорошо, как для игроков, так и для разработчиков. Активное крупное сообщество уменьшает время ожидания, повышает спектр уровней навыков, доступных для игры, и делает более привлекательным для новых игроков начало игры.

Учитывая, как эта функция вошла в игры за последние несколько лет, вероятно, вскоре мы увидим больше игр с кроссплатформенной поддержкой. Игровые студии начали осознавать, насколько кроссплатформенность полезна для активности игроков. Не удивительно, если в большинстве многопользовательских игр в будущем будет поддержка кроссплатформенной игры.

1.1.1 Постановка задачи

Целью данной работы является проектирование и разработка кроссплатформенного клиент-серверного приложения «Сапёр». Программа должна включать сетевое взаимодействие. Серверная часть должна обеспечивать свободное подключение клиентов через локальную сеть и осуществлять кооперативную игру с другими пользователями. Отключение клиентов в любой момент игры. Также серверная программа должна иметь графический интерфейс для игры. Клиентская часть должна подключаться к серверу и иметь возможность выбора сложности игры.

Приложение должно содержать следующие требования:

- возможность запуска на любой платформе;
- система взаимодействия с игровыми объектами;
- режим совместной игры с другими людьми;
- интуитивно понятный графический интерфейс пользователя;
- многоуровневая система;
- серверная часть должна давать возможность подключения и отключения клиента в любое время игры.

1.1.2 Анализ аналогов

«Сапер» — это логическая игра-головоломка, действие которой разворачивается на сетчатом минном поле.

«Сапер» (*Minesweeper*) изначально назывался *Mine* и был создан работниками *Microsoft* Робертом Доннером и Куртом Джонсоном. Доннер адаптировал игру для *Windows 3.0*, взяв более раннюю версию Джонсона для *OS/2*. К слову, обе игры всегда распространялись лишь среди друзей.

Игра «Сапёр» содержит в себе плоское или объёмное игровое поле разбито на смежные ячейки отдельные из которых, «заминированы». Количество «заминированных» ячеек известно. Целью игры является открытие всех ячеек, не содержащих мины.

Игрок раскрывает ячейки, пытаясь не открыть ячейку с миной. Обнаружив ячейку с миной, он проигрывает. Если под обнаруженной ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано» используя данные числа, игрок старается рассчитать положение мин, впрочем, порой даже в середине и в конце игры отдельные ячейки всё же приходится раскрывать наугад. Если под соседними ячейками тоже нет мин, то открывается определенная «не заминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы нечаянно не открыть их. Открыв все «не заминированные» ячейки, игрок выигрывает.

На сегодняшний момент вышло много разновидностей игры данного жанра, и почти каждая старается внедрить что-то новое.

Aqueduct — это игра в жанрах стратегии и головоломки, созданная *Kieffer Bros*. Она вышла 29 июня 2010 г. *Kieffer Bros* выступила издателем игры. Цель в этой игре заключается в том, чтобы вы провели водичку в небольшие озёра, чтобы не дать им засохнуть. Перед любым своим шагом думайте, однако спустя несколько уровней, необходимо будет протянуть водичку с одного источника аж в несколько озёр. Есть возможность разрушать детали, а также и менять их расположение. На рисунке 1 представлен игровой интерфейс *Aqueduct*.

Zombie Minesweeper - головоломка и однопользовательская видеоигра, выпущенная для мобильных платформ, таких как *Android* и *iOS* *Octo Industries*. Игра доставит вас в приключение, полное насыщенных взрывов зомби и трудных головоломок. Согласно истории, основной герой застрял в апокалипсисе зомби, и его цель - обнаружить способ убежать и повстречать своего бойфренда. На рисунке 3 представлен игровой интерфейс *Zombie Minesweeper*.

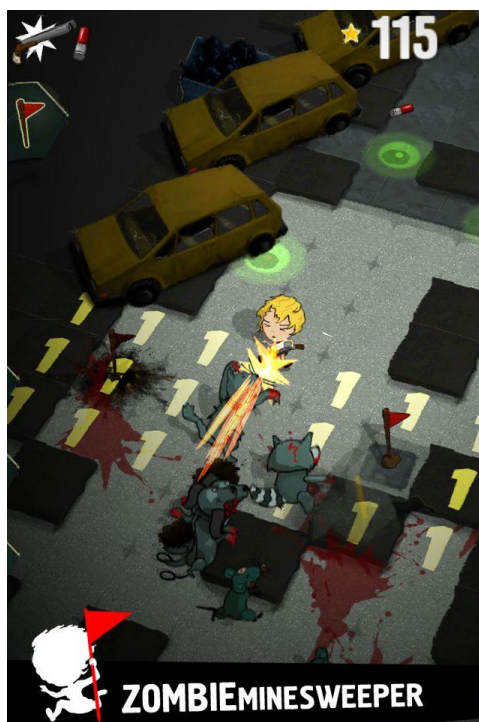


Рисунок 3 – Интерфейс *Zombie Minesweeper*

Pony Minesweeper, разработанный Коринфскими играми, представляет собой головоломную и однопользовательскую видеоигру на базе давней классической игры под названием «Сапер». Игра предлагает фантастический игровой процесс и включает в себе девять различных пони, таких как *Lazy Days*, *Lemon Suds*, *Puddles*, *Time-Jump*, *Serenity*, *Orchard*, *Starmist* и многое другое. Имеется четыре предмета, которые можно использовать во время игры, чтобы повысить ваши способности, такие как резиновые сапоги, яблоки, вилы. На рисунке 4 представлен игровой интерфейс *Pony Minesweeper*.



Рисунок 4 – Интерфейс *Pony Minesweeper*

Новичок *MineSweeper* — это однопользовательская головоломка, сделанная *News All*. Игра предлагает похожий игровой процесс для *Minesweeper* и предлагает разные режимы и масштабы для игры. Есть множество уровней, и все уровни делятся на различные категории, такие как *Beginner*, *Intermediate* и *Expert*. Каждый из них состоит из клеток, и ваша цель - отыскать клетку, не взорвав бомбу. На рисунке 5 представлен игровой интерфейс *MineSweeper*.



Рисунок 5 – Интерфейс Новичок *MineSweeper*

1.1.3 Анализ инструментов разработки

На сегодняшний день существует множество различных сред разработки приложений, каждая из которых обладает своими достоинствами, которые её выделяют, и недостатками. Рассмотрим поподробнее некоторые из них.

NetBeans IDE - бесплатная интегрированная среда разработки с открытым исходным кодом для разработчиков программного обеспечения. Среда предоставляет все средства, необходимые для создания профессиональных десктоп приложений, корпоративных, мобильных и веб-приложений на платформе *Java*, а также *C/C++*, *PHP*, *JavaScript*, *Groovy* и *Ruby* [3].

Возможности программирования в *NetBeans*:

- рабочая область среды *IDE* является полностью настраиваемой - существует возможность пользовательской настройки действий, выполняемых с помощью панели, назначения "горячих" клавиш и т.д.;
- редактор может генерировать и вставлять в исходный код стандартные фрагменты кода на *Java* или других языках;
- редактор *NetBeans* делает отступы строк, проверяет соответствие скобок и слов, подсвечивает синтаксис исходного кода;
- браузер классов позволяет просматривать иерархию и структуру любого класса *Java* - отображаются интерфейсы, базовые классы, производные классы и члены классов;
- *NetBeans IDE* является платформой для построения десктоп приложений с функциональным пользовательским интерфейсом, т.к. представляет из себя фреймворк к *Java* библиотеке *Swing*;
- расширенные средства для выполнения контекстно-зависимого поиска по всей среде *IDE*, справочным материалам и всем открытым проектам, и файлам.

На рисунке 6 представлен интерфейс *NetBeans IDE* и окно пользовательского интерфейса.

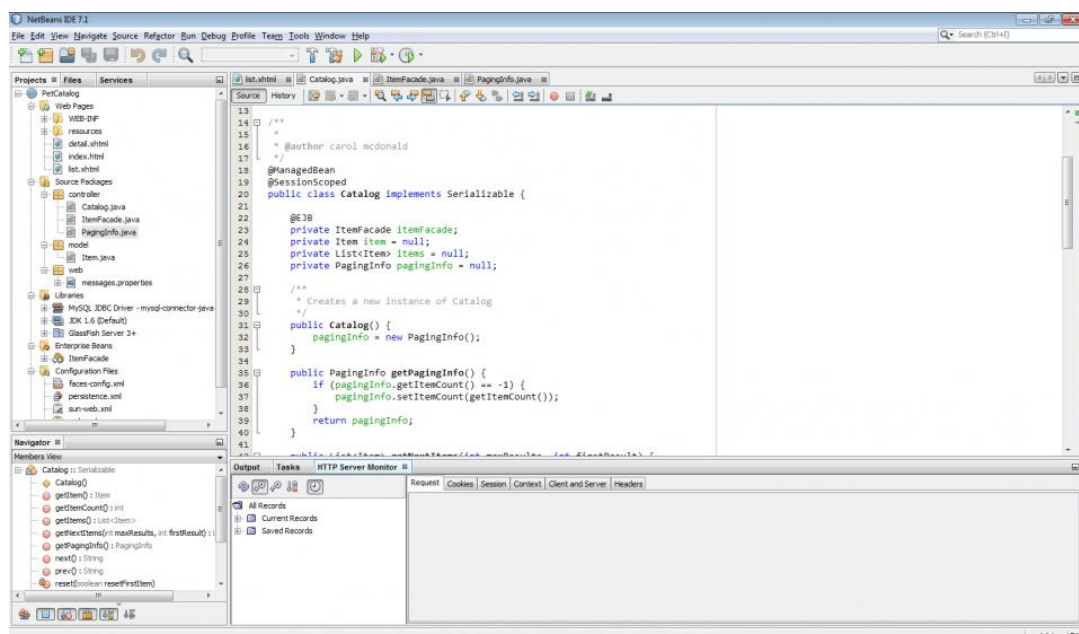


Рисунок 6 – Интерфейс *NetBeans IDE*

Eclipse IDE — свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается *Eclipse Foundation*.

Особо популярные приложения на базе *Eclipse Platform* — различные «*Eclipse IDE*» для разработки ПО на множестве языков. Например, особо популярный «*Java IDE*», поддерживавшийся изначально, не полагается на какие-либо закрытые расширения, применяет стандартный открытый API для доступа к *Eclipse Platform* [4].

На *Eclipse Marketplace* доступно множество плагинов, которые дополняют функциональность. *Eclipse* также поставляется с пользовательским компилятором. Для *Java*-программистов, желающих разработать определенную функциональность для *Eclipse*, доступна *PDE* (среда разработки плагинов). *IDE* также показывает мощные инструменты для построения графиков, моделирования, отчетности и тестирования.

Eclipse также поддерживает разработку приложений на многих языках программирования с помощью плагинов, включая *C*, *C++*, *Clojure*, *Groovy*, *Haskell*, *JavaScript*, *Julia*, *Perl*, *PHP*, *Ruby*, *Rust* и *Scala*.

Изм.	Лист	№ докум.	Подпись	Дата

Среди явных достоинств среды разработки *Eclipse IDE* можно выделить: доступность множества полезных и удобных плагинов, возможность использовать доступ в облако, хорошие инструменты для построения графиков, моделирования, отчетности и тестирования, также среда разработки является бесплатной.

На рисунке 7 представлен интерфейс *Eclipse IDE* и окно пользовательского интерфейса.

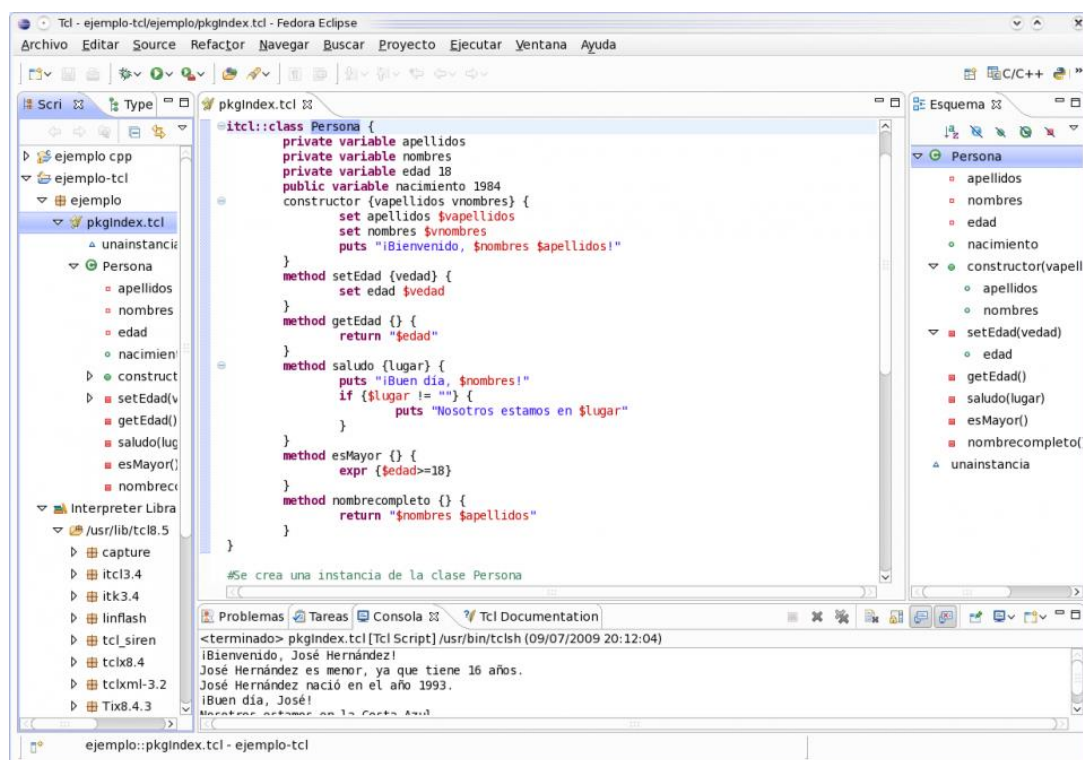


Рисунок 6 – Интерфейс *Eclipse IDE*

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности *Java*, *JavaScript*, *Python*, созданная компанией *JetBrains*. Выпущенная в 2001 году, *IntelliJ IDEA* - еще одна из трех больших *Java IDE*. Он доступен в 2 разных изданиях, лицензионном выпуске сообщества *Apache 2* и проприетарном коммерческом издании. *IntelliJ IDEA* может похвастаться функциями межъязыкового рефакторинга и анализа потоков данных.

Эта *IDE* для *Java* имеет функции, что упрощают жизнь создателя *Java*, такие как завершение цепочки, внедрение языка, интеллектуальное окончание и статическое завершение члена. В дополнение к поддержке *Java* и целого ряда *Java*-фреймворков *IntelliJ IDEA* также гарантирует поддержку иных языков программирования на базе *JVM*, таких как *Kotlin*.

Начиная с шестой версии продукта *IntelliJ IDEA* дает интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди иных возможностей, среда хорошо совместима со многими знаменитыми свободными инструментами разработчиков, такими как *CVS*, *Subversion*, *Apache Ant*, *Maven* и *JUnit*. В феврале 2007 года разработчики *IntelliJ* объявили раннюю версию плагина для поддержки программирования на языке *Ruby*.

Начиная с версии 9. 0 среда доступна в двух редакциях: *Community Edition* и *Ultimate Edition*. *Community Edition* является абсолютно независимой версией, доступной под лицензией *Apache 2.0*, в ней реализована полная поддержка *Java SE*, *Kotlin*, *Groovy*, *Scala*, а также интеграция с наиболее известными системами управления версиями. В редакции *Ultimate Edition*, доступной под коммерческой лицензией, реализована поддержка *Java EE*, *UML*-диаграмм, подсчёт покрытия кода, а также поддержка иных систем управления версиями, языков и фреймворков.

Среди явных достоинств среды разработки *Eclipse IDE* можно выделить: поддержка других языков программирования на основе *JVM*, интеллектуальное завершение цепочки, функции межъязыкового рефакторинга, анализ потоков данных, также среда разработки является бесплатной.

На рисунке 8 представлен интерфейс *IntelliJ IDE* и окно пользовательского интерфейса.

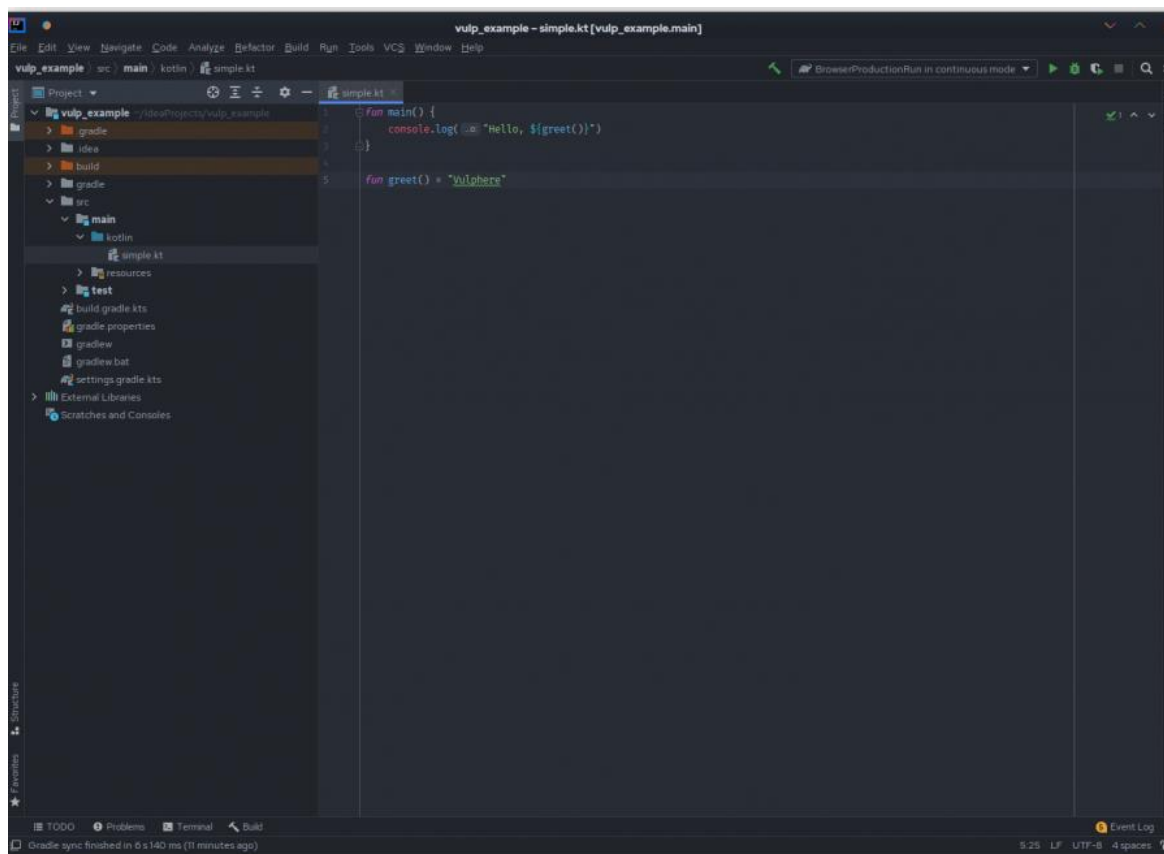


Рисунок 8 – Интерфейс *IntelliJ IDEA*

1.1.4 Анализ языков программирования

На текущий момент есть широкий выбор языков программирования, позволяющих задавать условия и команды в игре.

C++ — компилируемый и статически типизированный язык программирования. Поддерживает ООП и процедурное программирование. Также является кроссплатформенным языком, который позволяет эффективно управлять памятью и при этом не загружать её. Без него не обходится ни одна разработка серьезного игрового продукта. Кроссплатформенность позволяет создавать игры для разных платформ: веб, онлайн или мобильные устройства. Этот язык помогает создавать механику игры — взаимодействие игрока с миром и мира с игроком. На нём пишут оптимизированные крупномасштабные игры уровня AAA. C++ также используют в мобильных играх, где нужно бережно обходиться с памятью устройства и поддерживать высокую производительность игры.

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

C# — компилируемый и статически типизированный язык программирования. Объектно-ориентированный язык программирования. Уступает по скорости *C++*, но обладает более ёмким, в плане возможностей, синтаксисом. Не настолько производителен как *C++* и, соответственно, не подойдёт для больших требовательных проектов. Но *C#* отличается кроссплатформенностью и универсальностью, поэтому на нём можно создавать 2D- и 3D-игры для *Android* и *iOS*. Знание *C#* позволяет разрабатывать мобильные приложения, которые успешно запускаются на всех моделях современных консолей, смартфонов и планшетов. Ключевую роль язык высокого уровня *C#* играет при разработке крупных проектов, связанных с созданием сайтов и интернет-порталов. Применение *C#* позволяет реализовать сложные интернет-магазины, форумы, доски объявлений, интернет-банкинг.

Java — на сегодняшний момент представляется одним из самых распространенных и популярных языков программирования. Первая версия языка возникла еще в 1996 году в недрах фирмы *Sun Microsystems*, дальше поглощенной фирмой *Oracle*. *Java* задумывался как универсальный язык программирования, который возможно использовать для разнообразного рода задач. И к настоящему времени язык *Java* проделал огромный путь, было выпущено множество разнообразных версий. А *Java* превратилась из просто универсального языка в целую платформу и экосистему, которая связывает различные технологии, используемые в целом ряде задач: от создания десктопных приложений до написания больших веб-порталов и сервисов. Кроме того, язык *Java* активно используется для создания программного обеспечения для целого ряда устройств: обычных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники. Достаточно вспомнить популярность мобильной ОС *Android*, большинство программ для которой пишутся только на *Java*.

Отличительными особенностями *Java* в сравнении с другими языками программирования общего назначения является:

– платформонезависимость: в отличие от многих других языков, включая *C* и *C++*, *Java*, когда был создан, он не компилировался в платформе конкретной машины, а в независимом от платформы байт-коде. Этот байт код распространяется через интернет и интерпретируется в *Java Virtual Machine (JVM)*, на которой он в настоящее время работает;

– безопасность: методы проверки подлинности основаны на шифровании с открытым ключом;

– архитектурная-нейтральность: компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системы *Java Runtime*;

– портативность: архитектурно-нейтральный и не имеющий зависимости от реализации аспектов спецификаций — все это делает *Java* портативным. Компилятор в *Java* написан на *ANSI C* с чистой переносимостью, который является подмножеством *POSIX*;

– ООП: в *Java* все является объектом. Дополнение может быть легко расширено, так как он основан на объектной модели;

– интерпретированность: *Java* байт-код переводится на лету в машинные инструкции и нигде не сохраняется. Делая процесс более быстрым и аналитическим, поскольку связывание происходит как дополнительное с небольшого весом процесса;

– высокая производительность: введение *Just-In-Time* компилятора, позволило получить высокую производительность.

По итогу проведения анализа предметной области и технологий, для разработки кроссплатформенного клиент-серверного приложения были выбраны межплатформенная среда разработки *NetBeans*, как самая доступная и простая в обращении, и язык программирования *Java*.

1.2 Проектирование и разработка

1.2.1 Выбор решения

Для написания данной программы использован язык программирования *Java*.

«Сапёр» (англ. *Minesweeper*) — популярная компьютерная игра-головоломка, входящая в состав практически всех версий операционной системы *Windows*.

Для реализации проекта используется:

- библиотека *Swing*;
- интерфейс *Runnable*;
- пакеты, содержащие файлы ресурсов, пользовательский интерфейс и сетевое взаимодействие, и игровую логику;
- унаследованные от класса *Thread* классы, обработка исключительных ситуаций;
- сетевое взаимодействие;
- *Java Stream API* и тд.

1.2.2 Описание работы клиент – серверного приложения

Программа реализует кооперативное прохождение игры «Сапёр» для пользователей, подключенных к серверу по локальной сети.

Для корректной работы необходимо запустить сначала проект «*SaperKursServer*». Появится окно с кнопкой «Запустить сервер» (рисунок 9), при нажатии на которую появится диалоговое окно с полем ввода порта сервера и тремя кнопками выбора уровня сложности, при нажатии на любую из которых сервер будет запущен под указанным ранее в поле портом.

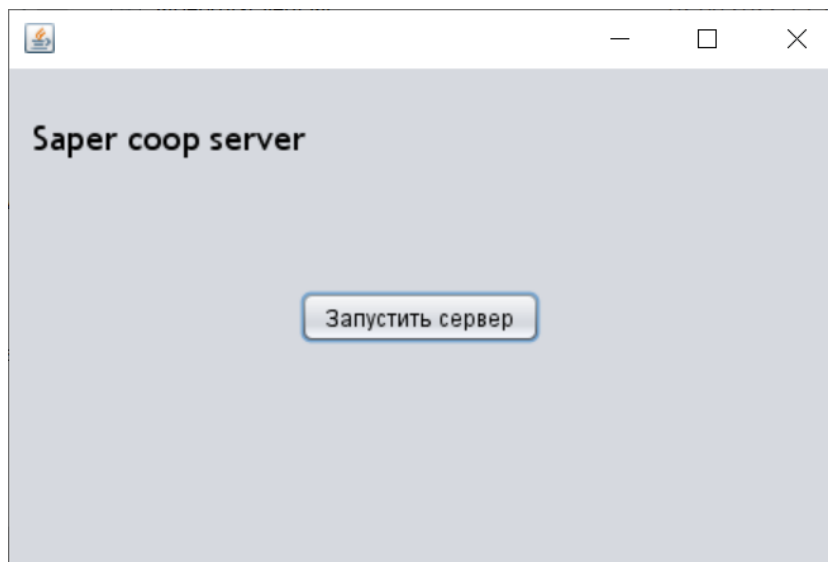


Рисунок 9 – Стартовое окно сервера

После запуска сервера запускается проект «*SaperKursClient*». Появится диалоговое окно с двумя полями ввода: «Адрес» и «Порт», и кнопкой «Подключиться» (рисунок 10).

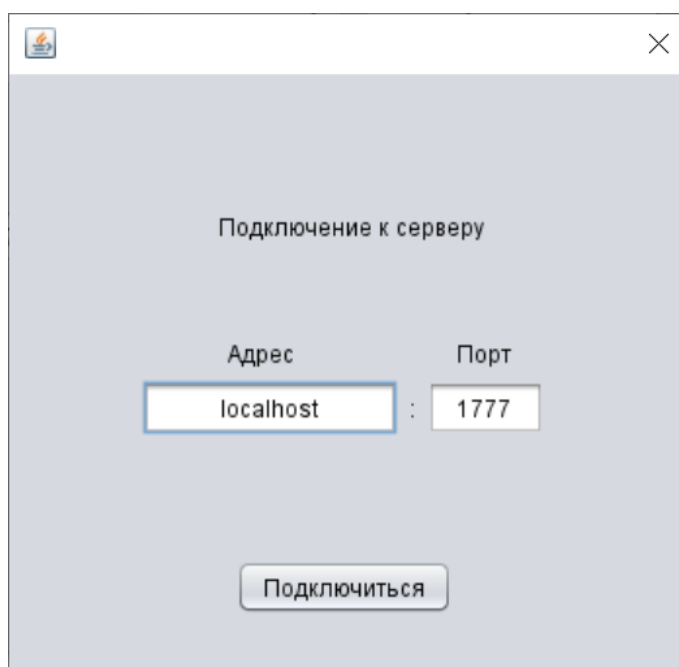


Рисунок 10 – Окно сервера для изменения адреса и порта

При корректно введенных данных, после нажатия на кнопку происходит подключение к серверу и на экран выводится окно с полученным от сервера игровым полем. Игровое поле представлено на рисунке 11.

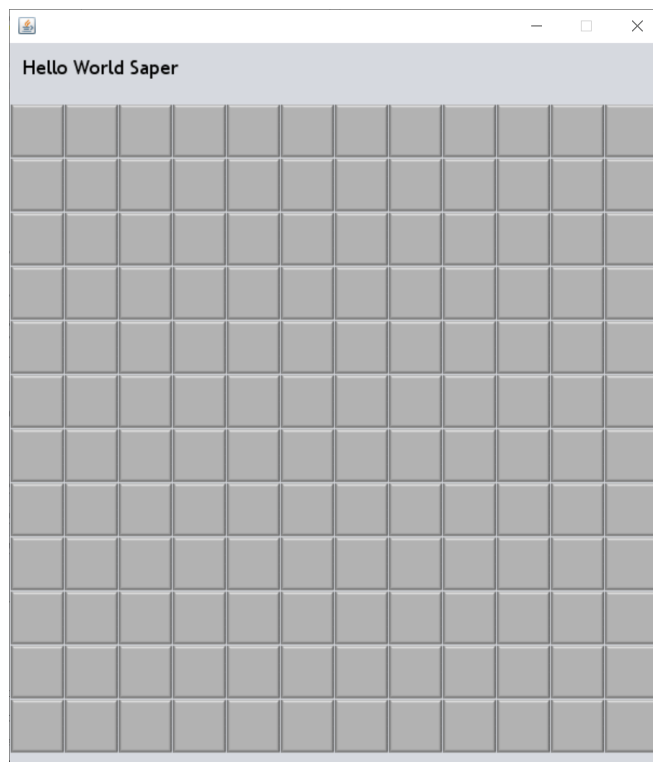


Рисунок 11 – Игровое поле

На рисунке 12 показано игровое поле после открытия свободных ячеек.

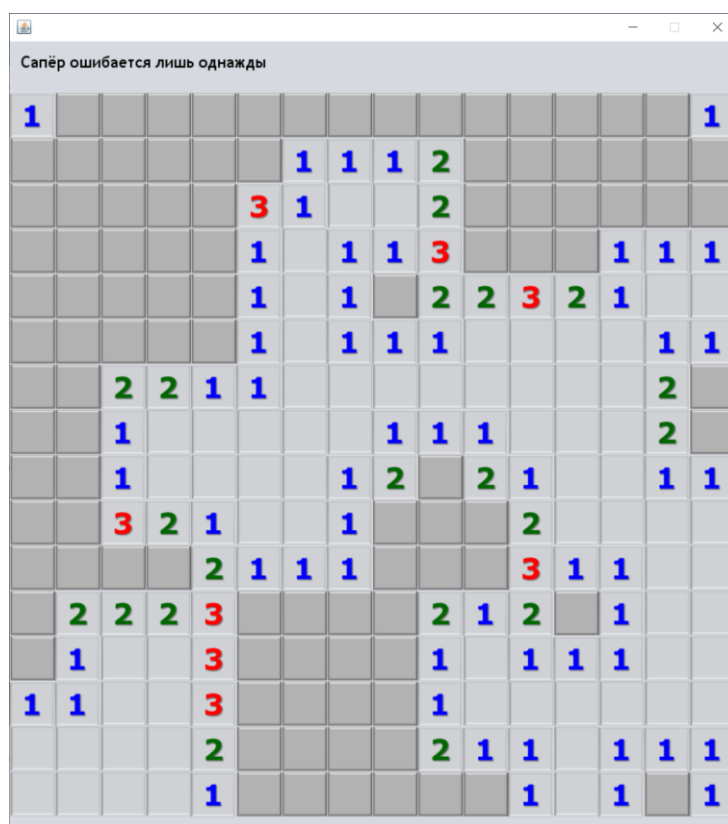


Рисунок 12 – Игровое поле при открытии ячеек

На рисунке 13 показано игровое поле при попадании на ячейку с бомбой.



Рисунок 13 – Игровое поле при попадании на ячейку с бомбой

Результатом программы является кооперативное прохождение игры «Сапёр», т.е. подключение к серверу и обмен действиями между участниками.

Запуск сервера происходит следующим образом. В зависимости от выбранной кнопки уровня сложности выбирается количество строк и столбцов ячеек игрового поля, а также бомб: «Легко – 9 столбцов, 9 строк, 10 бомб», «Норм – 12 столбцов, 12 строк, 20 бомб», «*Saper* – 16 столбцов, 16 строк, 40 бомб» (рисунок 14), - после чего создаётся серверный сокет под указанным при запуске портом. Далее запускается новый поток, который будет отслеживать подключение новых клиентов к серверу и вместе с тем записывать их в коллекцию *ArrayList*. Каждое подключение клиента в дальнейшем также будет обрабатываться в новом потоке.

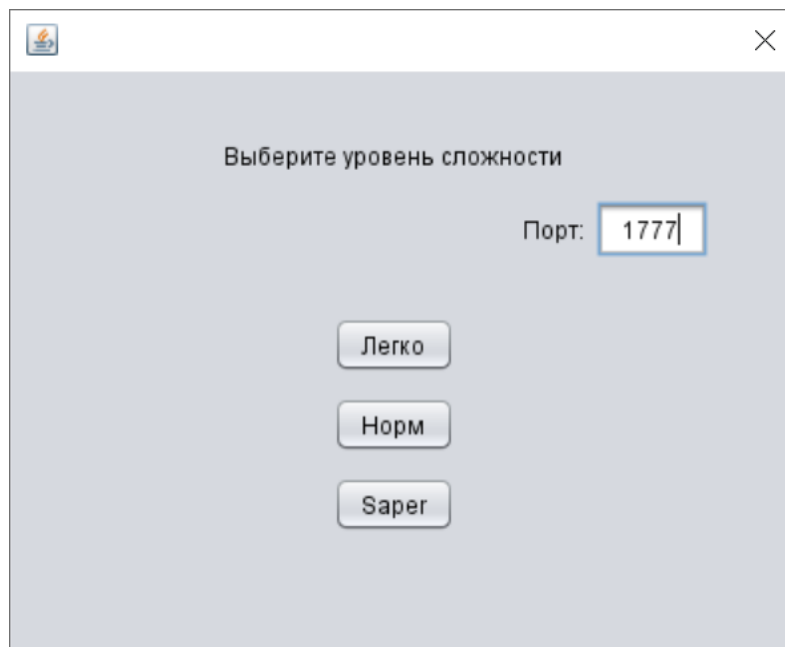


Рисунок 14 – Окно выбора сложности

При обнаружении запроса на подключение от клиента при условии, что серверный сокет корректно запущен, создаётся объект, унаследованный от класса «*Thread*» класса «*SocketThread*», в конструктор которого передаётся сокет клиента, его уникальный номер и экземпляр созданной игры.

После запускается новый поток, который записывает в выходной поток (*OutputStream*) размер игрового поля и экземпляр созданной игры, после чего инициализирует класс координат и входит в бесконечный цикл в ожидании запросов от клиента. Запросы представляют из себя простые слова класса *String*.

Запрос «*Left*», «*Right*» - клиент оповещает, что была нажата левая или правая кнопка мыши, в этом случае входной поток (*InputStream*) считывает координаты, на которые было произведено нажатие кнопки мыши, и передаёт их статическому методу класса «*Server*» «*PressButton*» вместе со своим уникальным номером и полученным сообщением. Метод перебирает коллекцию клиентских сокетов и передаёт им полученные данные при условии, если клиентский сокет в рабочем состоянии, и он не является адресатом (то есть не от того же самого сокета был произведён запрос). После чего конкретно связанный со своим клиентом сокет записывает в

выходной поток сообщение, обозначающее нажатую кнопку, и координаты.

Запрос «*NgPls*» - клиент оповещает о завершении игрового процесса и просит новый экземпляр игры. Сервер инициализирует объект класса *Game* и записывает его в выходные потоки клиентских сокетов по указанному ранее принципу.

Запрос «*Exit*» - клиент оповещает о завершении своей работы. В этом случае при условии, что подключение с клиентом всё ещё установлено (эта проверка нужна на случай, если разрыв соединения был произведён самим сервером, в этом случае поток принимает сообщение о завершении работы клиента и просто прекращает свою работу), поток отдельного клиентского сокета закрывает входной и выходной поток и прекращает свою работу. На рисунке 15 представлена *UML*-диаграмма последовательности.



Рисунок 15 – *UML*-диаграмма последовательности

При корректном подключении к серверу клиент считывает из входного потока размеры игрового поля и экземпляр созданной игры. Далее клиент запускает новый поток, который будет отслеживать запросы других клиентов, переданные через сервер.

При получении одного из таких, клиент будет обновлять игровое поле в соответствии с выполненным другим клиентом действием.

Пакет «*res.img*» содержит картинки типа *png*, которые отображаются на панели игрового поля.

В пакете «*sweeper*», что является общим и для сервера, и для клиента, находятся классы и перечисления, реализующие игровую логику.

Перечисление (*enum*) «*Box*» содержит название картинок в специальном порядке, объект «*image*» для отображения картинок, метод «*GetNumber*» для получения числа, который эквивалентен номеру картинки в перечислении; и метод «*GetNextNumberBox*» для получения следующего числа относительно текущего.

Класс *Coord.java* хранит координаты клетки поля и сравнивает координаты друг с другом. Поля класса *Coord.java* представлены в таблице 1.

Таблица 1 – Описание полей класса *Coord.java*

Тип данных	Название	Описание
<i>int</i>	<i>x</i>	Координаты поля
<i>int</i>	<i>y</i>	Координаты поля

Класс *Coord.java* включает в себя метод *Public boolean equals()* который, сравнивает координаты объектов поступающих в класс.

Класс *Ranges.java* хранит в себе размеры поля (статическое поле *Coord size*), статическая коллекция *ArrayList* «*allCoords*» типа «*Coord*», где хранятся все игровые координаты; и объект класса «*Random*» для создания генератора случайных чисел, который будет генерировать случайные координаты. Поля класса *Ranges.java* представлены в таблице 2.

Таблица 2 – Описание полей класса *Ranges.java*

Тип данных	Название	Описание
<i>Coord</i>	<i>size</i>	Объект класса <i>Coord</i>
<i>ArrayList<Coord></i>	<i>allCoords</i>	Массив координат объектов
<i>Random</i>	<i>random</i>	Объект типа <i>Random</i> , генерирует случайные координаты объектов игры

Класс *Ranges.java* включает в себя 6 методов, описанных ниже.

Static void SetSize() – метод, устанавливающий координаты размера игрового поля.

Public static Coord GetSize() – метод, возвращающий размеры игрового поля.

Public static ArrayList<Coord> GetAllCoords() – метод, возвращающий координаты игровых объектов.

Static boolean InRange() – метод, который проверяет координату на то, находится ли она в пределах игрового поля.

Static Coord GetRandomCoord() – метод, возвращающий координаты для нового объекта.

Static ArrayList<Coord> GetCoordsAround() – метод, который возвращает коллекцию *ArrayList* координат, находящихся вокруг переданной в параметр координаты.

Класс *Matrix.java* хранит матрицу объектов перечисления «*Box*». Конструктор заполняет всю матрицу указанным элементом «*Box*». Метод «*Get*» для получения по координатам конкретного объекта «*Box*». И метод «*Set*» для установки объекта «*Box*» по координатам. Поля класса *Matrix.java* представлены в таблице 3.

Таблица 3 – Описание полей класса *Matrix.java*

Тип данных	Название	Описание
<i>Box[][]</i>	<i>matrix</i>	Массив координат игрового поля

Класс *Matrix.java* включает в себя 3 метода, представленных ниже.

Matrix() – конструктор класса.

Box Get() – метод, возвращающий координаты объекта.

Void Set() – метод, устанавливающий координаты объекта.

Класс *Bomb.java* хранит в себе матрицу «*bombMap*» класса «*Matrix*» с расположением всех бомб и чисел вокруг них. Поле «*totalBombs*» хранит число общего количества бомб. Поля класса *Bomb.java* представлены в таблице 4.

Таблица 4 – Описание полей класса *Bomb.java*

Тип данных	Название	Описание
<i>Matrix</i>	<i>bombMap</i>	Объект класса <i>Matrix</i> , содержащий расположением всех бомб и чисел вокруг них.
<i>int</i>	<i>totalBombs</i>	Число общего количества бомб

Класс *Bomb.java* включает в себя 7 методов описанных ниже.

Bomb() – конструктор класса, устанавливающий общее количество бомб, расположенных на игровом поле.

Void Start() – метод, который инициализирует матрицу картинками пустых клеток и вызывает метод для размещения бомб.

Public int GetTotalBombs() – метод, возвращающий общее количество бомб, расположенных на игровом поле.

Box Get () – метод, необходим для получения конкретного объекта «*Box*» по координатам.

Private void FixBombsCount () – метод, предназначенный для ограничения максимального количества бомб относительно размера поля.

Private void PlaceBomb() – метод, размещающий «Box» с бомбой относительно случайной координаты.

Private void IncNumbersAroundBomb() – метод предназначен для увеличения значения «Box» вокруг бомбы.

Класс *Flag.java* хранит в себе матрицу «*flagMap*» класса «*Matrix*» для работы с верхним слоем поля: закрытые «Box», на которые можно ставить флаги. Поле «*countOfClosedBoxes*» для отслеживания числа закрытых объектов «Box». Поля класса *Flag.java* представлены в таблице 5.

Таблица 5 – Описание полей класса *Flag.java*

Тип данных	Название	Описание
<i>Matrix</i>	<i>flagMap</i>	Матрица для работы с верхним слоем игрового поля
<i>int</i>	<i>countOfClosedBoxes</i>	Число закрытых объектов

Класс *Flag.java* включает в себя 11 методов.

Void Start() – метод, который инициализирует матрицу картинками пустых закрытых клеток и устанавливает их общее количество.

Box Get () – метод, необходимый для получения конкретного объекта «Box» по координатам.

Void SetOpenedToBox() – метод, предназначенный для установки обозначенных «Box» по координатам.

Void ToggleFlagedToBox () – метод, который для переключения между закрытым «Box» и «Box» с флагом.

Void SetFlagedToBox() – метод, предназначенный для установки обозначенных «Box» по координатам.

Void SetClosedToBox() – метод, предназначенный для установки обозначенных «Box» по координатам.

Int GetCountOfClosedBoxes() – метод, возвращающий число закрытых объектов.

Void SetBombedToBox() – метод, устанавливающий флаг на передаваемые координаты.

Void SetOpenedToClosedBombBox() – метод выполняющий установку «Вох» с бомбой в месте, где она была размещена случайным образом.

Void SetNoBombToFlagedSafeBox() – метод, предназначенный для установки «Вох» с отсутствием бомбы в месте, где неверно был размещён «Вох» с флагом.

Int GetCountOfFlagedBoxesAround() – метод, возвращающий числа «Вох» с флагом вокруг указанной координаты.

Перечисление *GameState.java*, где записаны возможные состояния игры. Поля класса *GameState.java* представлены в таблице 6.

Таблица 6 – Описание полей класса *GameState.java*

Название	Описание
<i>PLAYED</i>	Состояние игрока
<i>BOMBED</i>	Состояние бомбы
<i>WINNER</i>	Состояние победы
<i>READY</i>	Состояние готовности

Класс *Game.java* предназначен для управления процессами игры. Объект «*bomb*» класса «*Bomb*» хранит нижний слой поля, где расположены бомбы и цифры. Объект «*flag*» класса «*Flag*» хранит верхний слой поля, ячейки которого можно открывать и ставить на них флаги. Поля класса *Game.java* представлены в таблице 7.

Таблица 7 – Описание полей класса *Game.java*

Тип данных	Название	Описание
<i>Bomb</i>	<i>bomb</i>	Хранит сведения о расположении бомб и их координаты.

Тип данных	Название	Описание
<i>Flag</i>	<i>flag</i>	Хранит сведения о расположении флагов и их координаты.
<i>GameState</i>	<i>gState</i>	Объект класса <i>GameState</i>

Класс *Game.java* включает в себя 19 основных методов.

Public Game() – конструктор класса.

Public Game() – конструктор класса.

Public void StartSolo() – мтеод, инициализирующий матрицы нижнего и верхнего слоя поля и устанавливает состояние игры, используется только на стороне сервера.

Public void StartCoop() – метод, устанавливающий состояние игры, используется на стороне клиента.

Public Bomb GetBomb() – метод, возвращающий бомбу.

Public Flag GetFlag() – метод, возвращающий флаг.

Public void SetState() – метод, устанавливающий текущее состояние игры.

Public void SetBomb() – метод установки бомбы.

Public void SetFlag() – метод установки флага.

Public GameState GetState() – метод возвращающий текущее состояние игры.

Public Box GetBox() – метод для получения конкретного «Box» в указанных координатах.

Public void PressLeftButton() – метод обработки действий при нажатии кнопки мыши.

Public void PressRightButton() – метод обработки действий при нажатии кнопки мыши.

Public void CheckWinner() – метод проверяющий текущее состояние игры.

Public void OpenBox() – метод, который открывает всё игровое поле в случае поражения.

Private void OpenBombs() – метод, который открывает все бомбы, расположенные на игровом поле.

Private void OpenBoxesAround() – метод, раскрывающий часть игрового поля.

Private boolean GameOver() – метод проверяющий текущее состояние игры.

Void SetOpenedToClosedBoxesAroundNumber() – метод для открытия закрытых «Box» вокруг открытой клетки при условии клика на неё, если количество «Box» с флагами равно количеству «Box» с бомбами.

На рисунке 16 представлена UML-диаграмма классов клиента, а на рисунке 17 описана UML-диаграмма классов сервера, демонстрирующие общую структуру игры и взаимодействие между классами.

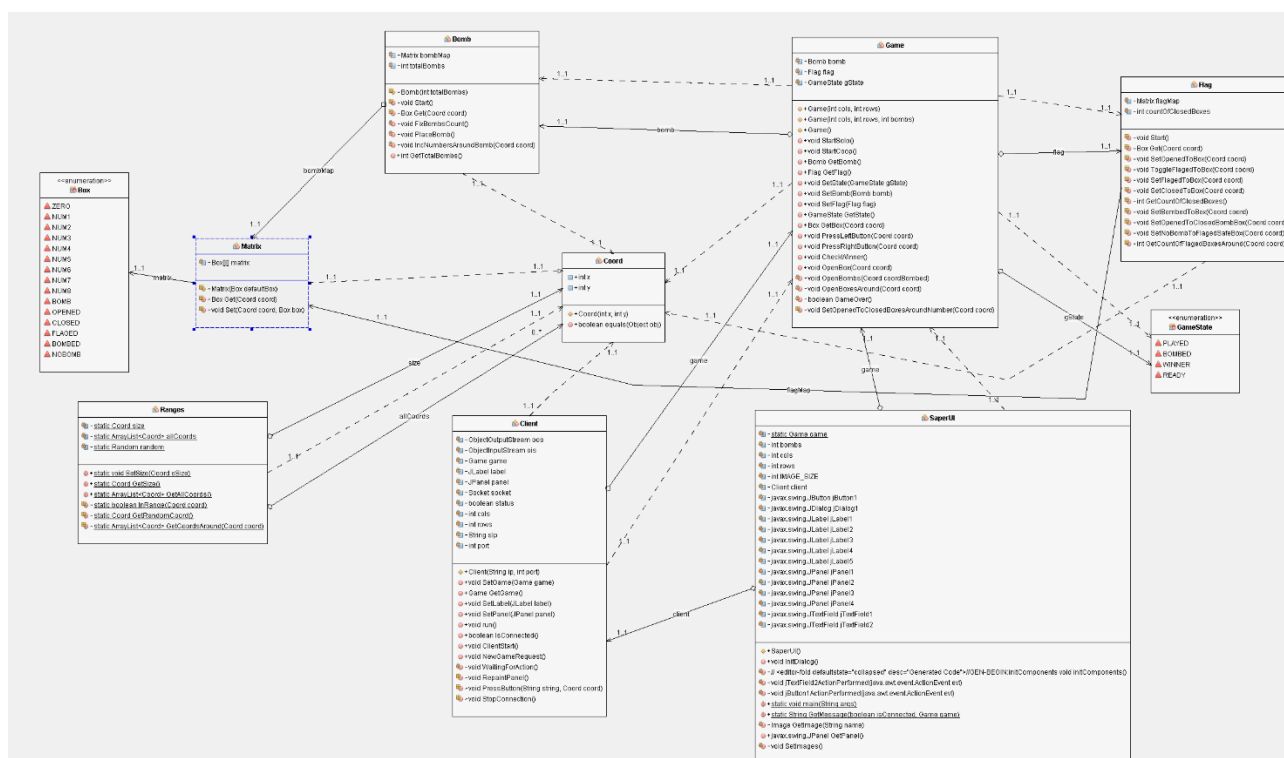


Рисунок 16 – UML-диаграмма классов клиента

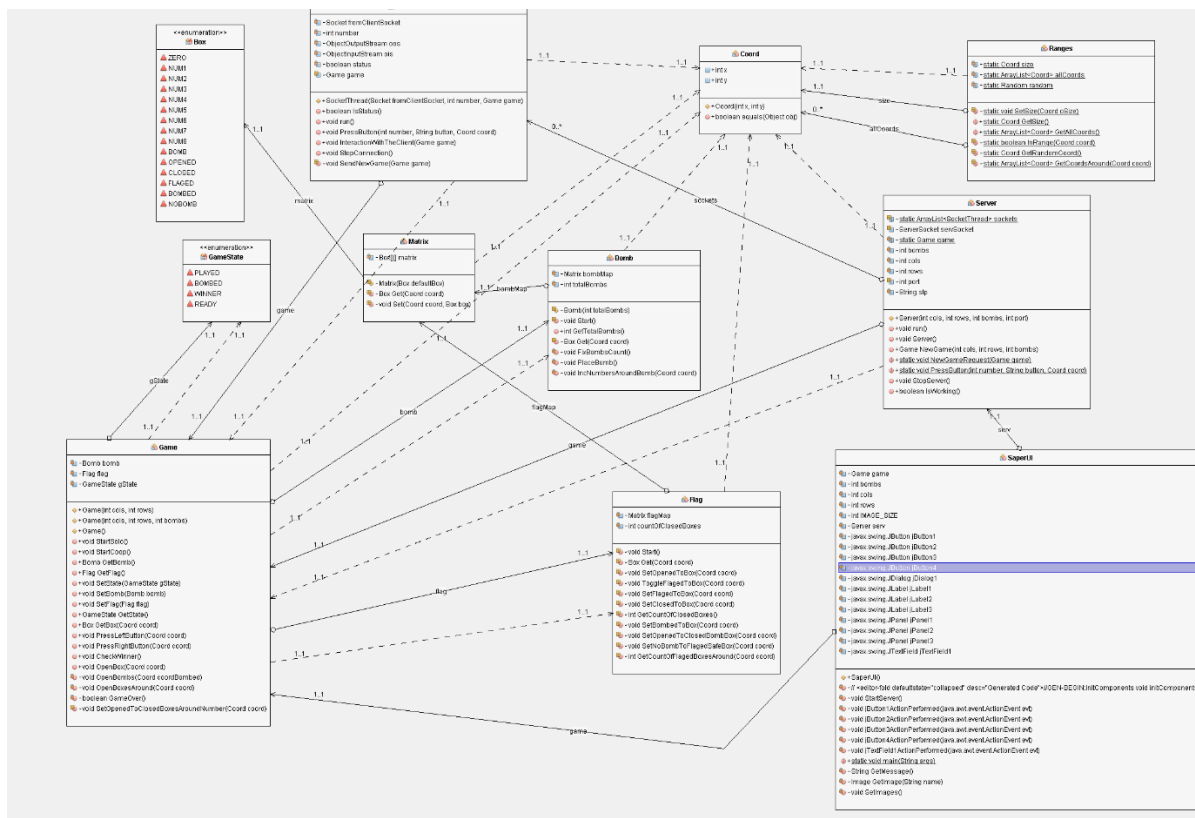


Рисунок 17 – UML-диаграмма классов сервера

1.2.3 Описание способа организации пользовательского интерфейса

Для реализации пользовательского интерфейса была использована библиотека *Swing*. На рисунке 18 показана UML-диаграмма прецедентов.



Рисунок 18 – UML-диаграмма прецедентов

В приложении «SaperKursServer» основной *JFrame* содержит кнопку «*jButton4*» для запуска сервера и панель «*jPanel2*» с меткой «*jLabel1*», которая после попытки включения сервера будут отображать его состояние. Окно *JFrame* сервера представлено на рисунке 19.

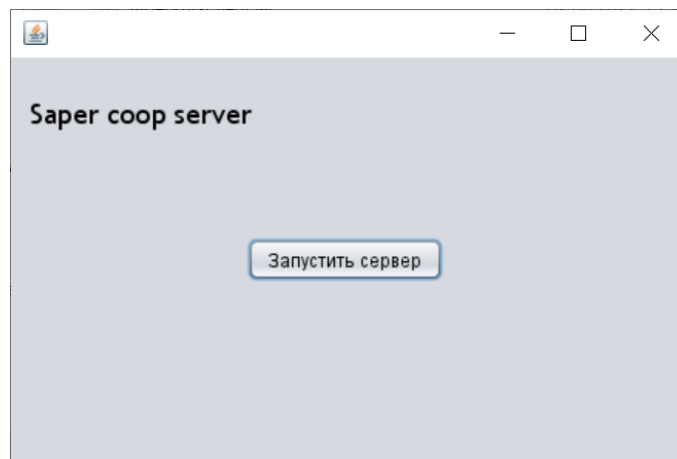


Рисунок 19 – Окно *JFrame* сервера

Нажатие «*jButton4*» вызывает диалоговое окно «*jDialog1*» с двумя панелями. *jPanel1* включает в себя две текстовые метки и текстовое поле «*jTextField1*», откуда в процессе работы будет считываться значение порта. «*jPanel3*» содержит 3 кнопки «*jButton1-3*», являющиеся выбором уровня сложности игры. Диалоговое окно *JDialog1* сервера представлено на рисунке 20.

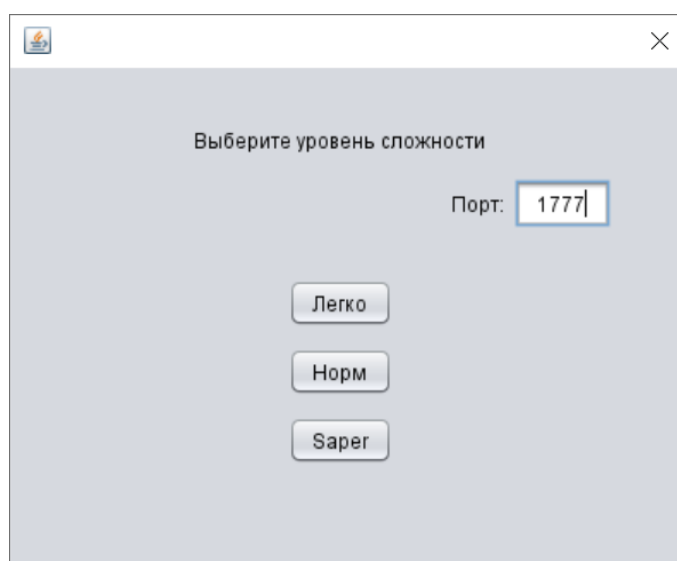


Рисунок 20 – Диалоговое окно *JDialog1*

При запуске приложения «*SaperKursClient*» пользователя сначала встречает диалоговое окно с двумя панелями и кнопкой «*jButton1*». Панель «*jPanel3*» содержит два текстовых поля «*jTextField1*» и «*jTextField2*», которые нужны для ввода адреса сервера и его порта, и две пояснительные текстовые метки. Панель «*jPanel4*» содержит только одну текстовую метку. Диалоговое окно *JDialog2* сервера представлено на рисунке 21.

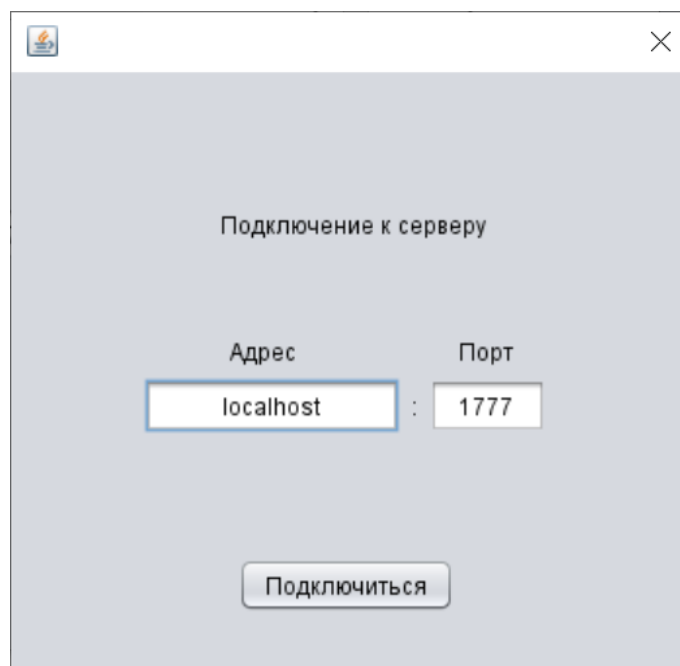


Рисунок 21 – Диалоговое окно *JDialog2*

При корректно введенных значениях адреса и порта, после нажатия «*jButton1*», диалоговое окно закрывается и открывается основной *JFrame* приложения, на панели «*jPanel1*» которого будет размещаться игровое поле, представляющее из себя матрицу картинок размером 50x50 пикселей. Панель «*jPanel2*» содержит одну текстовую метку «*jLabel1*», которая в процессе работы приложения отображает состояние игры и сервера. На рисунке 22 представлено окно *JFrame* игры.

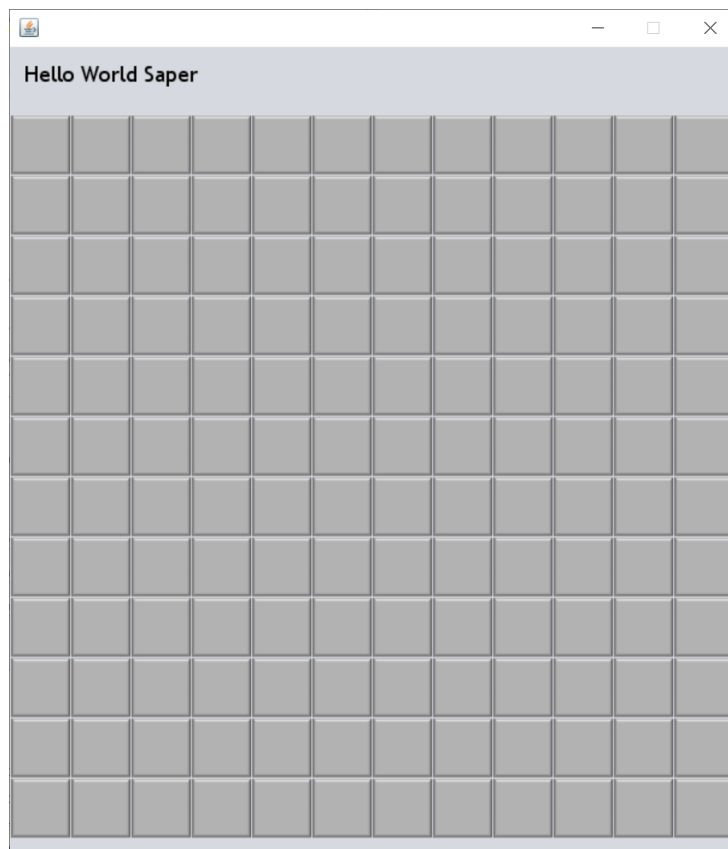


Рисунок 22 – Окно *JFrame* игры

1.3 Тестирование результатов работы

Среда разработки *NetBeans IDE* предоставляет все средства, необходимые при проектировании и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна клиентской части, алгоритмом программы.

Ниже продемонстрирован результат тестирования программы сначала при работе с двумя клиентами, а после и с подключением третьего.

На рисунке 23 показаны окна сервера и клиентов. После нажатия на кнопку «Запустить сервер», открывается начальное окно настроек, в котором можно выбрать уровень сложности игры и задать порт сервера. Клиент имеет на начальном экране поля для ввода адреса и порта сервера, а так же кнопку подключения к серверу.

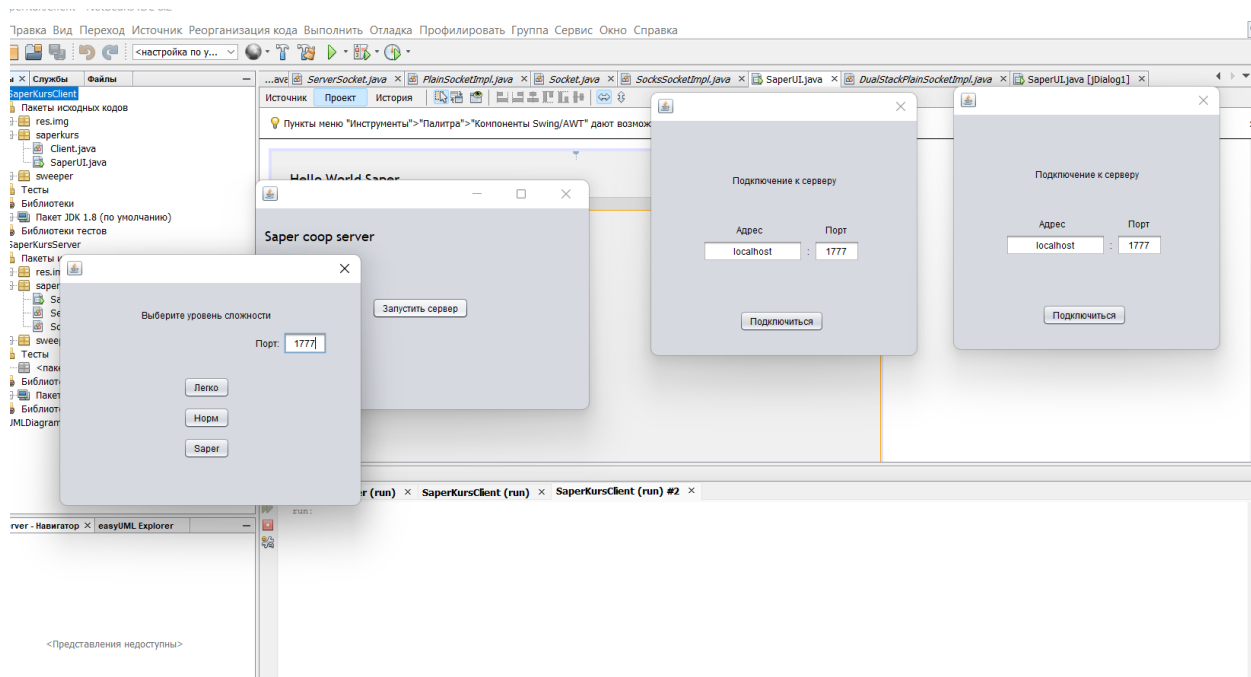


Рисунок 23 – Запуск приложений

На рисунке 24 представлен процесс игры. На сервере была запущена игра с нормальным уровнем сложности. У клиентов отображается одно и тоже поле, в котором они ходят по очереди. Все действия, выполняемые одним клиентом, моментально транслируются у другого клиента.

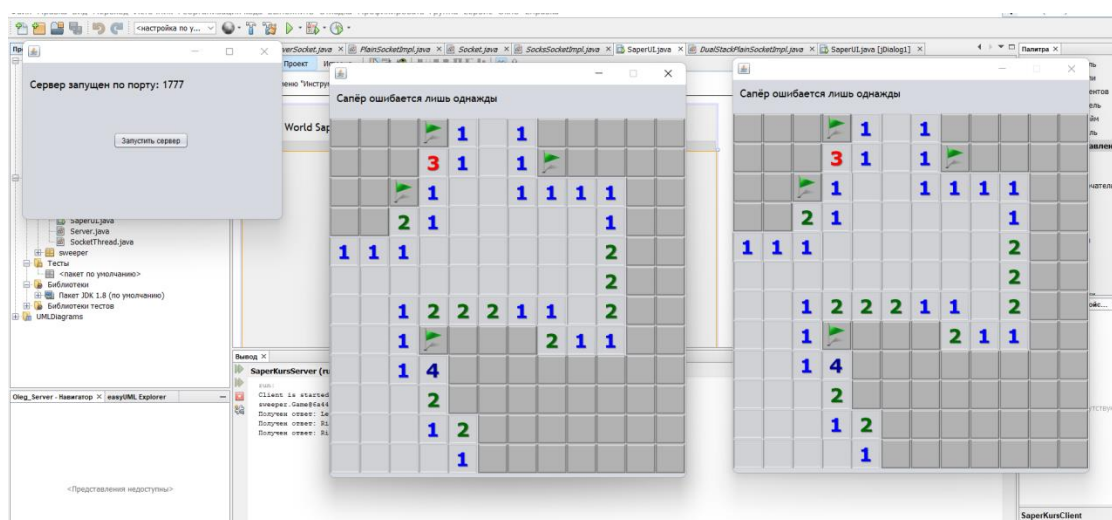


Рисунок 24 – Результаты работы графического интерфейса

На рисунке 25 представлена консоль сервера. В консоль выводится информация о подключении клиентов и логирование ходов каждого клиента.

```

Выход ×
SaperKursServer (run) × SaperKursClient (run) × SaperKursClient (run) #2 ×

run:
Ждём подключения к порту 1777
Ждём подключения к порту 1777
Я живой, я клиент 0
Ждём подключения к порту 1777
Я живой, я клиент 1
Клиент 0 сделал Left на 4 4
Клиент 1 сделал Right на 3 7
Клиент 1 сделал Right на 7 1
Клиент 0 сделал Right на 2 2
Клиент 0 сделал Right на 3 0
|

```

Рисунок 25 – Результаты работы консоли

На рисунках 26 и 27 предстала консоль клиентов, в которой отображается история ходов соперника. Эта информация необходима для отображения выполняемых действий на другом клиенте.

```

Выход ×
SaperKursServer (run) × SaperKursClient (run) × SaperKursClient (run) #2 ×

run:
Client is started
sweeper.Game@6c822e0f
Получен ответ: Right на 3 7
Получен ответ: Right на 7 1

```

Рисунок 26 – Результаты работы консоли

```

Выход ×
SaperKursServer (run) × SaperKursClient (run) × SaperKursClient (run) #2 ×

run:
Client is started
sweeper.Game@6a441084
Получен ответ: Left на 4 4
Получен ответ: Right на 2 2
Получен ответ: Right на 3 0

```

Рисунок 27 – Результаты работы консоли

На рисунке 28 предстален процесс поражения, при игре трех пользователей.

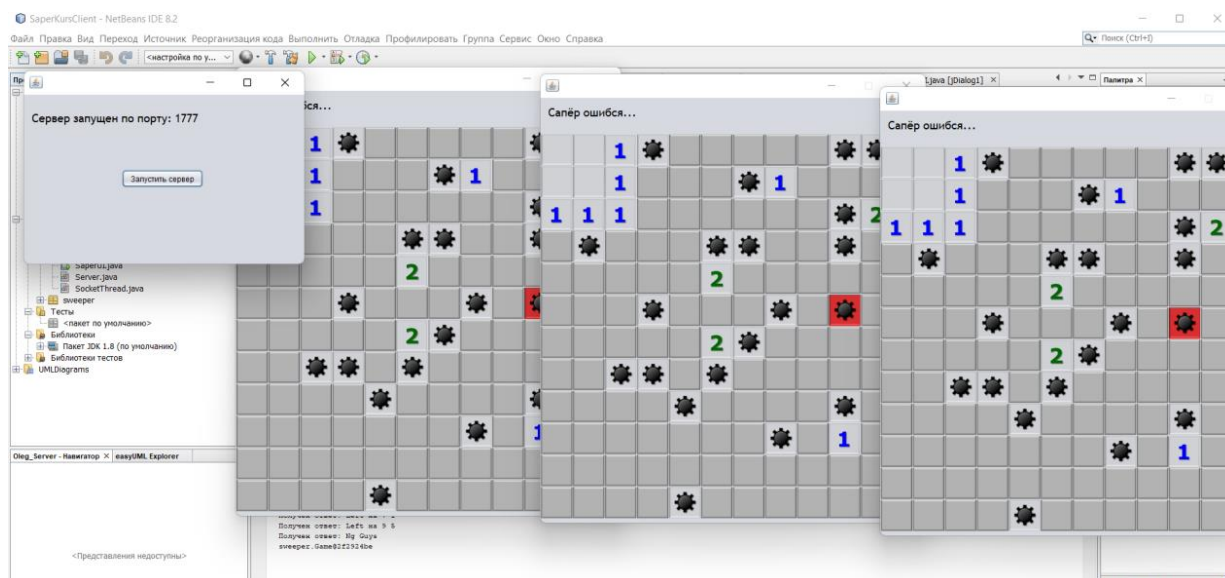


Рисунок 28 – Подключение третьего клиента

Окно сервера при подключении третьего клиента представлено на рисунке 29.

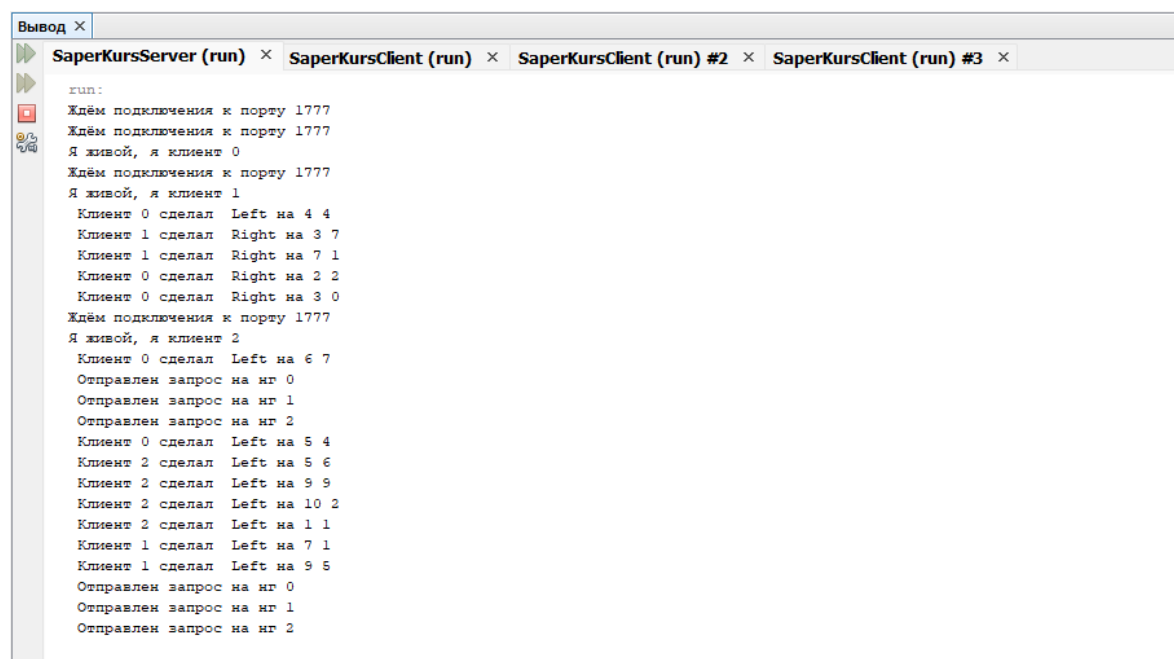


Рисунок 29 – Подключение третьего клиента

На рисунке 30 и 31 представлен процесс выбора более сложного уровня игры и подключение нового клиента, на котором будет запущена новая игра с более сложным уровнем. Уже запущенные при этом игры у клиентов, продолжают работать.

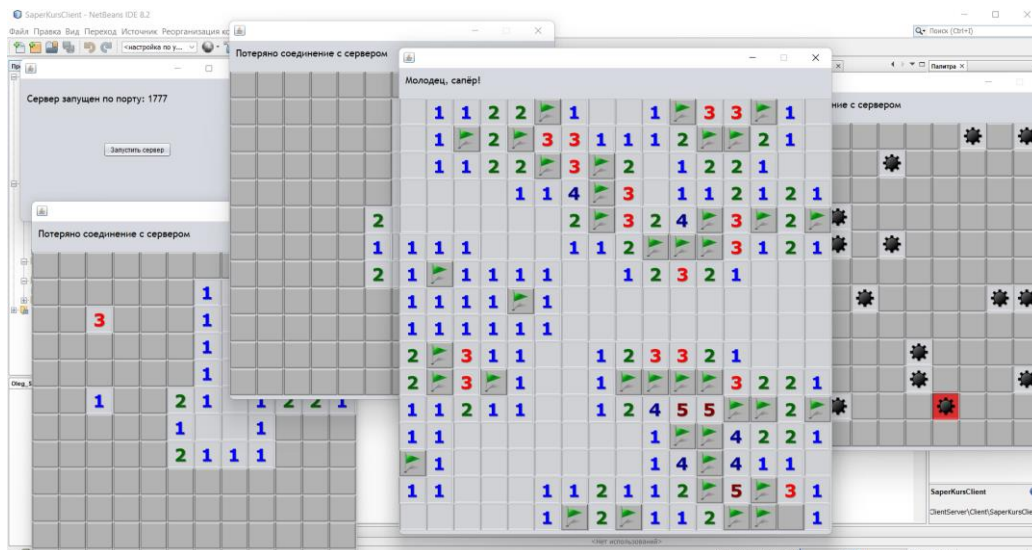


Рисунок 30 – Перезапуск сервера с выбором другого уровня сложности и подключением нового клиента

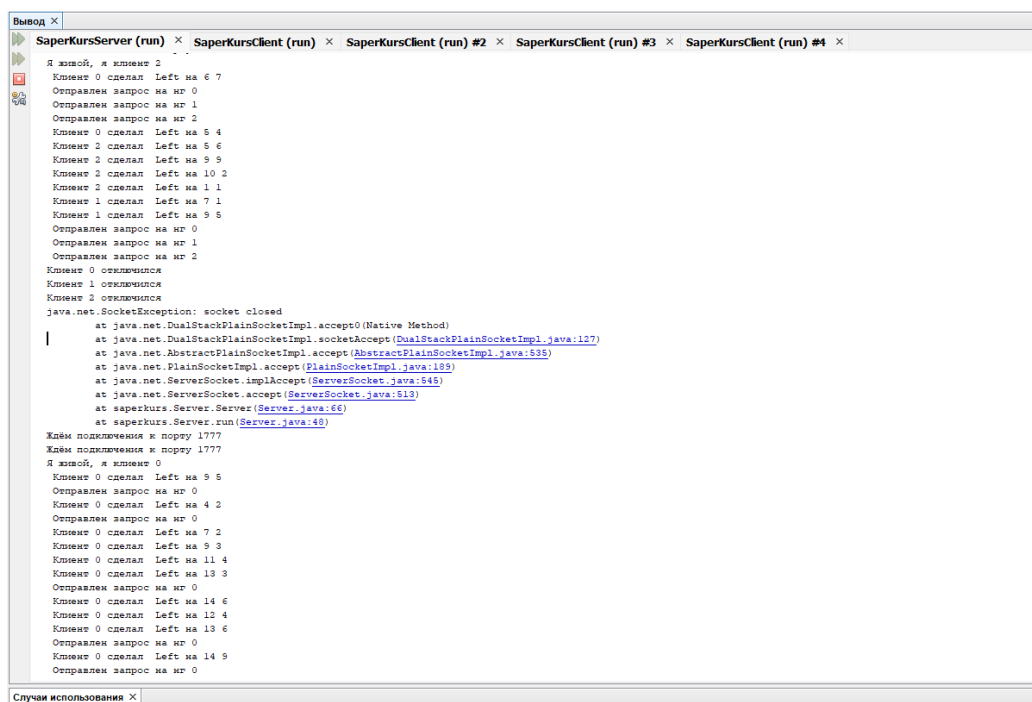


Рисунок 31 – Перезапуск сервера с выбором другого уровня сложности и подключением нового клиента

По итогам тестирования было выявлено, что многомодульное клиент-серверное игровое приложение «Сапёр» успешно выполняет свою работу. На рисунке 32 представлена UML-диаграмма деятельности.

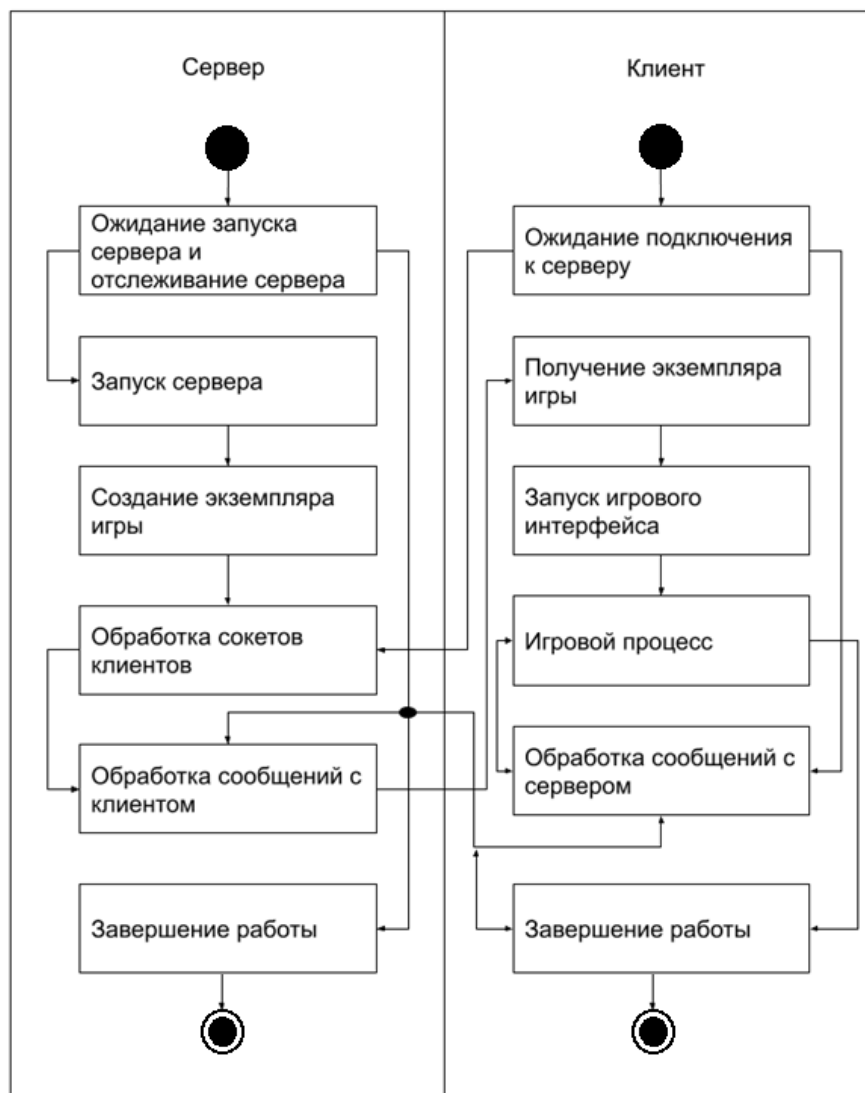


Рисунок 32 – UML-диаграмма деятельности

2 Безопасность жизнедеятельности

Персональная электронная вычислительная машина является источником ряда вредных и опасных факторов производственной среды: электромагнитных полей, статического электричества. Нередко условия труда при работе на ПЭВМ усугубляются повышенными уровнями шума, неудовлетворительными микроклиматическими условиями и недостаточной освещенностью на фоне зрительного и нервно – эмоционального напряжения.

ПЭВМ используются для разработки учебного стимулятора микропрограммируемой ЭВМ, с написанием программного кода, его отладки и компиляции, а также тестированием и оптимизацией. Данные операции характеризуются высокой скоростью переработки информации, высоким темпом работы, низкой потребностью в обмене информацией и сопровождается принятием решений. Работа требует большого умственного и зрительного напряжения, сопровождается локальными мышечными нагрузками.

Особенностью работы на ПЭВМ является постоянное напряжение функций зрительного анализатора, обусловленное необходимостью различения объектов символов, знаков и т.п.

Нервно-эмоциональное напряжение возникает вследствие дефицита времени, большого объема и плотности информации, особенностей диалогового режима общения человека и ПЭВМ, ответственности за безошибочность информации.

Темп работы на ПЭВМ при вводе информации определяется объемом и характером производственного задания, и временем его выполнения. В период выполнения операций ввода данных количество мелких стереотипных движений кистей и пальцев рук за рабочий день может превысить 60 тыс., что в соответствии с гигиенической классификацией труда относится к категории вредных и опасных.

Характеристики помещения отдела офиса приведены в таблице 8.

Таблица 8 – Характеристика помещения отдела офиса

Наименование	Значение
Ширина	4,2 м
Длина	3,8 м
Высота потолка	3,0 м
Площадь помещения	15,96 м ²
Объём помещения	47,88 м ³
Количество рабочих	2 человека
Объём помещения на каждого работающего	23,94 м ³

2.1 Характеристика потенциальных опасностей и вредностей.

Одним из наиболее важных и масштабных направлений в охране трудовой деятельности – это обеспечение максимально комфортных и безопасных для работника факторов при выполнении им функциональных обязанностей.

В процессе реализации разрабатываемого продукта на рабочем месте на программиста могут воздействовать опасные, вызывающие травмы и вредные, вызывающие заболевания производственные факторы. Вредный производственный фактор, в зависимости от интенсивности и продолжительности воздействия, может стать опасным.

Обеспечение безопасной жизнедеятельности программиста в значительной степени зависит от правильной оценки опасных, вредных производственных факторов. Одинаковые по тяжести изменения в организме человека могут быть вызваны различными причинами. Это могут быть какие-либо факторы производственной среды, чрезмерная физическая и умственная нагрузка, нервно-эмоциональное напряжение, а также разное сочетание этих

причин. Рассмотрим подробнее опасные и вредные производственные факторы на рабочем месте программиста:

- повышенный уровень напряжения в электрической цепи, замыкание которой может пройти через тело работающего;
- превышающий уровень шума;
- недостаточная освещенность рабочей зоны;
- повышенная и пониженная температура воздуха;
- чрезмерная запыленность и загазованность воздуха;
- повышенная и пониженная влажность воздуха;
- повышенный уровень ионизирующего излучения;
- повышенный уровень электромагнитных полей;
- повышенный уровень статического электричества;
- опасность поражения электрическим током;
- блеклость экрана дисплея.

Также на программиста влияют опасные и вредные производственные факторы, обладающие свойствами психофизиологического воздействия: умственное перенапряжение, перенапряжение зрительных и слуховых анализаторов, монотонность труда, эмоциональные перегрузки.

Действие указанных неблагоприятных факторов приводит к снижению работоспособности, вызванное развивающимся утомлением.

2.2 Классификация и категорирование помещения

По взрывопожарной и пожарной опасности помещения подразделяются на категории А, Б, В1-В4, Г и Д. Согласно Нормам пожарной безопасности НПБ 105-03, помещения с ЭВМ и ПЭВМ относятся к категории В (пожароопасные), то есть горючие и трудногорючие жидкости, твердые горючие и трудногорючие вещества и материалы, вещества и материалы, способные при взаимодействии с водой, кислородом воздуха или друг с другом только гореть.

Пожары классифицируются по виду горючего материала и подразделяются на следующие классы: пожары твердых горючих веществ и материалов (А), пожары горючих жидкостей или плавящихся твердых веществ и материалов (В), пожары газов (С), пожары металлов (D), пожары горючих веществ и материалов электроустановок, находящихся под напряжением (Е), пожары ядерных материалов, радиоактивных отходов и радиоактивных веществ (F).

Наиболее вероятные классы пожаров в помещениях с ПЭВМ – «А» и «Е», то есть могут гореть в основном твердые вещества, горение которых сопровождается тлением – класс А, или возможны пожары, вызванные возгоранием электрооборудования – класс Е.

Для защиты от поражения человека электрическим током при устройстве помещений необходимо предусматривать меры обеспечения безопасности. С целью их оптимального выбора разработана классификация помещений. Все помещения делятся по степени поражения людей электрическим током на три класса: без повышенной опасности, с повышенной опасностью, особо опасные.

Помещения без повышенной опасности – это сухие, беспыльные помещения с нормальной температурой воздуха и с изолирующими полами.

Помещения с повышенной опасностью характеризуются наличием хотя бы одного из следующих пяти условий: сырости, когда относительная влажность воздуха длительно превышает 75%; высокой температуры, когда температура воздуха свыше суток превышает +35° С; токопроводящей пыли, когда по условиям производства в помещениях выделяется токопроводящая технологическая пыль в таком количестве, что она оседает на проводах, проникает внутрь машин и т. п.; токопроводящих полов; возможности одновременного прикосновения человека к имеющим соединение с землей металлоконструкциям зданий, механизмам и т.п., с одной стороны, и к металлическим корпусам электрооборудования – с другой.

Особо опасные помещения характеризуются наличием одного из следующих трех условий: особой сырости, когда относительная влажность воздуха близка к 100%; помещениями с химически активной или органической средой; одновременного наличия двух и более условий.

Из выше перечисленных условий, можно сделать вывод, что помещение где ведется разработка стимулятора относится к категории помещений без повышенной опасности, так как оно сухое, беспыльное, с нормальной температурой воздуха 21 °С и с изолирующими полами.

2.3 Санитарно-гигиеническая характеристика помещения

Требования к условиям труда программиста, работающего с ПЭВМ и регламентирует СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»; общие рекомендации по электробезопасности, а также особенности воздействия других факторов даны в «Типовой инструкции по охране труда при работе на персональном компьютере» (ТОИ Р-45-084-01).

2.3.1 Требования к помещениям для работы с ПЭВМ

Помещения для работы на ПЭВМ не разрешается размещать в подвалах. Помещения должны иметь оконные проемы для проникновения естественного света, ориентированные преимущественно на север или северо-запад. Окна, следует оборудовать устройствами для регулирования яркости естественного освещения, такими как шторы, жалюзи.

Объем производственного помещения на одного работающего должен составлять не менее 20 м³, площадь – не менее 6,0 м².

Материалы, применяемые для внутренней отделки помещения, должны иметь гигиенический сертификат, подтверждающий их безопасность.

Помещения должны находиться вдали от силовых кабелей, высоковольтных трансформаторов и другого оборудования, способного создавать помехи для офисной техники. В комнатах необходимо

организовать защитное заземление, обеспечивающее безопасную работу компьютеров.

Звукоизоляция помещений должна отвечать гигиеническим требованиям и обеспечивать нормируемые параметры шума.

Поверхность пола должна быть ровной, удобной для очистки и влажной уборки, обладать антистатическими свойствами.

2.3.2 Организация рабочих мест

Эргономическими аспектами организации рабочих мест инженера-программиста являются: высота рабочей поверхности, размеры пространства для ног, расстояние от глаз пользователя до экрана, документа, клавиатуры, характеристики рабочего кресла, требования к поверхности рабочего стола, регулируемость элементов рабочего места. Рабочие условия должны обеспечивать комфортную работу, так как правильная организация рабочего места программиста повышает производительность труда. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям.

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с мониторами должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения и высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5 – 2 м.

Экран видеомонитора должен находиться от глаз на расстоянии 600 – 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Визуальные эргономические параметры ПЭВМ и пределы их изменений, в которых должны быть установлены оптимальные и допустимые диапазоны значений, представлены в таблице 9.

Таблица 9 – Визуальные эргономические параметры и пределы их изменений

Наименование параметров	Пределы значений параметров	
	Минимальные	Максимальные
Яркость знака (яркость фона), кд/куб. м	35	120
Внешняя освещенность экрана ПК	100	250
Угловой размер знака, угл. мин.	16	60

Конструкция рабочей мебели должна обеспечивать возможность индивидуальной регулировки, соответственно росту программиста, и создавать удобную позу. Часто используемые предметы труда и органы управления должны находиться в оптимальной рабочей зоне.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики: ширину – 800, 1000, 1200 и 1400 мм, глубину – 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 – 300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы. Опорное приспособление, позволяющее изменить угол наклона поверхности клавиатуры в пределах от 5 до 15 градусов. Высота среднего ряда клавиш не более 30 мм, с одинаковым ходом для всех клавиш с минимальным сопротивлением нажатия 0,25 Н и максимальным – не более 1,5 Н.

Конструкция рабочего стула (кресла) кресла должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Кресло должно быть снабжено подъемно-поворотным, регулируемым по высоте в пределах 400 – 500 мм, глубине не менее 380 мм и ширине не менее 400 мм. Угол наклона спинки кресла к плоскости сиденья должен изменяться в пределах 90 – 110°. Регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов кресла должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием.

2.3.3 Микроклимат рабочего помещения

Микроклимат рабочего помещения – это климат внутренней среды помещения, который объединяет такие параметры, действующие на организм человека, как температура, относительная влажность и скорость движения воздуха (см. Таблица 2.3).

Офис является помещением Ia категории, где выполняются легкие физические работы, поэтому соблюдаются следующие требования: средняя температура воздуха в помещении офиса составляет +22°C, относительная влажность – 47%, атмосферное давление – 750 мм.рт.ст., содержание пыли – не более 10 мг/м воздуха рабочего места, максимальные размеры частиц – 2 мкм, скорость движения воздуха до 0,1м/с.

Для поддержания оптимального микроклимата, независимо от наружных условий: в холодное время года используется водяное отопление, в теплое время года применяется кондиционирование воздуха. Температура воздуха в помещении офиса не должна опускаться ниже +19°C, и не должна превышать +25°C. Оптимальные нормы микроклимата для помещений с ПЭВМ представлены в таблице 10.

Таблица 10 – Оптимальные нормы микроклимата для помещений с ПЭВМ

Период года	Категория работ	Температура воздуха, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая – 1а	22 – 24	40 – 60	Не более 0,1
	Легкая – 1б	21 – 23	40 – 60	Не более 0,1
Теплый	Легкая – 1а	23 – 25	40 – 60	Не более 0,1
	Легкая – 1б	22 – 24	40 – 60	Не более 0,1

Оптимальные показатели микроклимата обеспечивают состояние нормального функционального и теплового состояния организма человека без напряжения механизма терморегуляции. Они обеспечивают ощущение комфорта и высокую работоспособность сотрудников.

2.3.4 Шум и вибрация

Шум ухудшает условия труда и оказывает вредное действие на организм человека. Вентиляционные установки, кондиционеры и периферийное оборудование для ПЭВМ могут стать источниками шумовых помех на рабочем месте программиста. Длительное воздействие этих шумов отрицательно сказывается на эмоциональном состоянии персонала, они испытывают раздражительность, головокружение, головные боли, снижение памяти, повышенную утомляемость, боли в ушах. Все это снижает работоспособность человека и его производительность, качество и безопасность труда.

Согласно ГОСТ 12.1.003-76 ССБТ эквивалентный уровень шума не должен превышать 50 дБ(А). Длительное воздействие интенсивного шума выше 80 дБ(А) на слух человека приводит к его частичной или полной потере.

Мое рабочее место в офисе с низким уровнем общего шума. В качестве мер по снижению шума предприняты следующие меры:

- облицовка потолка и стен выполнены звукопоглощающим материалом;
- экранирование рабочего места;
- использование в работе оборудования, производящего минимальный шум.

Вибрация в помещении отсутствует.

2.3.5 Ионизирующие и неионизирующие излучения

Основным источником электромагнитных излучений от мониторов ПЭВМ является трансформатор высокой частоты строчной развертки. Конструкция монитора ПЭВМ должна обеспечивать мощность экспозиционной дозы рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса монитора при любых положениях регулирующих устройств и не должна превышать $7,74 \times 10^{-4}$ А/кг, что соответствует эквивалентной дозе, равной 0,1 мбэр/час (100 мкр/час).

Напряженность электромагнитного поля на расстоянии 50 см вокруг ПЭВМ по электрической составляющей должна быть не более:

- в диапазоне частот 5 Гц - 2 кГц - 25 в/м;
- в диапазоне частот 2 - 400 кГц - 2,5 в/м.

Плотность магнитного потока индукции должна быть не более:

- в диапазоне частот 5 Гц - 2 кГц - 250 нТл;
- в диапазоне частот 2 - 400 кГц - 25 нТл.

Поверхностный электростатический потенциал не должен превышать 500 в.

2.3.6 Освещение рабочего места

Правильно спроектированное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на персонал, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах.

Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям.

Рабочие столы следует размещать таким образом, чтобы экраны были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Согласно СанПин 2.2.2/2.4.1340-03 необходимо применить систему комбинированного освещения, так как естественное освещение слабое, на рабочем месте должно применяться и искусственное освещение. Освещение не должно создавать бликов на поверхности экрана ПЭВМ, освещенность не должна быть более 300 лк.

2.4 Электробезопасность

Электробезопасность – это комплекс мер и действий, направленных на минимизацию риска причинения вреда электротоком, а также магнитным и электрическим полями или статическим электричеством.

Офисное оборудование, включая ПЭВМ, относится к электрическим приборам, поэтому работодатель должен обеспечить электробезопасность работников. Во избежание короткого замыкания, а значит, возникновения пожара и получения электротравмы, помещения офиса, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации СанПиН 2.2.2/2.4.1340-03.

Помещение офиса для разработки стимулятора относится к классу помещений без повышенной опасности, помещение сухое, с нормальной температурой воздуха, с токонепроводящим полом, без токопроводящей

пыли и малым числом заземленных приборов. Сопротивление изоляции для таких помещений должно соответствовать норме $R_{\text{изол}}$ больше 0,5 МОм.

Питание оборудования осуществляется электричеством 220В, 50Гц, однофазная сеть с дополнительным заземляющим проводом. Данное помещение относится к 1а классу помещений без повышенной опасности, а также не имеет наружных электроустановок. Из сказанного выше следует, что защитное заземление выполнять не следует. При необходимости, оно должно составлять $R_{\text{заземления}}$ меньше 4 Ом.

В офисе имеется плоттер, сканер и лазерный принтер. Все эти устройства являются законченными изделиями и удовлетворяют требованиям электробезопасности. Вероятность поражения человека электрическим током при правильном использовании оргтехники практически сведена к нулю.

Электробезопасность в помещении офиса обеспечивается техническими способами и средствами защиты, а также организационными и техническими мероприятиями.

Разберем основные причины поражения человека электрическим током на рабочем месте при разработке симулятора:

- прикосновение к металлическим нетоковедущим частям, таким как корпус, периферия компьютера, которые могут оказаться под напряжением в результате повреждения изоляции;
- при касании незащищенных контактов, в случае разборки ПЭВМ;
- нерегламентированное использование электрических приборов;
- отсутствие инструктажа сотрудников по правилам электробезопасности.

При работе ПЭВМ, на его корпусе накапливается статическое электричество. На расстоянии 5 – 10 см от экрана напряженность электростатического поля составляет 60 – 280 кВ/м, то есть в 10 раз превышает норму 20 кВ/м. Для уменьшения напряжённости применять применение увлажнители и нейтрализаторы, антистатическое покрытия пола.

2.5 Пожарная безопасность

Пожарная безопасность – это вероятность исключения возможности возникновения и развития пожара и воздействия на людей опасных факторов пожара, а также обеспечивается защита материальных ценностей [5].

Основными причинами пожара при работе программиста являются: короткое замыкание, плавление изоляции проводов, их оголение, и как следствие, короткое замыкание, которое сопровождается искрением и ведет к перегрузкам элементов электронных схем, которые, перегреваясь, сгорают с искрением, поэтому следует большое внимание оказывать пожарной безопасности.

Помещение, где установлены ПЭВМ, относится к категории “В” - пожароопасные.

В соответствии с требованиями пожарной безопасности в помещении установлена пожарная сигнализация, в отделе офиса имеются дымовые извещатели в количестве 2 штук. Количество извещателей выбирается исходя из площади помещения, высоты потолков и требований по установке извещателей, для высоты потолка до 3,5 м.

Для тушения пожаров в начальной стадии развития могут быть использованы огнетушители углекислотный ОУ-5 и порошковый ОП-5-01 предназначенный для тушения электроустановок, находящихся под напряжением до 1000 В, применяется для тушения ПЭВМ и оргтехники. При входе в помещение отдела стоит один огнетушитель ОУ-5, из расчета один огнетушитель на 40 – 50 м² площади, но не менее двух в помещении.

Для ликвидации пожара в начальной стадии в коридоре офиса размещается пожарный кран. В отделе, где установлена вычислительная техника, недопустимо применять воду и пенные огнетушители, так как в этом случае существует опасность повреждения или полного выхода из строя ПЭВМ и другого оборудования.

Для профилактики возникновения пожара необходимо вовремя выявлять и устранять неисправности, проводить плановый осмотр и своевременно устранять все неисправности электрооборудования, электропроводки, электрических розетках и выключателях. Выполнять своевременный ремонт электроприборов, качественное исправление поломок. В целях уменьшения вероятности возникновения пожара вследствие короткого замыкания необходимо, чтобы электропроводка была скрытой.

Мебель в помещениях не должна препятствовать быстрой эвакуации людей. Расположение электрических кабелей и различных проводов должно исключать их повреждение, поражение работников электрическим током, а также они не должны мешать передвижению по помещению.

В летний период во время грозы возможно попадание молнии вследствие чего возможен пожар. Во избежание этого необходимо установить на крыше здания молниеотвод.

В целях предотвращения пожара рекомендуется проводить с программистами, работающими в офисе, противопожарный инструктаж, на котором ознакомить с правилами противопожарной безопасности, а также обучить использованию первичных средств пожаротушения.

В случае возникновения пожара необходимо отключить электропитание, вызвать по телефону пожарную команду, эвакуировать людей из помещения согласно плану эвакуации, и приступить к ликвидации пожара огнетушителями. При наличии небольшого очага пламени можно воспользоваться подручными средствами с целью прекращения доступа воздуха к объекту возгорания.

2.6 Расчет общего искусственного освещения помещения

Целью расчета является определение числа и расположения светильников, а также числа и мощности ламп, необходимых для обеспечения требуемого нормативного уровня освещенности для работы программиста на ПЭВМ.

Расчет искусственного освещения будет проводиться для помещения площадью $S = 15,96 \text{ м}^2$, длина $a = 3,8 \text{ м}$, ширина $b = 4,2 \text{ м}$, высота $h = 3,0 \text{ м}$.

В соответствии с СанПиНом 2.2.2/2.4.1340-03 «Гигиенические требования к ПЭВМ и организации работы» освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 – 500 лк. Для расчета требуемый нормативный уровень освещенности E_n выберем равной 400 лк [6].

Определяем разряд и подразряд зрительной работы, нормы освещённости на рабочем месте по таблице 4: характеристика зрительной работы – высокой точности, разряд – 3, подразряд – г, комбинированное освещение – 400 лк, общее освещение – 200 лк.

Размещение светильников определяется следующими размерами:

Высота светильников над полом рассчитывается по формуле (2.1):

$$h_{\text{п}} = H - h_{\text{с}}, \quad (2.1)$$

где, $H = 3 \text{ м}$ высота помещения;

$h_{\text{с}} = 0,25 \text{ м}$ – расстояние светильников от перекрытия.

Расчёт высоты от пола до светильника по формуле (2.1):

$$h_{\text{п}} = 3 - 0,25 = 2,75 \text{ м}.$$

Расчет высоты от рабочей поверхности до светильника по формуле (2.2):

$$h = h_{\text{п}} - h_{\text{р}}, \quad (2.2)$$

где, $h_{\text{п}}$ – высота от пола до светильника;

$h_{\text{р}}$ – высота расчетной поверхности равна 0,7 м (для помещений, связанных с работой ПЭВМ).

Математический расчет высоты от поверхности до светильников выполняется по формуле (2.2):

$$h = 2,7 - 0,7 = 2,05 \text{ м.}$$

Рассчитываем число светильников N по формуле (2.3):

$$N = \frac{S}{L \cdot M}, \quad (2.3)$$

где, S – площадь помещения;

L – расстояние между центрами светильников;

M – расстояние между параллельными рядами.

Площадь помещения находится по формуле (2.4):

$$S = a \cdot b, \quad (2.4)$$

где, a – длина;

b – ширина.

Расстояние между центрами светильников рассчитывается по формуле (2.5):

$$L = 1,75 \cdot H. \quad (2.5)$$

Выполним расчеты по формулам (2.4) и (2.5):

$$S = 4,2 \cdot 3,8 = 15,96 \text{ м}^2;$$

$$L = 4 \cdot 1,75 = 5,25 \text{ м.}$$

Рассчитаем расстояние между параллельными рядами – M по формуле (2.6):

$$M = 0,6 \cdot H_p, \quad (2.6)$$

где, $H_p = H$.

Исходя из формулы (2.6) принимаем $M = 2$ м. Подставим полученные значения в формулу (2.3):

$$N = \frac{15,96}{5,25 \cdot 2} = 1,5.$$

Таким образом, принимаем $N = 2$ шт.

Определим расчетный световой поток по формуле (2.7):

$$\Phi = \frac{E_n \cdot S \cdot Z \cdot K}{N \cdot \mu}, \quad (2.7)$$

где, Z – коэффициент неравномерности освещения равный 1,1;

K – коэффициент учитывающий загрязнение светильника равный 1,5;

E_n – нормируемая минимальная освещенность равная 300;

μ – коэффициент использования светового потока равный 0,61.

Произведем математический расчет светового потока по формуле (2.7):

$$\Phi = \frac{300 \cdot 15,96 \cdot 1,1 \cdot 1,5}{2 \cdot 0,61} = 6\,475 \text{ лм.}$$

Индекс помещения определяется по формуле (2.8):

$$i = \frac{a \cdot b}{h_p \cdot (a + b)}, \quad (2.8)$$

где, a – ширина помещения;

b – длина помещения;

h_p – высота рабочей поверхности над полом.

Подставляя значение в формулу (2.8) найдем индекс помещения:

$$i = \frac{4,2 \cdot 3,8}{0,7 \cdot (4,2 + 3,8)} = 2,85.$$

Зная индекс помещения и коэффициенты отражения, из таблицы коэффициентов использования определяем $n = 0,46$.

Подставляем полученные значения в формулу (2.3) для определения количества светильников:

$$N_{cu} = \frac{400 \cdot 15,96 \cdot 1,1 \cdot 1,5}{6475 \cdot 2 \cdot 0,46} = 1,77.$$

Таким образом, принимаем количество светильников равным 2, каждый из которых укомплектован двумя люминесцентными лампами.

В данном разделе дипломной работы был проведен анализ вредных и опасных производственных факторов, действующих на рабочем месте инженера-программиста. На основании изученной литературы по данной проблеме, были указаны оптимальные размеры рабочего стола и кресла, а также сформулированы предложения по улучшению параметров рабочего места. Соблюдение условий, определяющих оптимальную организацию рабочего места инженера-программиста, позволит сохранить хорошую работоспособность в течение всего рабочего дня, повысит как в количественном, так и в качественном отношении производительность труда программиста, что в свою очередь будет способствовать быстрой разработке и отладке программного продукта.

Так как работа программиста по виду трудовой деятельности относится к группе В – творческая работа в режиме диалога с ПЭВМ, а по напряженности работы ко II категории тяжести (СанПиН 2.2.2.542-96), необходимо сократить время работы за ПЭВМ, делать перерывы, суммарное время которых должно составлять 50 минут при 8-ми часовой смене. Площадь рабочего помещения составляет 7,9 м² и объем помещения на одного работающего составляет 23,94 м³, что соответствует нормам СанПин. Созданные условия должны обеспечивать комфортную работу программиста.

Заключение

В ходе выполнения данной работы было создано кроссплатформенное клиент – серверное приложение «Сапёр».

В ходе выполнения проекта решался ряд задач, среди которых необходимо выделить следующие:

- анализ разработки кроссплатформенных приложений;
- определение набора задач для разрабатываемого приложения;
- выбор средств разработки приложения;
- написание программного кода;
- создание графеской части приложения;
- проектирование структурных схем функционирования приложения.

Были определены основные функции, которые должны быть реализованы в современном кроссплатформенное клиент – серверное приложении, что позволило перейти к реализации проектных решений. Приложение полностью соответствует всем заявленным требованиям.

В разработанном приложении реализованы актуальные технические решения. За время проектирования и реализации данного проекта была освоена интегрирования среда разработки приложений *NetBeans*, улучшены навыки разработки приложений на языке программирования *Java*.

Приложение было протестировано на различных устройствах и ОС, приложение работает корректно и может использоваться игроком.

Список использованных источников

1. Что такое нативные и кроссплатформенные приложения? [Электронный ресурс] / Режим доступа: <https://itvdn.com/ru/blog/article/native-cross-platform>
2. Плюсы и минусы кроссплатформенной разработки? [Электронный ресурс] / Режим доступа: <https://merehead.com/ru/blog/pros-and-cons-cross-platform-native-mobile-app-development/>
3. Эккель, Б. Философия Java. – 4-е полное издание – СПб.: Питер, 2016. – 1168 с.: ил.
4. Справочная система компании Oracle. Материалы сайта <https://docs.oracle.com/> – Режим доступа: <https://docs.oracle.com/>
5. Нормы пожарной безопасности НПБ 105-03 "Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности" / утв. Приказом №314 МЧС РФ от 18 июня 2003 г.
6. СанПиН 2.2.2_2.4.1340-03 гигиенические требования к ПЭВМ и организации работы / Министерство здравоохранения РФ, 2003.

Приложение А

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма классов клиента

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

Приложение Б

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма классов сервера

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

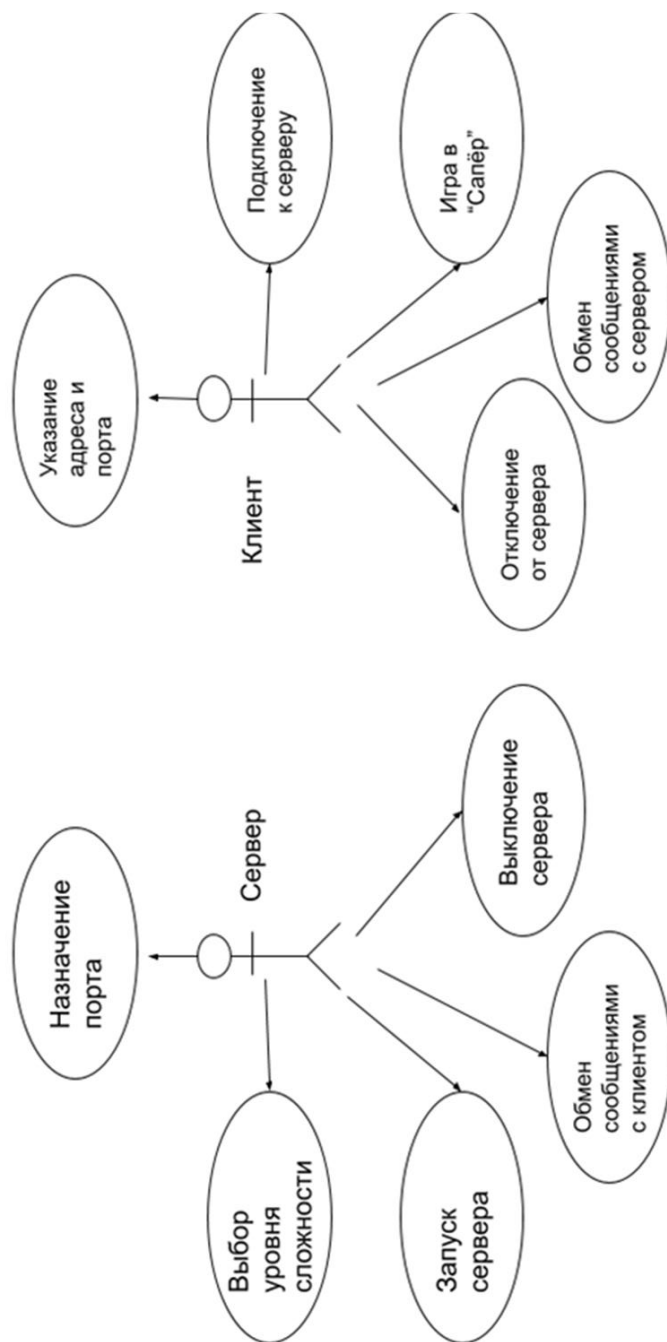
Приложение В

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма прецедентов

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66



ПГУ 1.090301.45.004 ПЗ									
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.
Диагност.	Учебная	Порт	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.	Диагност.

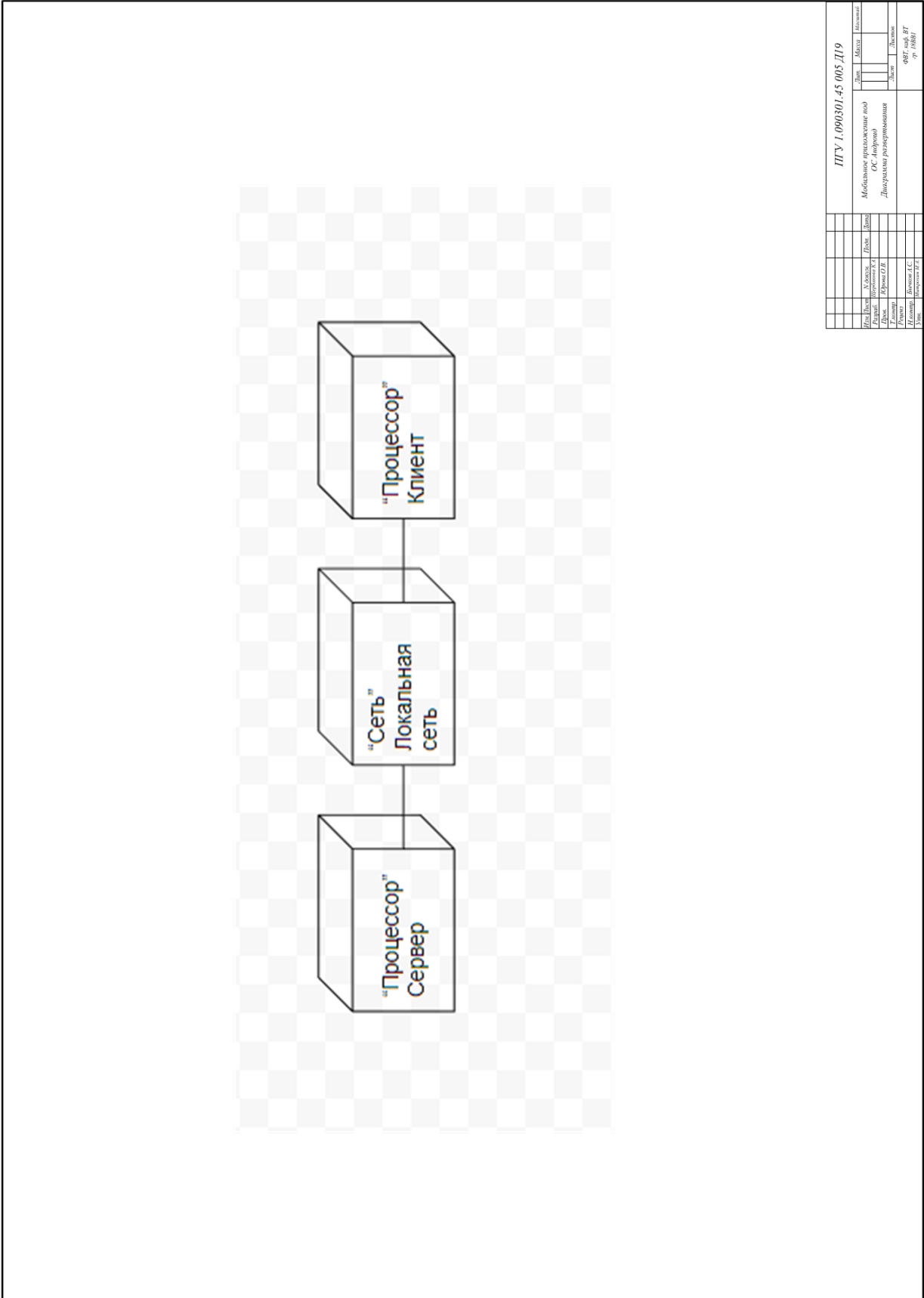
Приложение Г

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма развертывания

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		68



ПГУ 1.090301.45.005 П19									
Мобильное приложение под ОС Android					Документы регламентирующие				
Дет.	Метод	Исход.	Дет.	Исход.	Дет.	Метод	Исход.	Дет.	Исход.
ФОРМЫ ИД. ИД. ИД. ИД. ИД. ИД. ИД. ИД. ИД. ИД.									

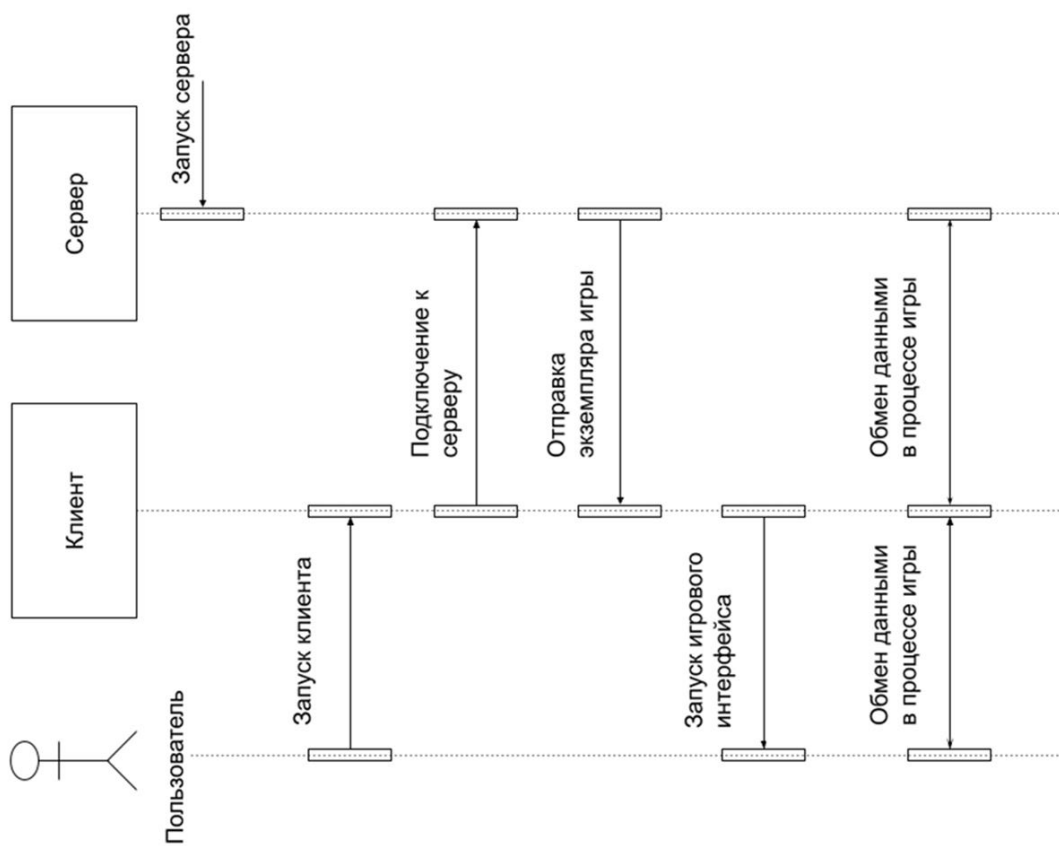
Приложение Д

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма последовательностей

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70



ПГУ 1.090301.45.006.П14									
Мобильное приложение под ОС Android									
последовательностей									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Должность	Должность	Должность	Должность	Должность	Должность	Должность	Должность	Должность	Должность
Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата

Приложение Е

(Обязательное)

Кроссплатформенное клиент-серверное приложение

UML-диаграмма деятельности

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

Приложение Ж

(Обязательное)

Кроссплатформенное клиент-серверное приложение

Листинг программы

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

Файл *SaperUI.java*

```
package saperkurs;

import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SaperUI extends javax.swing.JFrame {

    private int bombs;
    private int cols;
    private int rows;

    private Server serv;

    public SaperUI() {

        initComponents();

        this.addWindowListener(new WindowAdapter()
        {
            @Override
            public void windowClosing(WindowEvent e)
            {
                if(serv != null){
                    if(serv.IsWorking()){
                        serv.StopServer();
                    }
                    System.exit(0);
                }
            }
        });

        @SuppressWarnings("unchecked")
        // <editor-fold defaultstate="collapsed" desc="Generated
        Code">
        private void initComponents() {

            jDialog1 = new
            javax.swing.JDialog((java.awt.Dialog)null);
            jPanel1 = new javax.swing.JPanel();
            jLabel2 = new javax.swing.JLabel();
            jTextField1 = new javax.swing.JTextField();
            jLabel3 = new javax.swing.JLabel();
            jPanel3 = new javax.swing.JPanel();
            jButton1 = new javax.swing.JButton();
            jButton2 = new javax.swing.JButton();
```

					ИПУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

```

        jButton3 = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jButton4 = new javax.swing.JButton();

        jLabel2.setText("Выберите уровень сложности");

jTextField1.setHorizontalAlignment(javax.swing.JTextField.CENTER
);
        jTextField1.setText("1777");
        jTextField1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel3.setText("Порт:");

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(64, 64, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.TRAILING)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
181, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(26, 26, 26)
            );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
33, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel3)
            );

```

```

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
out.Alignment.BASELINE)
.addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabel3))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
);

jButton1.setText("Лерко");
jButton1.addActionListener(new
java.awt.event.ActionListener() {
    public void
actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("HopM");
jButton2.addActionListener(new
java.awt.event.ActionListener() {
    public void
actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Saper");
jButton3.addActionListener(new
java.awt.event.ActionListener() {
    public void
actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
.addContainerGap(29, Short.MAX_VALUE)
.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.
out.Alignment.LEADING, false)
.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

```

```

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addGap(27, 27, 27))
    );
    jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
.addGroup(jPanel3Layout.createSequentialGroup())
.addContainerGap()
.addComponent(jButton1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRE
LATED)
.addComponent(jButton2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELA
TED, 12, Short.MAX_VALUE)
.addComponent(jButton3)
.addContainerGap())
    );

javax.swing.GroupLayout jDialog1Layout = new
javax.swing.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jDialog1Layout.createSequentialGroup())
.addContainerGap(43, Short.MAX_VALUE)
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(39, 39, 39))
.addGroup(jDialog1Layout.createSequentialGroup())
.addGap(133, 133, 133)
.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
    jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
.addGroup(jDialog1Layout.createSequentialGroup())
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)

```

```

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(55, Short.MAX_VALUE)
    );

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setSize(new java.awt.Dimension(0, 0));

        jLabel1.setFont(new java.awt.Font("Trebuchet MS", 0,
18)); // NOI18N
        jLabel1.setText("Saper coop server");

javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel2Layout.createSequentialGroup()
.addContainerGap()
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
389, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(31, Short.MAX_VALUE)
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel2Layout.createSequentialGroup()
.addContainerGap()
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
48, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(25, Short.MAX_VALUE)
        );

        jButton4.setText("Запустить сервер");
        jButton4.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

```

```

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addContainerGap())
.addGroup(layout.createSequentialGroup()
.addGap(153, 153, 153)
.addComponent(jButton4)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(32, 32, 32)
.addComponent(jButton4)
.addContainerGap(119, Short.MAX_VALUE))
);

pack();
    setLocationRelativeTo(null);
} // </editor-fold>

private void StartServer(){

if(serv != null){
    if(serv.IsWorking()){
serv.StopServer();
    }
}

serv = new Server(cols, rows, bombs,
Integer.parseInt(jTextField1.getText()));
serv.start();

try {
    Thread.sleep(55);
} catch (InterruptedException ex) {

Logger.getLogger(SaperUI.class.getName()).log(Level.SEVERE,
null, ex);
}

if(serv.IsWorking()){
jLabel1.setText("Сервер запущен по порту: " +
jTextField1.getText() );
}

```

```

        jLabel1.repaint();
    }
    else
    {
        jLabel1.setText("Ошибка при запуске сервера");
        jLabel1.repaint();
    }

}

private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    cols = 9;
    rows = 9;
    bombs = 10;
    StartServer();
    if(serv.IsWorking()){

        jDialog1.dispose();
        repaint();

    }

}

private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    cols = 12;
    rows = 12;
    bombs = 20;
    StartServer();

    if(serv.IsWorking()){

        jDialog1.dispose();
        repaint();

    }

}

private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    cols = 16;
    rows = 16;
    bombs = 40;
    StartServer();

    if(serv.IsWorking()){

        jDialog1.dispose();

```

```

repaint();

    }

}

private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    jDialog1.pack();
    jDialog1.setLocationRelativeTo(null);
    jDialog1.setVisible (true);
}

private void
jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

```

```

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SaperUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}

```

Файл *Server.java*

```

package saperkurs;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import sweeper.Coord;
import sweeper.Game;
public class Server extends Thread {

    static ArrayList<SocketThread> sockets;
    ServerSocket servSocket;
    private static Game game;
    private int bombs;
    private int cols;
    private int rows;
    int port;
    private String sIp;

    public Server(int cols, int rows, int bombs, int port) {
        this.cols = cols;
        this.rows = rows;
        this.bombs = bombs;
        // sIp = ip;
    }
}

```

```

this.port = port;

    }

    public void run() {
Server();

    }

    public void Server() {

        // port = 1777;

        try {

            servSocket = new ServerSocket(port);

            sockets = new ArrayList<SocketThread>();
NewGame(cols, rows, bombs);

            int countSocket = 0;
            while (true) {
                System.out.println("Ждём подключения к порту " +
port);
                Socket fromClientSocket = servSocket.accept();
if(IsWorking()){
sockets.add(new SocketThread(fromClientSocket, countSocket,
game));

                sockets.get(countSocket).start();

                countSocket++;

            }
else{

                return;

            }

        } catch (IOException ex) {
ex.printStackTrace(System.out);
        }

    }

    public Game NewGame(int cols, int rows, int bombs){

        game = new Game(cols, rows, bombs);
game.StartSolo();

        return game;

    }

    public static void NewGameRequest(Game game) {

        Server.game = game;

```

```

for(int i = 0; i < sockets.size(); i++){

    if(sockets.get(i).IsStatus() == true){
        System.out.println(" Отправлен запрос на нг
" + i);
sockets.get(i).SendNewGame(game);
    }

}

    public static void PressButton(int number, String button,
Coord coord) {

        System.out.println(" Клиент " + number + " сделал " + " "
+ button + " на " + coord.x + " " + coord.y);

for(int i = 0; i < sockets.size(); i++){

if(i != number){
    if(sockets.get(i).IsStatus() == true){
//  System.out.println(" Обнаружен клиент " + i);
sockets.get(i).PressButton(number, button, coord);
    }

}

}

    public void StopServer() {

if(IsWorking()){
for(int i = 0; i < sockets.size(); i++){

        if(sockets.get(i).IsStatus() == true){
            sockets.get(i).StopConnection();
        }

    }

    try {
        servSocket.close();
    } catch (IOException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null,
ex);

    }

}

    public boolean IsWorking(){

if((servSocket != null))

```

```

return !servSocket.isClosed();
    else
        return false;
}
}

```

Файл *SocketThread.java*

```

package saperkurs;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import sweeper.Coord;
import sweeper.Game;
import sweeper.Ranges;

class SocketThread extends Thread
{
    private Socket fromClientSocket;
    private int number;
    private ObjectOutputStream oos;
    private ObjectInputStream ois;
    private boolean status;
    Game game;

    public boolean IsStatus() {
        return status;
    }

    public SocketThread(Socket fromClientSocket, int number,
        Game game) {
        this.fromClientSocket = fromClientSocket;
        this.number = number;
        this.game = game;
        status = true;
    }

    @Override
    public void run() {
        InteractionWithTheClient(game);
    }

    public void PressButton(int number, String button, Coord
        coord) {
        try {
            oos.writeObject(button);
            oos.writeObject(coord);
        } catch (IOException ex) {

```

```

    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    public void InteractionWithTheClient(Game game) {

        try {

            oos = new
ObjectOutputStream(fromClientSocket.getOutputStream());
            ois = new
ObjectInputStream(fromClientSocket.getInputStream());

            oos.writeInt(Ranges.GetSize().x);
            oos.writeInt(Ranges.GetSize().y);

            oos.writeObject(game);

            String str;

            Coord coord = new Coord(0,0);

            System.out.println("Я живой, я клиент " + number);

            while ((str = (String) ois.readObject()) != null) {

                switch(str){

                    case "Left":
                        coord = (Coord) ois.readObject();

                        Server.PressButton(number, str, coord);

                        break;

                    case "Right":
                        coord = (Coord) ois.readObject();

                        Server.PressButton(number, str, coord);

                        break;

                    case "NgPls":

                        int cols = Ranges.GetSize().x;
                        int rows = Ranges.GetSize().y;
                        int bombs = game.GetBomb().GetTotalBombs();

                        game = new Game(cols, rows, bombs);

```

```

game.StartSolo();
                                Server.NewGameRequest(game);

                                break;

                                case "Exit":
                                    if(status)
StopConnection();
                                return;

                                default:

                                System.out.println(" Получен необработанный
ответ: " + str);

                                break;

                                }

                                }
                                } catch (IOException ex) {

Logger.getLogger(SocketThread.class.getName()).log(Level.SEVERE,
null, ex);
                                }
                                catch (ClassNotFoundException ex) {

Logger.getLogger(SocketThread.class.getName()).log(Level.SEVERE,
null, ex);
                                }
                                }

                                public void StopConnection(){

if(status == true){

                                try {
                                    status = false;
oos.writeObject("Exit");
                                    try {
                                        Thread.sleep(120);
                                    } catch (InterruptedException ex) {

Logger.getLogger(SocketThread.class.getName()).log(Level.SEVERE,
null, ex);

                                    }

oos.close();
ois.close();

                                    fromClientSocket.close();
                                    System.out.println("Клиент " + number + "
отключился");
                                    } catch (IOException ex) {

```

```

        Logger.getLogger(SocketThread.class.getName()).log(Level.SEVERE,
        null, ex);
    }
}

void SendNewGame(Game game) {
    try {
        oos.writeObject("Ng Guys");
        oos.writeObject(game);
    } catch (IOException ex) {

        Logger.getLogger(SocketThread.class.getName()).log(Level.SEVERE,
        null, ex);
    }
}
}

```

Файл *SaperUI.java*

```

package saperkurs;
import javax.swing.ImageIcon;
import java.awt.Image;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import sweeper.Box;
import sweeper.Coord;
import sweeper.Game;
import sweeper.Ranges;

import sweeper.GameState;

public class SaperUI extends javax.swing.JFrame {

    private static Game game;
    private int bombs = 10;
    private int cols = 9;
    private int rows = 9;
    private final int IMAGE_SIZE = 50;
    private Client client;

    public SaperUI() {

        InitDialog();

        jDialog1.pack();
    }
}

```

```

jDialog1.setLocationRelativeTo(null);
jDialog1.setVisible(true);

jDialog1.addWindowListener(new WindowAdapter()
{
    @Override
    public void windowClosing(WindowEvent e)
    {
        //e.getWindow().dispose();
        System.exit(0);
    }
});
}

/**
 * This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this
 * method is always
 * regenerated by the Form Editor.
 */

public void InitDialog(){

    jDialog1 = new
javax.swing.JDialog((java.awt.Dialog)null);
    jPanel3 = new javax.swing.JPanel();
    jTextField1 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel(){
        @Override
        protected void paintComponent(Graphics g){

super.paintComponent(g);
for(Coord coord : Ranges.GetAllCoords()){

g.drawImage((Image)game.GetBox(coord).image,
            coord.x * IMAGE_SIZE, coord.y *
IMAGE_SIZE, this);

        }
    }
};
jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();

```

```

jTextField1.setHorizontalAlignment(javax.swing.JTextField.CENTER
);
    jTextField1.setText("localhost");

    jLabel3.setText(":");

jTextField2.setHorizontalAlignment(javax.swing.JTextField.CENTER
);
    jTextField2.setText("1777");
    jTextField2.addActionListener(new
java.awt.event.ActionListener() {
        public void
actionPerformed(java.awt.event.ActionEvent evt) {
            jTextField2ActionPerformed(evt);
        }
    });

    jLabel4.setText("Адрес");

    jLabel5.setText("Порт");

javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
    jPanel3.setLayout(jPanel3Layout);
    jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
    .addGap(10, 10, 10)
    .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELA
TED)
    .addComponent(jLabel3)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELA
TED)
    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(jPanel3Layout.createSequentialGroup()
    .addGap(44, 44, 44)
    .addComponent(jLabel4)
    .addGap(83, 83, 83)

```

```

.addComponent(jLabel5)))
.addGap(30, 30, 30))
);
jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
.addGroup(jPanel3Layout.createSequentialGroup())
.addGap(21, 21, 21)
.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.BASELINE)
.addComponent(jLabel4)
.addComponent(jLabel5))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELA
TED)
.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.BASELINE)
.addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabel3)
.addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addContainerGap(34, Short.MAX_VALUE))
);

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER
);
jLabel2.setText("Подключение к серверу");
jLabel2.setToolTipText("");

javax.swing.GroupLayout jPanel4Layout = new
javax.swing.GroupLayout(jPanel4);
jPanel4.setLayout(jPanel4Layout);
jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup())
.addContainerGap(31, Short.MAX_VALUE)
.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
196, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(31, 31, 31))
);
jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)

```

```

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup())
.addGap(0, 14, Short.MAX_VALUE)
.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
36, javax.swing.GroupLayout.PREFERRED_SIZE))
);

jButton1.setText("Подключиться");
jButton1.addActionListener(new
java.awt.event.ActionListener() {
    public void
actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jDialog1Layout = new
javax.swing.GroupLayout(jDialog1.getContentPane());
jDialog1.getContentPane().setLayout(jDialog1Layout);
jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
.addGroup(jDialog1Layout.createSequentialGroup())
.addContainerGap(46, Short.MAX_VALUE)
.addGroup(jDialog1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.TRAILING, false)
.addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addGap(45, 45, 45))
.addGroup(jDialog1Layout.createSequentialGroup())
.addGap(116, 116, 116)
.addComponent(jButton1)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
);
jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
.addGroup(jDialog1Layout.createSequentialGroup())
.addContainerGap(46, Short.MAX_VALUE)
.addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)
.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(30, 30, 30)
.addComponent(jButton1)

```

```

.addGap(31, 31, 31))
    );

}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated
Code">
private void initComponents() {

    jDialog1 = new javax.swing.JDialog();
    jPanel3 = new javax.swing.JPanel();
    jTextField1 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel() {
        @Override
        protected void paintComponent(Graphics g) {

super.paintComponent(g);
for(Coord coord : Ranges.GetAllCoords()) {

g.drawImage((Image) game.GetBox(coord).image,
            coord.x * IMAGE_SIZE, coord.y *
IMAGE_SIZE, this);

        }
    }
};
jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();

jTextField1.setHorizontalAlignment(javax.swing.JTextField.CENTER
);
    jTextField1.setText("localhost");

    jLabel3.setText(":");

jTextField2.setHorizontalAlignment(javax.swing.JTextField.CENTER
);
    jTextField2.setText("1777");
    jTextField2.addActionListener(new
java.awt.event.ActionListener() {
        public void
actionPerformed(java.awt.event.ActionEvent evt) {

```

```

        jTextField2ActionPerformed(evt);
    }
});

jLabel4.setText("Адрес");

jLabel5.setText("Порт");

javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jTextField1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 131,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField2,
                javax.swing.GroupLayout.PREFERRED_SIZE, 60,
                javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(44, 44, 44)
                .addComponent(jLabel4)
                .addGap(83, 83, 83)
                .addComponent(jLabel5)))
        .addGap(30, 30, 30)
    );
jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel4)
            .addComponent(jLabel5))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    )
);

```

```

        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.
        out.Alignment.BASELINE)
        .addComponent(jTextField1,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel3)
        .addComponent(jTextField2,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap(34, Short.MAX_VALUE)
        );

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER
);
        jLabel2.setText("Подключение к серверу");
jLabel2.setToolTipText("");

javax.swing.GroupLayout jPanel4Layout = new
javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
        .addContainerGap(31, Short.MAX_VALUE)
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
196, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(31, 31, 31))
        );
        jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
        .addGap(0, 14, Short.MAX_VALUE)
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
36, javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        jButton1.setText("Подключиться");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
}
});

```

```

javax.swing.GroupLayout jDialog1Layout = new
javax.swing.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
    .addGroup(jDialog1Layout.createSequentialGroup())
    .addContainerGap(46, Short.MAX_VALUE)
    .addGroup(jDialog1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.TRAILING, false)
    .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGap(45, 45, 45))
    .addGroup(jDialog1Layout.createSequentialGroup())
    .addGap(116, 116, 116)
    .addComponent(jButton1)
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
    );
    jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
    .addGroup(jDialog1Layout.createSequentialGroup())
    .addContainerGap(46, Short.MAX_VALUE)
    .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(18, 18, 18)
    .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(30, 30, 30)
    .addComponent(jButton1)
    .addGap(31, 31, 31)
    );

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
setMaximumSize(new java.awt.Dimension(800, 910));
setMinimumSize(new java.awt.Dimension(450, 560));
setPreferredSize(new java.awt.Dimension((Ranges.GetSize().x *
IMAGE_SIZE) + 20,
    (Ranges.GetSize().y * IMAGE_SIZE) + 110));
    setResizable(false);
setSize(new java.awt.Dimension(0, 0));

jPanel1.addMouseListener(new MouseAdapter(){

```

```

@Override
public void mousePressed(MouseEvent event) {
    int x = event.getX() / IMAGE_SIZE;
    int y = event.getY() / IMAGE_SIZE;
    Coord coord = new Coord(x, y);
    if(game.GetState() == GameState.PLAYED){
        if (event.getButton() == MouseEvent.BUTTON1){

            if(client.IsConnected())
client.PressButton("Left", coord);
game.PressLeftButton (coord);
        }

        if (event.getButton() == MouseEvent.BUTTON3){
            if(client.IsConnected())
client.PressButton("Right", coord);
game.PressRightButton (coord);
        }

        jPanell.repaint();

        if(game.GetState() == GameState.BOMBED){
            if(client.IsConnected()){
client.NewGameRequest();

jLabell.setText(GetMessage(client.IsConnected(), game));
game.SetState(GameState.READY);
                return;
            }
        }

        if(game.GetState() == GameState.WINNER){
            if(client.IsConnected()){
client.NewGameRequest();

jLabell.setText(GetMessage(client.IsConnected(), game));
game.SetState(GameState.READY);
                return;
            }
        }
    }

    jLabell.setText(GetMessage(client.IsConnected(),
game));

    if (event.getButton() == MouseEvent.BUTTON2){
        if(game.GetState() == GameState.READY){
            if(client.IsConnected()){
                game = client.GetGame();
game.StartCoop();

                jPanell.repaint();

jLabell.setText(GetMessage(client.IsConnected(), game));
            }
        }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ИГУ 1.090301.45.001 ПЗ

Лист

98

```

        }

    }

}

});
jPanel1.setMaximumSize(new java.awt.Dimension(800, 800));
jPanel1.setMinimumSize(new java.awt.Dimension(450, 450));
jPanel1.setName("panel"); // NOI18N
jPanel1.setPreferredSize(new java.awt.Dimension(
    Ranges.GetSize().x * IMAGE_SIZE,
    Ranges.GetSize().y * IMAGE_SIZE));

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 800, Short.MAX_VALUE)
    );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 800, Short.MAX_VALUE)
    );

jLabel1.setFont(new java.awt.Font("Trebuchet MS", 0, 18));
// NOI18N
jLabel1.setText("Hello World Saper");

javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );
jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()

```

```

.addContainerGap()
.addComponent(jLabel1)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addContainerGap()
.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
800, Short.MAX_VALUE)
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
800, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(32, Short.MAX_VALUE))
);

pack();
setLocationRelativeTo(null);
} // </editor-fold>

private void
jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
client = new Client(jTextField1.getText(),
Integer.parseInt(jTextField2.getText()));
client.ClientStart();

if(client.IsConnected())
{
jDialog1.dispose();
}
}

```

```

        game = client.GetGame();

game.StartCoop();

client.start();

SetImages();
initComponents();

        setVisible(true);
client.SetPanel(jPanell1);
client.SetLabel(jLabel1);

this.addWindowListener(new WindowAdapter()
{
    @Override
    public void windowClosing(WindowEvent e)
    {
client.StopConnection();

        // e.getWindow().dispose();
        System.exit(0);
    }
});

        setVisible(true);
    }
else{
        jLabel2.setText("Ошибка подключения");
    }

}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```

```

        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex)
{
java.util.logging.Logger.getLogger(SaperUI.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        }
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new SaperUI();
    }
});

}

public static String GetMessage(boolean isConnected, Game
game){

    if(isConnected){
        switch(game.GetState()){

            casePLAYED :return "Сапёр ошибается лишь
однажды";
            case BOMBED : return "Сапёр ошибся...";
            case WINNER : return "Молодец, сапёр!";
            caseREADY :return "Новая игра готова. Нажмите
СКМ.";

            default : return "Hello World";

        }
    }
    else
return "Потеряно соединение с сервером";

}

private Image GetImage (String name)
{
    String filename = "/res/img/" + name + ".png";
    // System.out.println(filename);
    ImageIcon icon = new
ImageIcon(getClass().getResource(filename));
    return icon.getImage();
}

```

```

    }

    public javax.swing.JPanel GetPanel() {

        return jPanel1;
    }

    private void SetImages() {

for(Box box : Box.values())
box.image = GetImage(box.name().toLowerCase());
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
}

```

Файл *Client.java*

```

package superkurs;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import sweeper.Coord;
import sweeper.Game;
import sweeper.GameState;

public class Client extends Thread {
    private ObjectOutputStream oos;
    private ObjectInputStream ois;
    private Game game;
    private JLabel label;

```

```

private JPanel panel;
private Socket socket;
private boolean status;

private int cols;
private int rows;

private String sIp;
private int port;

public Client(String ip, int port) {

    sIp = ip;
    this.port = port;

}

public void SetGame(Game game) {
    this.game = game;
}

public Game GetGame() {
    return game;
}

public void SetLabel(JLabel label) {
    this.label = label;
}

public void SetPanel(JPanel panel) {
    this.panel = panel;
}

public void run() {
    try {
        WaitingForAction();
    } catch (Exception ex) {

        Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
        ex);
    }
}

public boolean IsConnected() {
    if((socket != null) && (!socket.isClosed()))
        return socket.isConnected();
    else
        return false;
}

public void ClientStart() {

```

```

        try {

            // socket = new Socket(sIp, port);

            socket = new Socket();

            socket.connect(new InetSocketAddress(sIp, port), 666);

            if(IsConnected()){

                oos = new
                ObjectOutputStream(socket.getOutputStream());
                ois = new
                ObjectInputStream(socket.getInputStream());
                try {

                    System.out.println("Client is started");

                    cols = ois.readInt();
                    rows = ois.readInt();

                    game = new Game(cols, rows);

                    game = (Game)ois.readObject();
                    System.out.println(game.toString());

                    status = true;

                } catch (ClassNotFoundException ex) {

                    Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
                    ex);

                }

            } catch (IOException ex) {

                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
                ex);

            }

        }

        public void NewGameRequest(){

            if(IsConnected()){
                try {
                    oos.writeObject("NgPls");
                } catch (IOException ex) {

                    Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
                    ex);

                }

            }

        }

```

```

        }
    }

}

private void WaitingForAction() throws Exception {

    Coord coord = new Coord(0, 0);
    String str = "123";

    while ((str = (String)ois.readObject()) != null) {

        switch(str){

            case "Left":
                coord = (Coord) ois.readObject();
                System.out.println("Получен ответ: " + str +
" на " + coord.x + " " + coord.y);
                game.PressLeftButton(coord);
                RepaintPanel();

                break;

            case "Right":
                coord = (Coord) ois.readObject();
                System.out.println("Получен ответ: " + str +
" на " + coord.x + " " + coord.y);
                game.PressRightButton(coord);
                RepaintPanel();

                break;

            case "Ng Guys":
                System.out.println("Получен ответ: " + str);
                game.SetState(GameState.READY);
                game = (Game)ois.readObject();
                System.out.println(game.toString());

                break;

            case "Exit":

                if(status)
                    StopConnection();
                return;

            default:

```

```

        System.out.println(" Получен необработанный
ответ: " + str);

        break;

    }

}

void RepaintPanel(){

    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
label.setText(SaperUI.GetMessage(IsConnected(), game));
panel.repaint();
        }

    });

}

void PressButton(String string, Coord coord) {
if(IsConnected()){
    try {
oos.writeObject(string);
oos.writeObject(coord);
    } catch (IOException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
ex);

    }

}

void StopConnection() {

if(IsConnected()){

    try {
        status = false;
oos.writeObject("Exit");
        try {
            Thread.sleep(120);
        } catch (InterruptedException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
ex);

        }

oos.close();
ois.close();
socket.close();
    } catch (IOException ex) {

```

```

        Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
        ex);
    }
}
}
}

```

Файл *Bomb.java*

```

package sweeper;

import java.io.Serializable;

public class Bomb implements Serializable {

    private Matrix bombMap;

    private int totalBombs;

    Bomb(int totalBombs){

        this.totalBombs = totalBombs;
        FixBombsCount();

    }

    void Start(){

        bombMap = new Matrix(Box.ZERO);
        for (int i = 0; i < totalBombs; i++) {

            PlaceBomb();

        }

    }

    public int GetTotalBombs() {
        return totalBombs;
    }

    Box Get (Coord coord){

        return bombMap.Get(coord);

    }

    private void FixBombsCount () {

```

```

        int maxBombs = Ranges.GetSize().x * Ranges.GetSize().y /
2;
if(totalBombs > maxBombs)
    totalBombs = maxBombs;

    }

    private void PlaceBomb(){

        while(true){

            Coord rCoord = Ranges.GetRandomCoord();
if(bombMap.Get(rCoord) == Box.BOMB)
            continue;

            bombMap.Set(rCoord, Box.BOMB);
            IncNumbersAroundBomb(rCoord);
            break;

        }

    }

    private void IncNumbersAroundBomb(Coord coord){

        /*for (Coord around : Ranges.GetCoordsAround(coord))
if(Box.BOMB != bombMap.Get(around))
            bombMap.Set(around,
bombMap.Get(around).GetNextNumberBox()); */

        Ranges.GetCoordsAround(coord).stream().filter((around) -
> (Box.BOMB != bombMap.Get(around))).forEachOrdered((around) ->
{
            bombMap.Set(around,
bombMap.Get(around).GetNextNumberBox());
        });
    }
}

```

Файл *Box.java*

```

package sweeper;
import java.io.Serializable;

public enum Box implements Serializable {

    ZERO,
    NUM1,
    NUM2,
    NUM3,
    NUM4,
    NUM5,
    NUM6,
    NUM7,

```

```

    NUM8,
    BOMB,
    OPENED,
    CLOSED,
    FLAGED,
    BOMBED,
    NOBOMB;

    public Object image;

    Box GetNextNumberBox () {

        return Box.values()[this.ordinal() + 1];
    }

    int GetNumber() {
        return this.ordinal();
    }
}

```

Файл *Coord.java*

```

package sweeper;
import java.io.Serializable;

/**
 *
 * @author User
 */
public class Coord implements Serializable {
    public int x;
    public int y;

    public Coord(int x, int y){
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object obj){

        if (obj instanceof Coord){

            Coord to = (Coord)obj;
            return to.x == x &&to.y == y;
        }

        return super.equals(obj);
    }
}

```

```
}
```

Файл *Flag.java*

```
package sweeper;

import java.io.Serializable;

class Flag implements Serializable {

    private Matrix flagMap;
    private int countOfClosedBoxes;

    void Start(){

        flagMap = new Matrix(Box.CLOSED);
        countOfClosedBoxes = Ranges.GetSize().x *
Ranges.GetSize().y;

    }

    Box Get (Coord coord){

        return flagMap.Get(coord);

    }

    void SetOpenedToBox(Coord coord) {

        flagMap.Set(coord, Box.OPENED);
        countOfClosedBoxes--;

    }

    void ToggleFlagedToBox (Coord coord){

switch(flagMap.Get(coord)){

        case FLAGED : SetClosedToBox(coord); break;
        case CLOSED : SetFlagedToBox(coord); break;

    }

    }

    void SetFlagedToBox(Coord coord) {

        flagMap.Set(coord, Box.FLAGED);

    }

    void SetClosedToBox(Coord coord) {

        flagMap.Set(coord, Box.CLOSED);

    }

}
```

Изм.	Лист	№ докум.	Подпись	Дата

ИГУ 1.090301.45.001 ПЗ

Лист

111

```

    }

    int GetCountOfClosedBoxes() {
        return countOfClosedBoxes;
    }

    void SetBombedToBox(Coord coord) {
        flagMap.Set(coord, Box.BOMBED);
    }

    void SetOpenedToClosedBombBox(Coord coord) {
        if(flagMap.Get(coord) == Box.CLOSED)
            flagMap.Set(coord, Box.OPENED);
    }

    void SetNoBombToFlagedSafeBox(Coord coord) {
        if(flagMap.Get(coord) == Box.FLAGED)
            flagMap.Set(coord, Box.NOBOMB);
    }

    int GetCountOfFlagedBoxesAround(Coord coord) {

        int count = 0;
        for(Coord around : Ranges.GetCoordsAround(coord))
            if(flagMap.Get(around) == Box.FLAGED)
                count++;

        return count;
    }
}

```

Файл *Game.java*

```

package sweeper;

import java.io.Serializable;
public class Game implements Serializable {

    private Bomb bomb;

    private Flag flag;

    private GameState gState;

    public Game(int cols, int rows){

        Ranges.SetSize(new Coord(cols, rows));

    }

    public Game(int cols, int rows, int bombs){

```

```

        Ranges.SetSize(new Coord(cols, rows));
        bomb = new Bomb(bombs);
        flag = new Flag();
    }

    public Game() {

    }

    public void StartSolo(){

        bomb.Start();
        flag.Start();
        gState = GameState.PLAYED;

    }

    public void StartCoop(){

        gState = GameState.PLAYED;

    }

    public Bomb GetBomb() {
        return bomb;
    }

    public Flag GetFlag() {
        return flag;
    }

    public void SetState(GameState gState) {
this.gState = gState;
    }

    public void SetBomb(Bomb bomb) {
this.bomb = bomb;
    }

    public void SetFlag(Flag flag) {
this.flag = flag;
    }

    public GameState GetState() {
        return gState;
    }

    public Box GetBox(Coord coord){

        if(flag.Get(coord) == Box.OPENED)
            return bomb.Get(coord);
        else

```

```

        return flag.Get(coord);
    }

    public void PressLeftButton(Coord coord){
        if(GameOver())
            return;
        OpenBox(coord);
        CheckWinner();
        /// flag.SetOpenedToBox(coord);
    }

    public void PressRightButton(Coord coord){
        if(GameOver())
            return;
        flag.ToggleFlagedToBox(coord);
    }

    public void CheckWinner(){
        if (gState == GameState.PLAYED)
            if (flag.GetCountOfClosedBoxes() ==
bomb.GetTotalBombs())
                gState = GameState.WINNER;
    }

    public void OpenBox(Coord coord){
        switch(flag.Get(coord)){

            case OPENED:
SetOpenedToClosedBoxesAroundNumber(coord); return;

            case FLAGED: return;

            case CLOSED:
                switch(bomb.Get(coord)){

                    case ZERO : OpenBoxesAround(coord); return;
                    case BOMB : OpenBombs(coord); return;
                    default    : flag.SetOpenedToBox(coord); return;

                }
            }
    }

    private void OpenBombs(Coord coordBombed) {
        gState = GameState.BOMBED;
        flag.SetBombedToBox(coordBombed);

        for(Coord coord : Ranges.GetAllCoords())

```

```

        if(bomb.Get(coord) == Box.BOMB)
flag.SetOpenedToClosedBombBox(coord);
        else
flag.SetNoBombToFlagedSafeBox(coord);

    }

    private void OpenBoxesAround(Coord coord) {
flag.SetOpenedToBox(coord);
for(Coord around : Ranges.GetCoordsAround(coord))
    OpenBox(around);
    }

    private boolean GameOver() {
if(gState == GameState.PLAYED)
    return false;
    System.out.println("Я подорвался");

return true;

    }

    void SetOpenedToClosedBoxesAroundNumber(Coord coord) {

        if(bomb.Get(coord) != Box.BOMB)
            if(flag.GetCountOfFlagedBoxesAround(coord) ==
bomb.Get(coord).GetNumber())
for(Coord around : Ranges.GetCoordsAround(coord))
            if (flag.Get(around) == Box.CLOSED)
                OpenBox(around);

    }

}

```

Файл *GameState.java*

```

package sweeper;

import java.io.Serializable;

public enum GameState implements Serializable {

    PLAYED,
    BOMBED,
    WINNER,
    READY

}

```

Файл *Matrix.java*

```

package sweeper;

```

```

import java.io.Serializable;

class Matrix implements Serializable {

    private Box[][] matrix;

    Matrix(Box defaultBox) {
        matrix = new
Box[Ranges.GetSize().x][Ranges.GetSize().y];

        for(Coord coord : Ranges.GetAllCoords())
            matrix [coord.x] [coord.y] = defaultBox;

    }

    Box Get(Coord coord)
    {
        if(Ranges.InRange(coord))
            return matrix[coord.x][coord.y];
        else
            return null;
    }

    void Set(Coord coord, Box box){

        if(Ranges.InRange(coord))
            matrix[coord.x][coord.y] = box;
        }

    }

```

Файл *Ranges.java*

```

package sweeper;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Random;

public class Ranges implements Serializable {
    private static Coord size;
    private static ArrayList<Coord> allCoords;
    private static Random random = new Random();

    static void SetSize(Coord oSize){

        size = oSize;
        allCoords = new ArrayList<Coord>();
        for(int y = 0; y < size.y; y++)
            for (int x = 0; x < size.x; x++) {
                allCoords.add(new Coord(x, y));
            }
    }

```

```

    }

    public static Coord GetSize() {
        return size;
    }

    public static ArrayList<Coord>GetAllCoords(){
        return allCoords;
    }
    static boolean InRange(Coord coord){

        return coord.x >= 0 && coord.x < size.x &&
coord.y>= 0 && coord.y < size.y;
    }

    static Coord GetRandomCoord(){

        return new Coord(random.nextInt(size.x),
random.nextInt(size.y));

    }

    static ArrayList<Coord>GetCoordsAround(Coord coord){ // для
перебора единиц

        Coord around;
        ArrayList<Coord> coordsList = new ArrayList<Coord>();

        for (int x = coord.x - 1; x <= coord.x + 1; x++)
            for (int y = coord.y - 1; y <= coord.y + 1; y++) {
if(InRange(around = new Coord(x, y)))
                if(!around.equals(coord))
                    coordsList.add(around);
            }

        return coordsList;
    }
}

```

Приложение 3

(Обязательное)

Кроссплатформенное клиент-серверное приложение

Презентация

					ПГУ 1.090301.45.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		118

КРОССПЛАТФОРМЕННОЕ КЛИЕНТ- СЕРВЕРНОЕ ПРИЛОЖЕНИЕ

Автор работы:
студент группы 18ВВ1 Щербакова К.А.

Научный руководитель:
к.т.н., доцент каф. ВТ Юрова О. В.

Кроссплатформенная разработка

Кроссплатформенные приложения – приложения, которые разрабатываются и затем мгновенно могут работать на многих платформах. Суть разработки в том, что исходный код приложения переводится в нативный, то есть очевидный определенному устройству. В результате программа сможет взаимодействовать с установленной на нём операционной системой.

Плюсы	Минусы
Единая база кода для всех платформ	Меньшая гибкость, чем у нативных приложений
Сокращение времени и стоимости разработки	Более низкая производительность и надежность
Покрытие более широкой аудитории пользователей	Возможное несоответствие в дизайне на двух платформах
Допустимость одинакового интерфейса и UX	

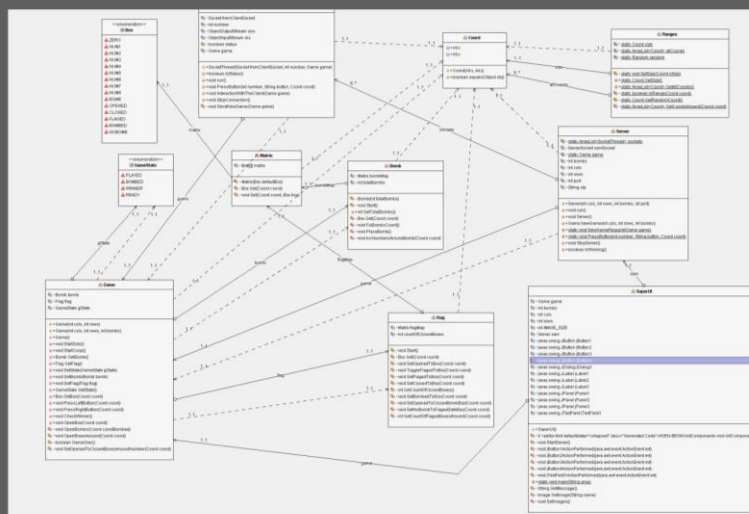
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.45.001 ПЗ

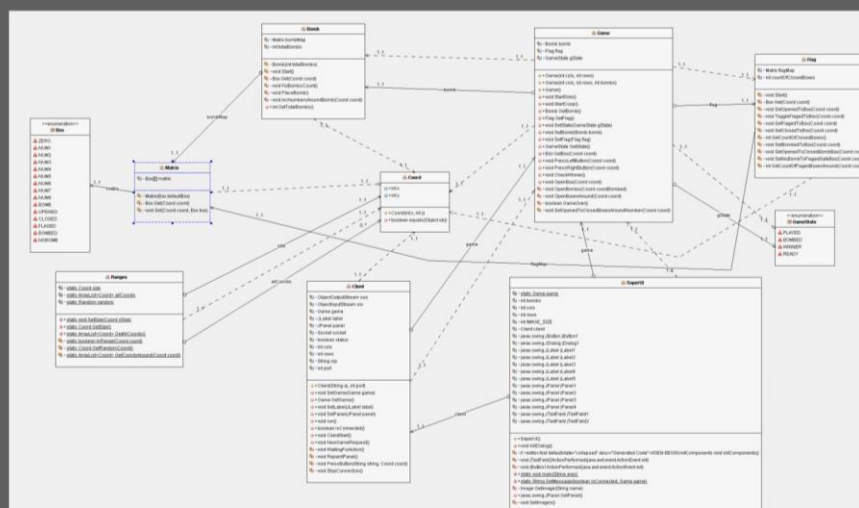
Лист

119

Структура серверного приложения



Структура клиентского приложения



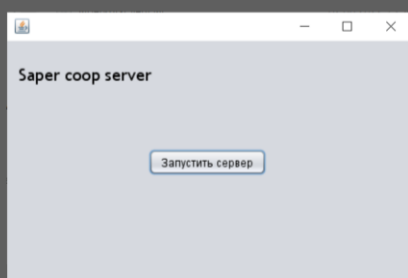
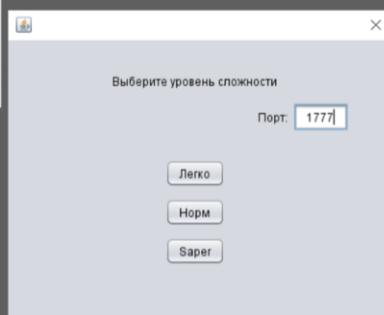
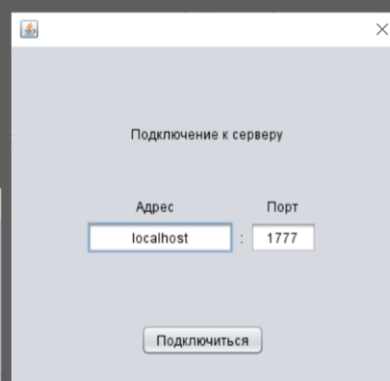
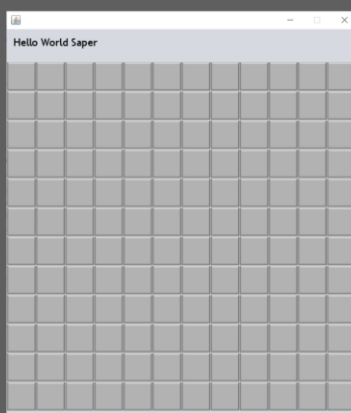
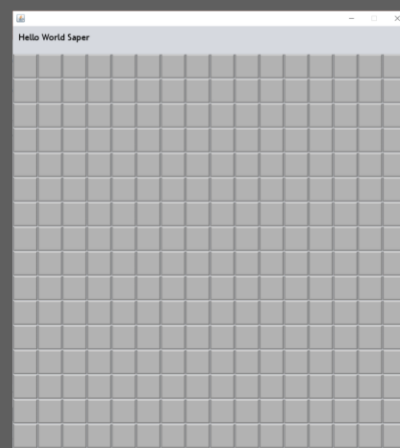
UML – диаграмма последовательностей



UML-диаграмма прецедентов



Результат работы

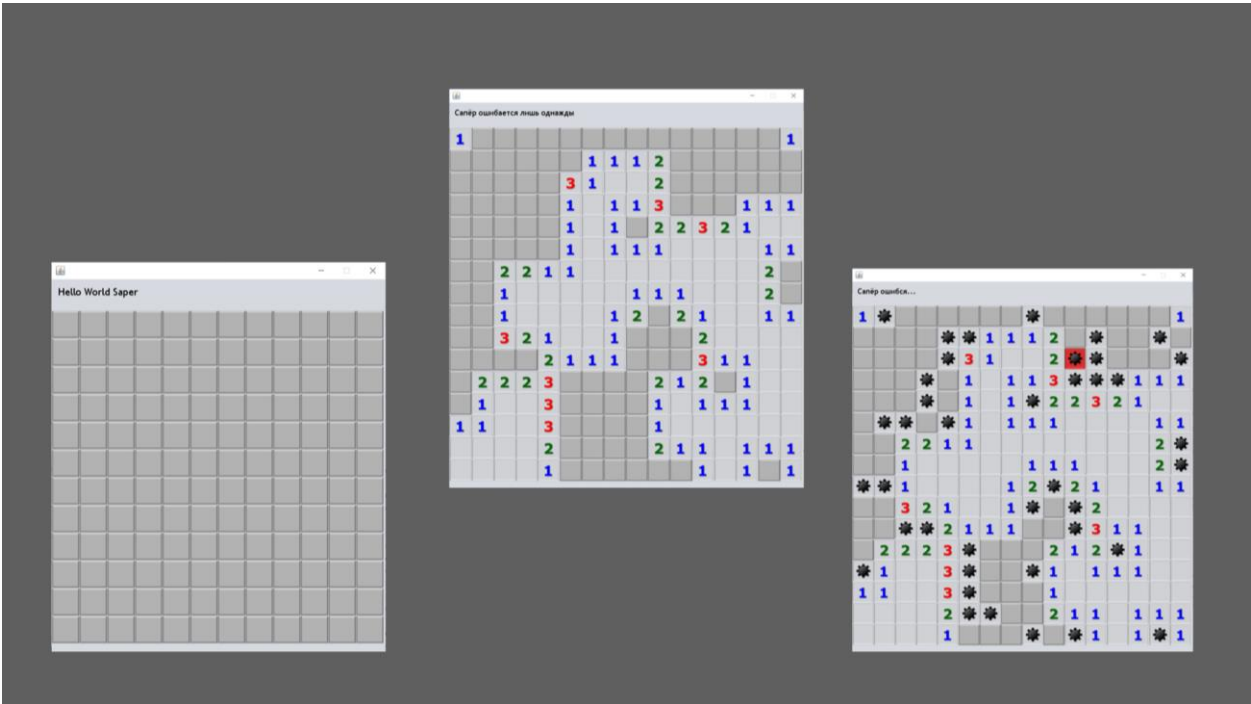


Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.45.001 ПЗ

Лист

122



Спасибо за внимание!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.45.001 ПЗ

Лист

123