Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |

|  |
| --- |
| Вычислительная техника |
| кафедра |

Коллективный проект

**Клиент-серверное приложение Pac-Man**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проверил |  | \_\_\_\_\_\_\_\_\_\_\_ |  | А.И.Легалов |
|  |  | подпись, дата |  | инициалы, фамилия |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент |  | КИ14-06Б |  |  |  | \_\_\_\_\_\_\_\_\_\_ |  | М.В.Рожков |
|  |  | номер группы |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ14-06Б |  |  |  | \_\_\_\_\_\_\_\_\_\_ |  | Д.Н.Галин |
|  |  | номер группы |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ14-06Б |  |  |  | \_\_\_\_\_\_\_\_\_\_ |  | Д.Е.Костыгин |
|  |  | номер группы |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ14-06Б |  |  |  | \_\_\_\_\_\_\_\_\_\_ |  | А.А.Шатоба |
|  |  | номер группы |  |  |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2018

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 3](#__RefHeading___Toc429_850256707)

[2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#__RefHeading___Toc431_850256707)

[3. ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА 5](#__RefHeading___Toc435_850256707)

[3.1. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ 5](#__RefHeading___Toc437_850256707)

[3.2. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ 8](#__RefHeading___Toc449_850256707)

[4. НАПИСАНИЕ БОТА 9](#__RefHeading___Toc451_850256707)

[5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 11](#__RefHeading___Toc453_850256707)

# ВВЕДЕНИЕ

Основной задачей работы является написание игры, в которой должна быть реализована возможность подключения бота и возможность наблюдения за ним и оценки его эффективности. В качестве игры выбран Pac-Man с несколькими модификациями в правилах.

Цель: создать приложение Pac-Man и API для подключения написанных пользователями ботов, а также провести анализ существующих продуктов.

Задачи:

* выполнить анализ предметной области;
* проанализировать оригинальные правила игры и, при необходимости, внести в них изменения;
* спроектировать архитектуру системы;
* реализовать все компоненты системы;
* составить инструкцию по сборке и развёртыванию системы;
* составить инструкцию по написанию бота.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Pac-Man — [аркадная видеоигра](https://ru.wikipedia.org/wiki/Аркадный_автомат). Задача игрока — управляя Пакманом, съесть все точки в лабиринте, избегая встречи с привидениями, которые гоняются за ним. Игровой процесс классического Pac-Man продемонстрирован на рисунке 1.

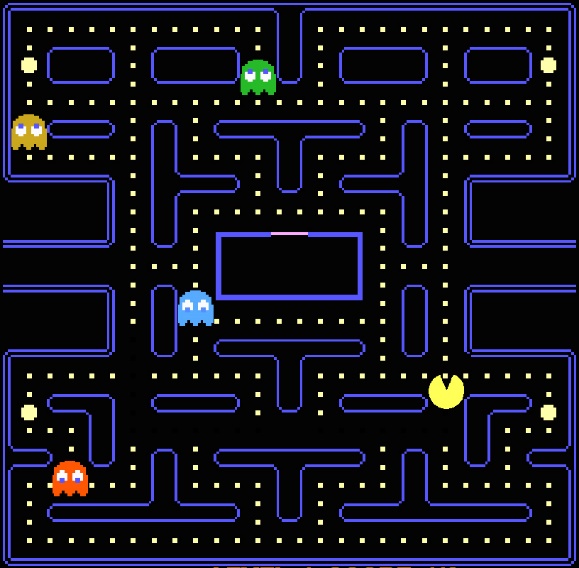


Рисунок 1 – классический Pac-Man

В качестве модификации правил было решено, что при столкновении с призраком игрок не умирает, а теряет очки; призрак уходит в центр поля, а игроку в счетчик начисляется 1 смерть, однако игра для него продолжается. Так же было решено, что в игре должен быть реализован режим игры на двоих. В случае игры на двоих победителем считается тот, кто набирает больше количество очков.

Специфика предлагаемой реализации заключается в том, что к программе можно подключить ботов; кроме того, модификация правил позволяет разрабатывать ботов более гибких, чем в оригинальной версии игры. Игровой процесс модифицированного Pac-Man показан на рисунке 2.

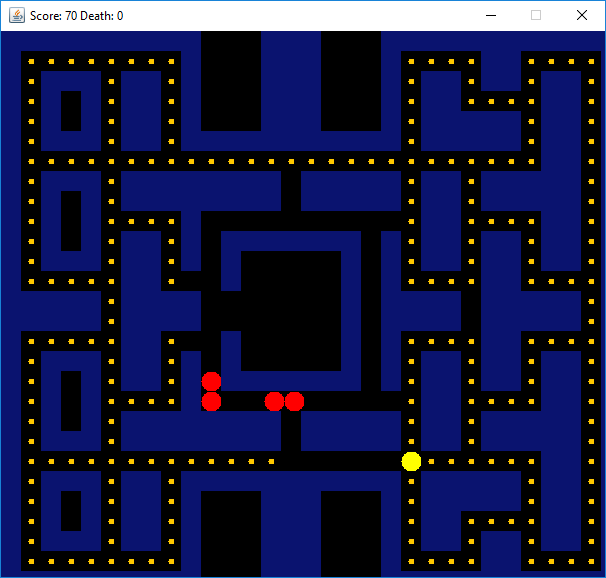


Рисунок 2 – игровой процесс данного проекта.

# ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА

## ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

На сервере постоянно выполняется главный процесс в обязанности которого входит:

* создание новых подключений;
* создание объекта игрока;
* приём игровых параметров;
* создание игровой комнаты;

Для каждого подключенного игрока создаётся отдельный поток, который слушает запросы и команды игрока, передаёт команды игровой комнате, выполняет построения ответа на запрос информации игры. Более детально посмотреть на процесс взаимодействия клиента и сервера, а также внутренних потоков сервера можно на рисунке 4.

Для каждой отдельной игры создаётся игровая комната, которая осуществляет контроль игровых правил и выполняет расчёт игровых параметров: траекторий движений приведений (если они присутствуют на карте), координат всех сущностей на карте, а также выполняют обработку столкновений.

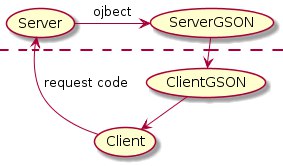


Рисунок 3 – процесс обмена сообщениями клиента и сервера.

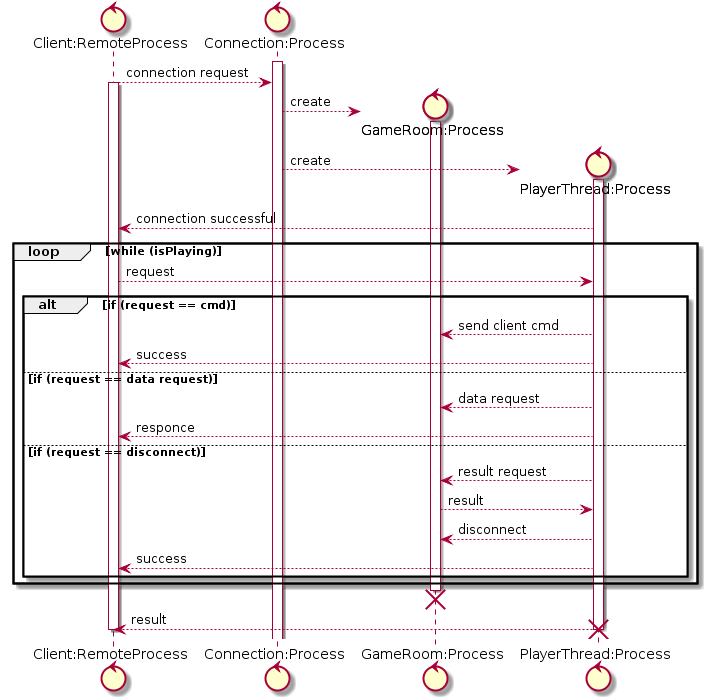


Рисунок 4 — Диаграмма последовательности для серверной части системы

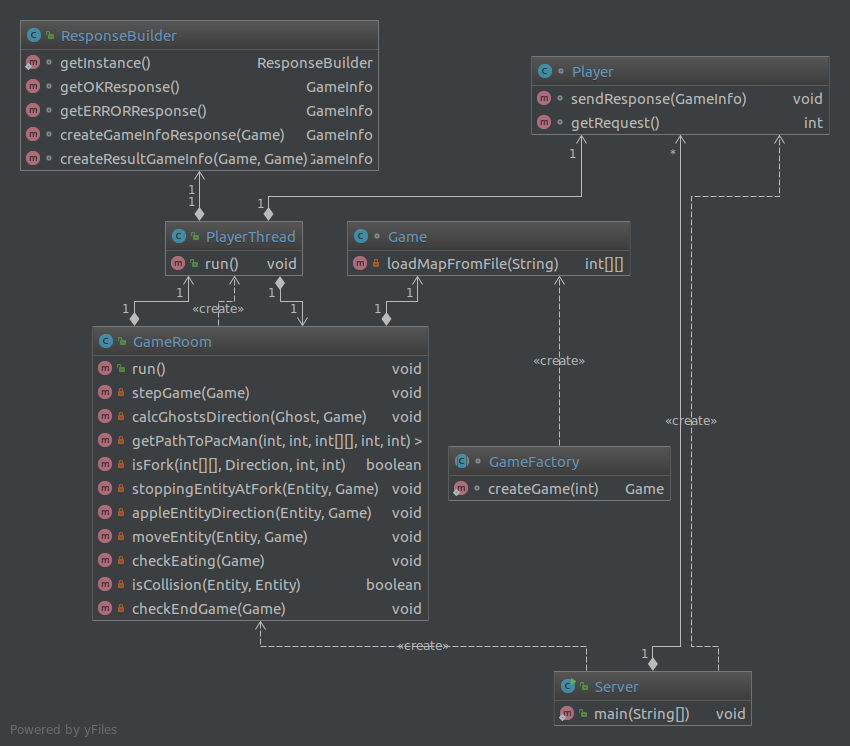


Рисунок 5 — Диаграмма классов серверной части приложения.

Server — главный класс. Отвечает за подключение новых игроков и создание новых игровых комнат.

Game — класс, хранящий состояние игры.

GameRoom — класс, реализующий процесс игры. Выполняет контроль правил игры, обработку столкновений и расчёт поведения ботов.

Player — предоставляет методы для получения и отправки сообщений клиенту.

PlayerThread — реализует общение между сервером и клиентом. Передаёт команды GameRoom, выполняет построение ответов клиенту.

ResponseBuilder — содержит в себе методы для построения ответов клиенту.

## ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

Клиентская часть приложения используется для игры человеком. При запуске пользователю предлагается ввести IP адрес и порт сервера, а также выбрать тип игры. Далее работа клиента сводится к регулярным запросам игровой ситуации, её отрисовки, а также в прослушивании нажатий кнопок клавиатуры и отправки соответствующих команд на сервер. На рисунке 5 можно увидеть процесс взаимодействия клиента и сервера, а также взаимодействие внутренних объектов клиентской части приложения.

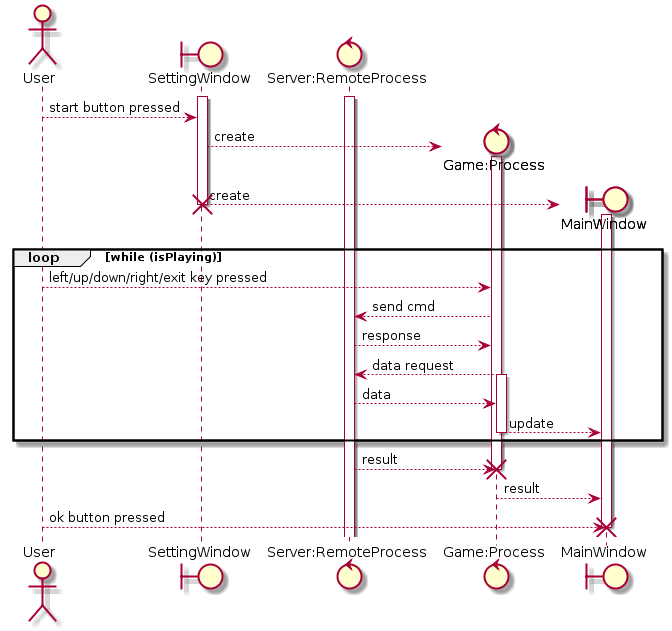


Рисунок 6 — Диаграмма последовательности для клиентской части приложения.

# НАПИСАНИЕ БОТА

Для написания и подключения бота к серверу используется соответствующее API.

boolean connection(String IP, int port, PacManAPI.GameType gameType);

Данный метод API используется для первоначального подключения к серверу. Возвращает true, если подключение успешно, иначе false.

boolean toUp();

boolean toDown();

boolean toLeft();

boolean toRight();

Набор методов API для передачи команды персонажу.

GameInfo disconnect(boolean isWait);

Метод API, используемый для отключения от сервера. Параметр isWait должен быть true, если вы хотите получить результаты игры, иначе false.

GameInfo getInfoAboutLeftPlayer();

Метод API используется для получения от сервера текущей информации об игре.

GameInfo getInfoAboutRightPlayer();

Метод API используется для получения от сервера текущей информации об игре противника (если выбран одиночный режим, то данный метод работает точно также, как и getInfoAboutLeftPlayer();)

// Описание структуры информации об игре...

public class GameInfo {

public int responseCode; // 200 — OK, 404 - ERROR

public boolean isPlaying; // Статус игры

public int[][] map; // 0 — пустая клетка, 1 — еда, 2 — стена

public ArrayList<Ghost> ghosts; // Список привидений

public PacMan pacMan; // Главный персонаж

public GameResult gameResult; // Итоговый результат игры

public ViewProperties viewProperties; // Настройки отображения

}

// Пример написания бота...

// Создание экземпляра API

IPacManAPI api = new PacManAPI();

Подключение к серверу…

if (api.connection("127.0.0.1", 7070,

PacManAPI.GameType.SINGLE\_WITHOUT\_GHOST)) {

// Цикл бота...

while (true) {

// Получение актуальной информации...

GameInfo gameInfo = api.getInfoAboutLeftPlayer();

// Анализ информации

if (gameInfo.pacMan.score != 500) {

// Отправка команды персонажу

api.toDown();

} else {

// Отключение от сервера без ожидания результатов

api.disconnect(false);

System.exit(0);

}

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

**Для развёртывания системы необходимо:**

* клонировать репозиторий Pac-Man командой:   
  **git clone https://github.com/shatoba97/Pac-Man**
* перейти в каталог ./jar
* запустить сервер командой: **java -jar server.jar <port\_number>**
* далее, либо запустить игровой клиент командой: **java -jar client.jar** либо выполнить подключение к серверу через API (см. раздел «Написание бота»).

**Для сборки частей системы в .jar выполните:**

сервер — **sh ./server\_build.sh**

клиент — **sh ./client\_build.sh**

тестовый бот — **sh ./bot\_build.sh**

api — **sh ./api\_build.sh**