Федеральное агентство связи Федеральное государственное бюджетное образовательное учреждения высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Курсовая работа по курсу «Технология разработки программного обеспечения» для бакалавров. Заочная форма обучения.

Выполнил:

Группа: ЗП-92

Проверил:_____

Новосибирск, 2020

# Введение

В мире разработки при создании своего проекта работа в команде будет неизбежной. Может казаться, что в команде работать проще, ведь больше рук, а значит можно гораздо быстрее справиться с работой. Но не все так просто.

Крупнейшим веб-сервисом для хостинга IT-проектов и их совместной разработки является GitHub**.** Создатели сайта называют GitHub «социальной сетью для разработчиков».

Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.

С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок.

Но что же насчет командной разработки? Рассмотрим этот вопрос поподробнее.

Небольшая юмористическая задачка:

Дано:

- N разработчиков;

- Рабочие места;

- Техническое задание (ТЗ);

- Интернет.

Вопрос:

- Как выполнить проект, не привлекая внимание санитаров?

Ответ:

Слаженная команда и упорный труд.

Что же представляет собой команда?

Команда – это небольшое количество людей:

- с взаимодополняющими умениями (кто-то программирует, кто-то занимается дизайном и т.д.)

- стремящимся к общим целям (им всем нужно выполнить задачу заказчика, чтобы получить полное самоудовлетворение)

- разделяющих ответственность за достижение цели проекта (если вдруг у кого-то из участников команды возникает трудность с его личным ТЗ, то его напарники всегда придут ему на помощь (этим не стоит злоупотреблять, иначе можно оказаться ненужным для команды)).

Целью данной работы является ознакомиться, с помощью каких инструментов ведется разработка, и что происходит внутри команды. Основной задачей данного курсового проекта является программная реализация логической компьютерной игры «сапер».

Необходимо предусмотреть различные варианты развития событий, чтобы программа не выдавала ошибок и работала правильно.

Данная программа может использоваться любой возрастной категорией людей для развития собственного мышления и логики, а также для отдыха и развлечения.

Техническое задание

Сапер - это обманчиво простая игра для развития памяти и логики, которая стала одной из самых популярных игр ОС Windows.

Игровое поле разделено на смежные ячейки (квадраты), некоторые из которых «заминированы». Целью игры является открытие всех ячеек, не содержащих мины.

Игрок открывает ячейки, стараясь не открыть ячейку с миной. Открыв ячейку с миной, он проигрывает. Если под открытой ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано»; используя эти числа, игрок пытается рассчитать расположение мин. Открыв все «не заминированные» ячейки, игрок выигрывает.

Сапёр имеет интерактивное поле.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- Создавать ячейки игрового поля
- Открывать ячейки игрового поля
- Проверять наличие бомб в ячейках
- Выводить в ячейках информацию о бомбах
- Засекать время до победы
- Определять количество открытых ячеек и каждый раз проверять, не завершилась ли игра
- Сброс текущей игры
- Выбор сложности

Игра должна иметь несколько уровней сложности:

- Легкий
- Средний
- Тяжелый
- Ручная настройка поля

Игра должна иметь дружественный для пользователя интерфейс.

Этапы разработки

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на неё;

5. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

Для законченных проектов будет организован процесс защиты. Каждый член команды представляет письменный отчет о проделанной работе. Отчет состоит из следующих обязательных частей:

1. ТЗ проекта и итоговый план работ на команду

2. Описание командной работы и полученного результата

3. Описание личного вклада в результат работы команды.

## Разработка программы

Исходный код проекта написан на паттерне MVC. MVC расшифровывается как модель-представление-контроллер (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения. Компоненты MVC:

- Модель (model)— этот компонент отвечает за данные, а также определяет структуру приложения.

- Представление (view) — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.

- Контроллер (controller) — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

Проект написан на языке программирования Java. По результатам ежегодного отчёта State of the Octoverse, который выпускает Github, язык программирования Java в 2019 году занимает третье место в списке самых популярных.

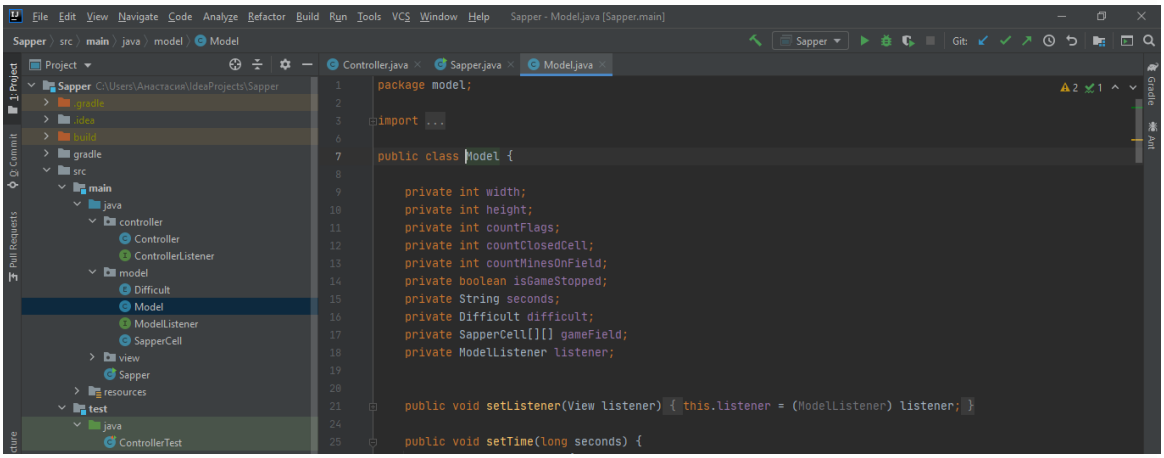
Проект разработан с использованием IntelliJ IDEA. IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования. При работе с IntelliJ IDEA использовалась интеграция с системой контроля версий GitHub.

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:

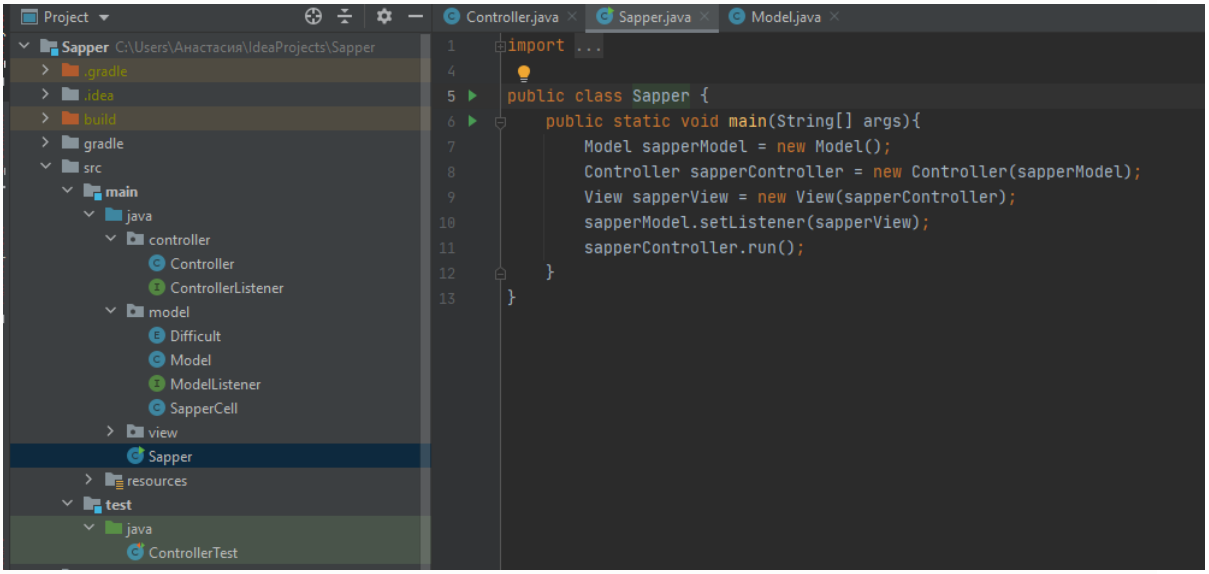
Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает, как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими.

Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

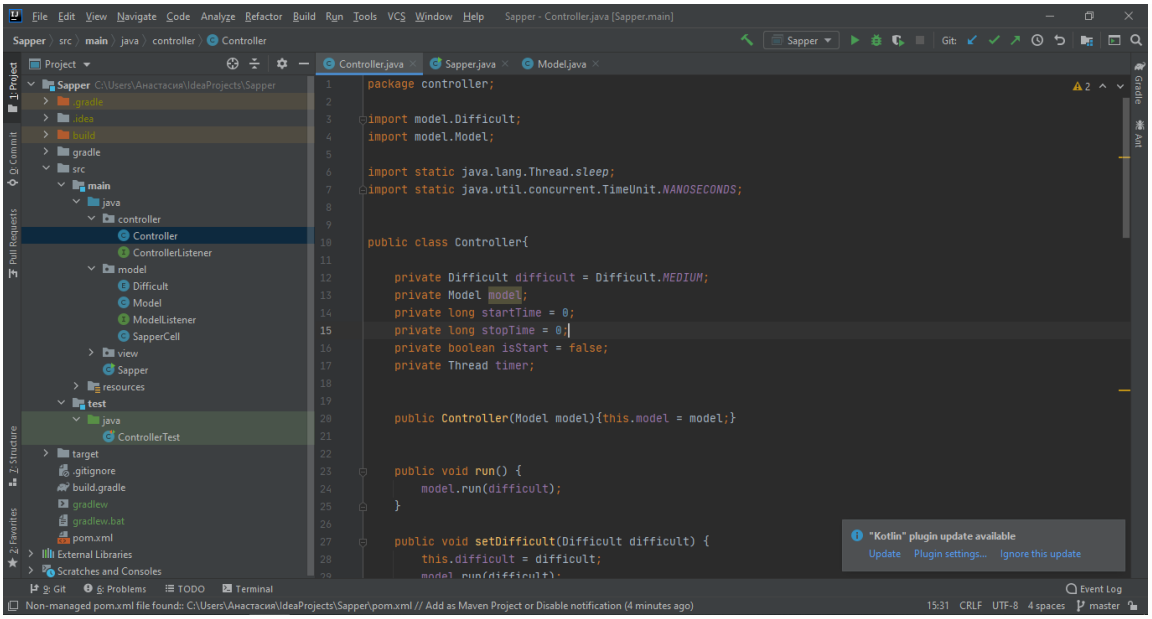
Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».




Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

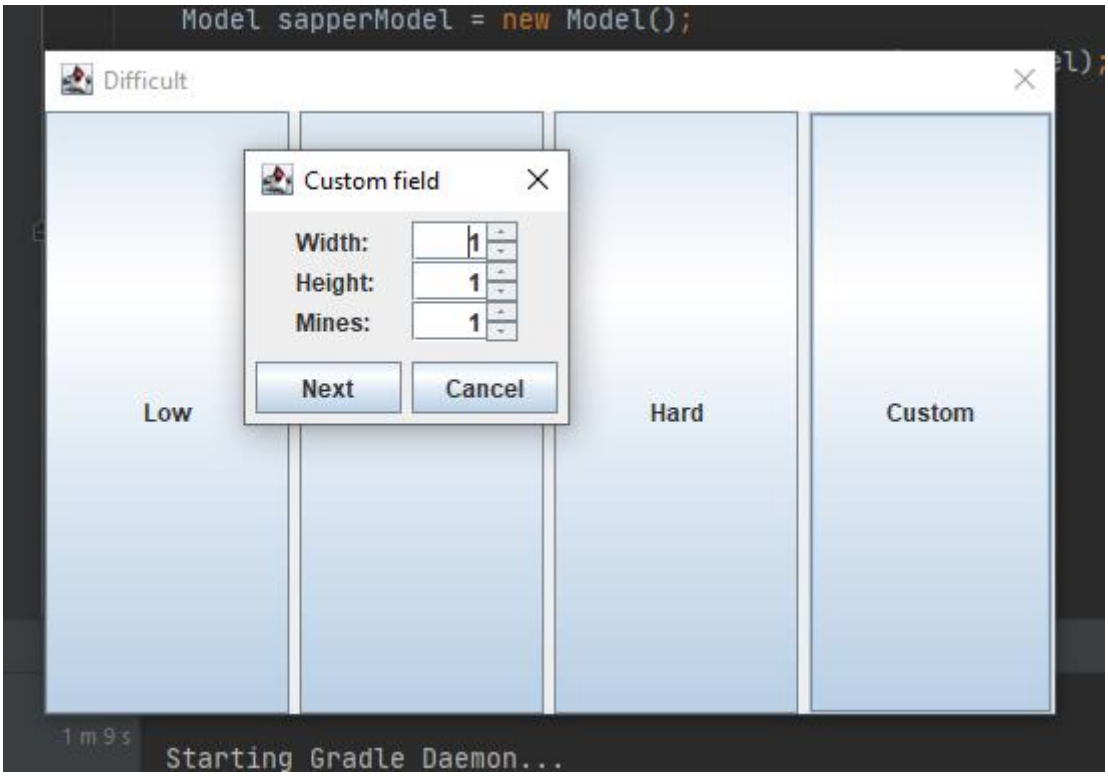



Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

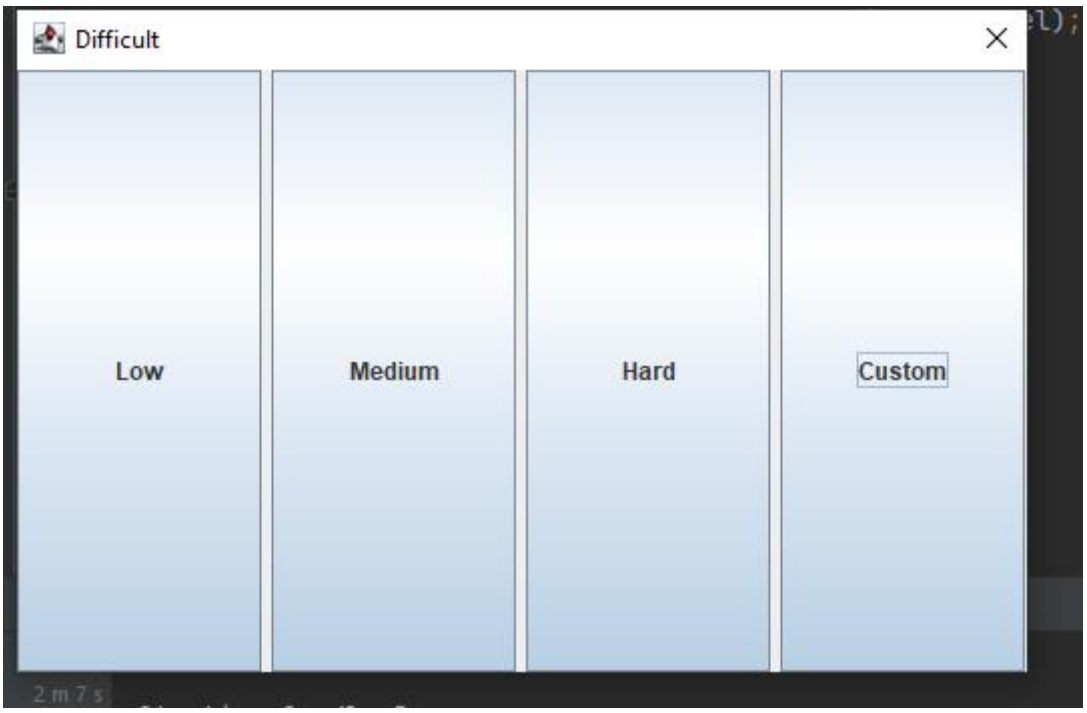



В результате получен детализированный процесс игры «Сапёр» из четырех функций:

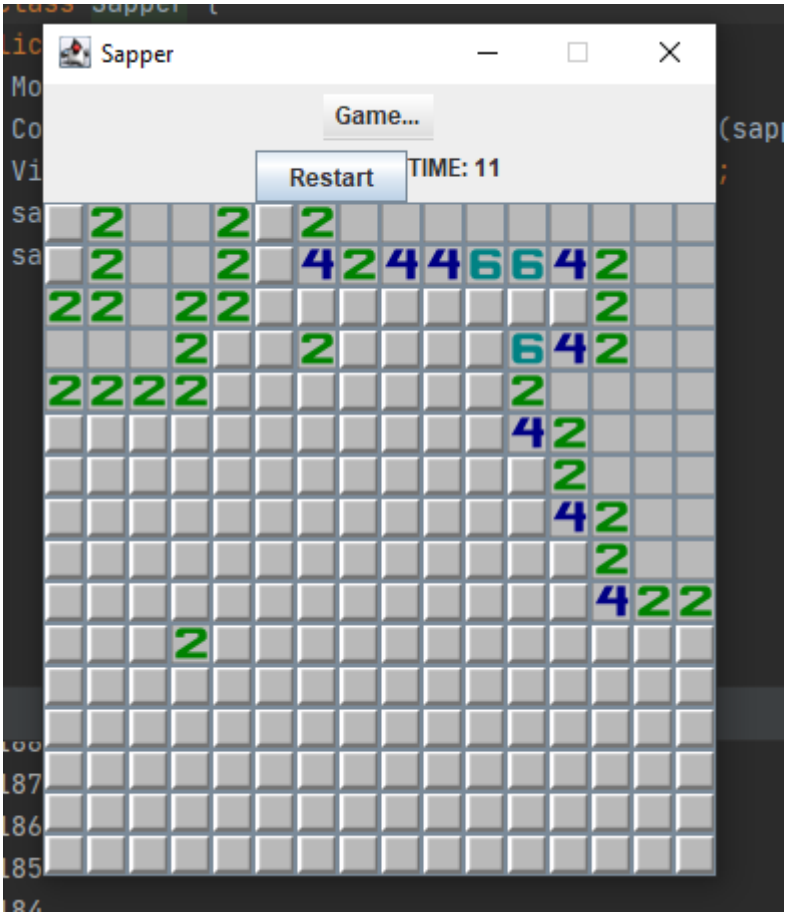
- «Ручная настройка поля», в результате работы этого процесса пользователь самостоятельно настраивает параметры игрового поля;




- «Выбор сложности» - в этом процессе пользователь выбирает одно из 3х заготовленных игровых полей, которые классифицируются по сложности;




- «Выполнение игры» - этот процесс отвечает за работу и выполнение всех игровых функций;




- «Вывод результатов» - после выполнения данного процесса пользователю выводятся результаты игры.


Sapper
Game...
Restart
TIME: 33
BOOM!
You Lose!
OK

## Тестирование

Модульное тестирование, иногда блочное тестирование или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Основное преимущество независимого тестирования маленького участка кода состоит в том, что если тест провалится, ошибку будет легко обнаружить и исправить.

Тест состоит из трёх этапов:

1. Задание тестируемых данных.
2. Использование тестируемого кода (вызов тестируемого метода).
3. Проверка результатов и сверка с ожидаемыми.

Модульными тестами покрыты основные команды контроллера.

Тест для выбора уровня сложности:

```
@Test
public void setDifficultTest(){
    Model model = new Model();
    Controller controller = new Controller(model);
    View view = new View(controller);
    model.setListener(view);
    assert model.getDifficult()== ExpectedDifficult;
```

Тест для левого нажатия по полю:

```
@Test
public void leftClickTest(){
    Model model = new Model();
    Controller controller = new Controller(model);
    View view = new View(controller);
    model.setListener(view);
    int ExpectedLine = 1;
    int ExpectedColumn = 1;
    controller.leftClick(ExpectedLine,ExpectedColumn);
    model.openCell(ExpectedColumn,ExpectedLine);
    assert false;
}
```

Тест для уровня сложности, который устанавливает пользователь:

```
@Test
public void setCustomLevelTest() {

    Model model = new Model();
    Controller controller = new Controller(model);
    View view = new View(controller);
    model.setListener(view);
    int ExpectedLine = 16;
    int ExpectedColumn = 16;
    int ExpectedMine = 40;
    controller.setCustomLevel(ExpectedLine,ExpectedColumn,ExpectedMine);
    assert model.getHeight() == ExpectedLine;
    assert model.getWidth() == ExpectedColumn;
    assert model.getCountMinesOnField() == ExpectedMine;
}
```

Тест начала игры:

```
@Test
public void startTimerTest (){
    Model model = new Model();
    Controller controller = new Controller(model);
    View view = new View(controller);
    model.setListener(view);
    controller.getTime();
    model.getSeconds();
    assert false;

}
```

Тест окончания игры:

```
@Test
public void stopTimerTest (){
    Model model = new Model();
    Controller controller = new Controller(model);
    View view = new View(controller);
    model.setListener(view);
    controller.stopTimer();
    assert false;
}
```

## Индивидуальный вклад в проект

У каждого участника в команде есть своя роль и свои задачи. В рамках данного проекта была выполнена роль тестировщика.

Тестировщик — это специалист, который занимается тестированием программного продукта с целью выявления ошибок в его работе и их последующего исправления.

Следует по возможности избегать тестирования программы ее автором, т.к. кроме уже указанной объективной сложности тестирования для программистов здесь присутствует и тот фактор, что обнаружение недостатков в своей деятельности противоречит человеческой психологии.

Главные обязанности тестировщика:

- Выявление и анализ ошибок и проблем, возникающих у пользователей при работе с программными продуктами.

- Разработка сценариев тестирования.

- Документирование найденных дефектов.

В реализации проекта «Сапёр» целью было максимальное покрытие кода контроллера тестами.