Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра вычислительной техники

Допускаю к защите

Руководитель\_\_\_\_\_\_А.С. Дорофеев

подпись, И.О. Фамилия

Сравнительный анализ двух языков программирования на

примере решения игровой задачи

Наименование темы

Пояснительная записка  
к курсовой работе

по дисциплине

**«Программирование»**

1.007.00.00 ПЗ

обозначение документа

|  |  |  |  |
| --- | --- | --- | --- |
| Разработал студент группы \_\_\_\_\_\_\_\_\_ |  |  | Д.Е. Лытин |
| подпись |  | И. О. Фамилия |
|  |  |  |  |
| Нормоконтроль |  |  | А.С. Дорофеев |
| подпись |  | И. О. Фамилия |

Курсовая работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

оценка подпись

**Иркутск 2019**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра вычислительной техники

# ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

По курсу     Программирование

Студенту  Лытин Д.Е.

(фамилия, инициалы)

Тема работы Сравнительный анализ двух языков программирования на         примере решения игровой задачи

Исходные данные

Задание к теоретической части: Указатели. Примеры применения при работе с массивами;

Задание к практической части: Игра «Спасти Самурая»

Рекомендуемая литература

1. Голицына О.Л. Языки программирования: учеб. пособие. Издательство: «ИНФРА-М, Форум», 2010.

2. Программирование на С++/ В. П. Аверкин, А. И. Бобровский, В. В. Веснич и др.; Под ред. А. Д. Хомоненко . – СПб.: Корона принт, 1999. – 252 с.

3. C++Builder 5: Рук. разработчика : [В 2т.: Пер. с англ.] / Джарод Холингвэрт, Дэн Баттерфилд, Боб Сворт и др. . – М.: Вильямс, 2001-, 2001. – 824 с.

4. Архангельский, А. Я. Программирование в С++Builder 6 / А. Я. Архангельский . – М.: Бином, 2002. – 1151 с.

5. Елманова, Н. З. Введение в Borland C++ Builder / Н. З. Елманова, С. П. Кошель . – М.: Диалог-МИФИ, 1997. – 272 с.

6. Паппас, Крис Программирование на С и С++ : [Учеб. пособие : Пер. с англ.] / Крис Паппас, Уильям Мюррей . – Киев: Ирина, 2000. – 318 с.

7. Павловская Т.А. C/C++: Программирование на языке высокого уровня: Учебник для вузов, Питер, 2012.

8. [СТО 005-2015](http://www.istu.edu/docs/education/normativ/2016/sto/sto_005.pdf) СИСТЕМА МЕНЕДЖМЕНТА КАЧЕСТВА. Учебно-методическая деятельность. Оформление курсовых проектов (работ) и выпускных квалификационных работ технических специальностей.

Графическая часть на - листах.

Дата выдачи задания “09” сентября 2019 г.

Задание получил \_\_\_\_\_\_\_   Д. Е. Лытин    
 подпись И.О. Фамилия

Дата представления работы руководителю “16” декабря 2019 г.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_ А. С. Дорофеев  
 подпись И.О. Фамилия

**Содержание**

[Задание 2](#_Toc27944225)

[Введение 4](#_Toc27944226)

[1 Теоретическая часть 5](#_Toc27944227)

[1.1 Постановка задачи 6](#_Toc27944229)

[2 Практическая часть 7](#_Toc27944230)

[2.1 Постановка задачи 7](#_Toc27944231)

[2.2 Спецификация программы 7](#_Toc27944232)

[2.3 Схема иерархии Классов 10](#_Toc27944233)

[2.4 Проектирование окна приложения 12](#_Toc27944234)

[2.5 Тестовые наборы 15](#_Toc27944241)

[2.6 Блок-схемы алгоритмов 16](#_Toc27944242)

[2.7 Протоколы тестирования 20](#_Toc27944262)

[2.8 Листинги 22](#_Toc27944263)

[Заключение 40](#_Toc27944264)

[Список использованных источников 41](#_Toc27944265)

**Введение**

Задача курсовой работы заключается в создании диалоговой программы и последующем сравнении двух разных языков программирования на решении одного практического задания. Были выбраны языки Java и C#. Практическое задание представляет собой игру «Спасти Самурая». Теоретическое задание – описать, что такое указатели, и их применение при работе с массивами.

В ходе написания курсовой работы необходимо решить следующие задачи:

1) Изучить способы описания синтаксиса языков;

2) Выполнить внешнее проектирование разрабатываемых игровых программ;

3) Выполнить сравнение указанных элементов языков;

4) Программирование игровых программ на двух языках программирования;

5) реализация логики приложения;

6) Отладка и тестирование;**1 Теоретическая часть**

Java – [строго](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8_%D1%81%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle)). Приложения Java обычно [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой компьютерной архитектуре с помощью [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM). На 2019 год Java — один из самых популярных языков программирования.

С# – [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

[C#](https://ru.wikipedia.org/wiki/C_Sharp) и [Java](https://ru.wikipedia.org/wiki/Java) — два [языка программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), развивающих язык программирования [С++](https://ru.wikipedia.org/wiki/%D0%A1%2B%2B), с синтаксисом, который во многом наследует синтаксис [С++](https://ru.wikipedia.org/wiki/%D0%A1%2B%2B). Соответственно у них схожие структуры написания программ.

Примитивные типы данных в Java:

* Bool
* Byte
* Int
* Long
* Char
* Float
* Double
* Short

# Все остальные типы данных являются объектами и наследуются от класса java.lang.Object.

Примитивные типы данных в С#:

* Bool
* Byte
* Sbyte
* Ushort
* Short
* Uint
* Int
* Ulong
* Float
* Double
* Decimal
* String
* Object

В отличие от java, С# имеет специальные свойства. Это переменная любого типа, для которых можно заранее задать «мутаторы» и « аксессоры». Нет необходимости писать для этого отдельные методы.

**1.1 Постановка задачи**

Указатели. Примеры применения при работе с массивами.

**1.2 Выполнение задачи**

Указатель (англ. pointer) — переменная, диапазон значений которой состоит из адресов ячеек памяти или специального значения — нулевого адреса. Последнее используется для указания того, что в данный момент указатель не ссылается ни на одну из допустимых ячеек.

В языке Java нет указателей. Традиционно считалось, что работать с указателями трудно, а их использование приводит к появлению трудно обнаруживаемых ошибок. Поэтому разработчики Java решили отказаться от использования указателей совсем. Язык Java позволяет использовать вместо указателей ссылки на объекты. Пользуясь этими ссылками, вы можете адресовать объекты по их имени, вызывая методы и изменяя значения данных объектов.

В языке C# указатели имеются, однако к ним прибегают только в некоторых случаях. Код с указателями в C# называют небезопасным кодом. Причина этого в том, что при работе с ним все действия по использованию памяти, в том числе по ее очистке, ложится целиком на разработчика, а не на среду CLR. И с точки зрения CLR такой код не безопасен, так как среда не может проверить данный код, поэтому повышается вероятность различного рода ошибок.

Блок кода или метод, в котором используются указатели, помечается ключевым словом *unsafe*.

Ключевой при работе с указателями является операция \*, которую еще называют операцией разыменовывания. Операция разыменовывания позволяет получить или установить значение по адресу, на который указывает указатель. Для получения адреса переменной применяется операция &.

При объявлении указателя указываем тип *int\* x;* - в данном случае объявляется указатель на целое число. Но кроме типа int можно использовать и другие. Также можно объявлять указатели на типы enum, структуры и другие указатели.

Например:

*unsafe*

*{*

*int\* x; // определение указателя*

*int y = 10; // определяем переменную*

*x = &y; // указатель x теперь указывает на адрес переменной y*

*}*

Выражение *x = &y;* позволяет нам получить адрес переменной *y* и установить на него указатель *x*. До этого указатель x не на что не указывал.

После этого все операции с *y* будут влиять на значение, получаемое через указатель *x* и наоборот, так как они указывают на одну и ту же область в памяти.

Для получения значения, которое хранится в области памяти, на которую указывает указатель *x*, используется выражение *\*x*.

Работа с массивами с помощью указателей в C#.

С помощью ключевого слова *stackalloc* можно выделить память под массив в стеке. Смысл выделения памяти в стеке в повышении быстродействия кода. Оператор *stackalloc* принимает после себя массив, на который будет указывать указатель. *int\* mas = stackalloc int[size];*. Для манипуляций с массивом создаем указатель: *int\* p = mas;*, который указывает на первый элемент массива. Чтобы получить следующий элемент и сместиться вперёд, можно использовать операцию инкремента *\*(++p)*, а для предыдущего - декремента *\*(--p)*. Обращение к указателям как к массивам представляет альтернативу операции разыменовывания для получения значения. Например, *p[-1]* получает значение предыдущего элемента.

**2 Практическая часть**

**2.1 Постановка задачи**

Индивидуальное задание было выбрано индивидуальное, и заключается в разработке игры «Спасти Самурая». Описание игры: с четырёх направлений летят стрелы, управляя Самураем, который может принимать четыре положения (на каждое направление стрел), требуется поймать щитом как можно большее количество стрел. За пойманную стрелу игроку выдаётся одно очко. Сначала в сторону Самурая медленно летит только одна стрела, а затем их количество и скорость увеличивается. В случае, если стрела долетела до Самурая, а Самурай её не поймал, у игрока отбирается одна из пяти жизней. Игра заканчивается, когда у Самурая заканчиваются все жизни.

**2.2 Спецификация программы**

В Таблице 2 продемонстрированы классы и методы, использованные в программах как на Java, так и на С#

Таблица 2 – Спецификация программ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название** | **Назначение** | **Методы (без методов get и set)** | **Поля** |
| **Java** | | | | |
| **1** | Main | Класс, запускающий приложение | main(); | int FRAME\_HEIGHT;  int FRAME\_WIDTH; |
| **2** | Launcher | Меню игры | Launcher();  newGame();  showTop();  showHowTo(); | JButton butNew;  JButton butTop;  JButton butHowTo;  JButton butExit;  JPanel pane;  Font fontForButtons; |
| **3** | GameFrame | Игровое окно | GameFrame(GameLogic game);  closeFrame(); | GameLogic game; |
| **4** | GameLogic | Интерфейс игровой логики | sendKey(int key);  resume();  getScore();  pause();  isOver();  GameField getGameField(); |  |
| **5** | GameField | Игровое поле | GameField();  add(Drawable element);  paintComponent(Graphics gr); | int GAME\_FIELD\_WIDTH;  int GAME\_FIELD\_HEIGHT;  Image background;  ArrayList<Drawable> elements; |
| **6** | Drawable | Интерфейс рисуемых объектов | draw(Graphics gr); |  |
| **7** | Game | Игровая логика | Game();  update();  gameOver();  addLevel();  sendKey(int key);  resume();  getScore();  pause();  isOver();  GameField getGameField(); | int GAME\_TICK;  int MIN\_DELAY;  GameField gameField;  Samurai samurai;  Arrow[] arrows;  Timer timer;  ValueString score;  ValueString life;  int speed;  int mainDelay;  private int delay;  private int level;  Banner pauseSign;  boolean isOver;  Banner overSign; |
| **8** | GameObject | Игровые объекты | draw(Graphics gr); | Image texture;  int x;  int y;  int width;  int height; |
| **9** | Arrow | Стрела | Arrow();  spawn();  moove(int speed);  isHitObject(GameObject object);  respawn();  getSpawnPlace();  hide(); | int TEXTURE\_WIDTH;  int TEXTURE\_HEIGHT;  int TRACE\_HEIGHT;  int TRACE\_WIDTH;  int TOP\_INDENT;  int INDENT;  int UP\_TRACE\_Y;  int DOWN\_TRACE\_Y;  Image[] images;  int direction = 1;  boolean hidden; |
| **10** | Samurai | Самурай | Samurai();  turn(int direction); | int TEXTURE\_WIDTH;  int TEXTURE\_HEIGHT;  Image[] images;  int direction; |
| **11** | Banner | Надпись в центре экрана | Banner(String text);  Banner(String text, Color color);  setVisible(boolean arg);  draw(Graphics gr); | int FONT\_HEIGHT;  int FONT\_WIDTH;  Font font;  int x;  int y;  String text;  Color color;  boolean visible; |
| **12** | ValueString | Строка состояния | ValueString(String string, int x, int y);  ValueString(String string, int startValue, int x, int y);  draw(Graphics gr); | int FONT\_HEIGHT;  Font font;  String string;  int value;  int x, y; |
| **13** | Files | Работа с файлами | Files();  getTextures(int count, filesName);  getTexture(String filesName);  changeTop(int score, String name);  getTop(); |  |
| **C#** | | | | |
| **1** | Program | См. спецификации программы > Java > Main | | |
| **2** | Launcher |  | Launcher();  butNew\_Click(Object sender, EventArgs e);  butTop \_Click(Object sender, EventArgs e);  butHowTo \_Click(Object sender, EventArgs e);  butExit \_Click(Object sender, EventArgs e);  newGame();  showTop();  showHowTo();  InitializeComponent(); | IContainer components;  Panel pane;  Button butNew;  Button butTop;  Button butHowTo;  Button butExit; |
| **3** | GameFrame |  | GameFrame(GameLogic game);  GameFrame\_FormClosed(Object sender, FormClosedEventArgs e);  GameFrame\_Activated(Object sender, EventArgs e);  GameFrame\_Deactivate(Object sender, EventArgs e);  GameFrame\_KeyDown(Object sender, KeyEventArgs e);  GameFrame\_FormClosing(Object sender, FormClosingEventArgs e);  closeFrame();  InitializeComponent(); | GameLogic game;  IContainer components; |
| **4** | InputBox |  | InputBox();  GetString();  Button\_Click(Object sender, EventArgs e); | TextBox t1;  String returnString; |
| **5** | GameLogic | См. спецификации программы > Java > GameLogic | | |
| **6** | GameField |  | GameField();  add(Drawable element);  GameField\_Paint(Object sender, PaintEventArgs e); | Image background;  List<Drawable> elements;  Graphics gr;  int GAME\_FIELD\_WIDTH;  int GAME\_FIELD\_HEIGHT; |
| **7** | Drawable | См. спецификации программы > Java > Drawable | | |
| **8** | Game | См. спецификации программы > Java > Game | | |
| **9** | GameObject | См. спецификации программы > Java > GameObject | | |
| **10** | Arrow | См. спецификации программы > Java > Arrow | | |
| **11** | SpawnPlace |  | SpawnPlace(int x, int y) | int x;  int y;  SpawnPlace RightUp;  SpawnPlace RightDown;  SpawnPlace LeftDown;  SpawnPlace LeftUp;  SpawnPlace Hide; |
| **12** | Samurai | См. спецификации программы > Java > Samurai | | |
| **13** | Banner |  | Banner(String text);  Banner(String text, Color color);  Banner();  setVisible(Boolean arg);  draw(Graphics gr); | int x;  int y;  String text;  SolidBrush brush;  bool visible;  int FONT\_HEIGHT;  int FONT\_WIDTH;  Font font; |
| **14** | ValueString | См. спецификации программы > Java > ValueString | | |
| **15** | Files | См. спецификации программы > Java > Files | | |

**2.3 Схема иерархии Классов**

Иерархия классов у разных языков различается. Схемы представлены на Рисунке 1 и Рисунке 2.

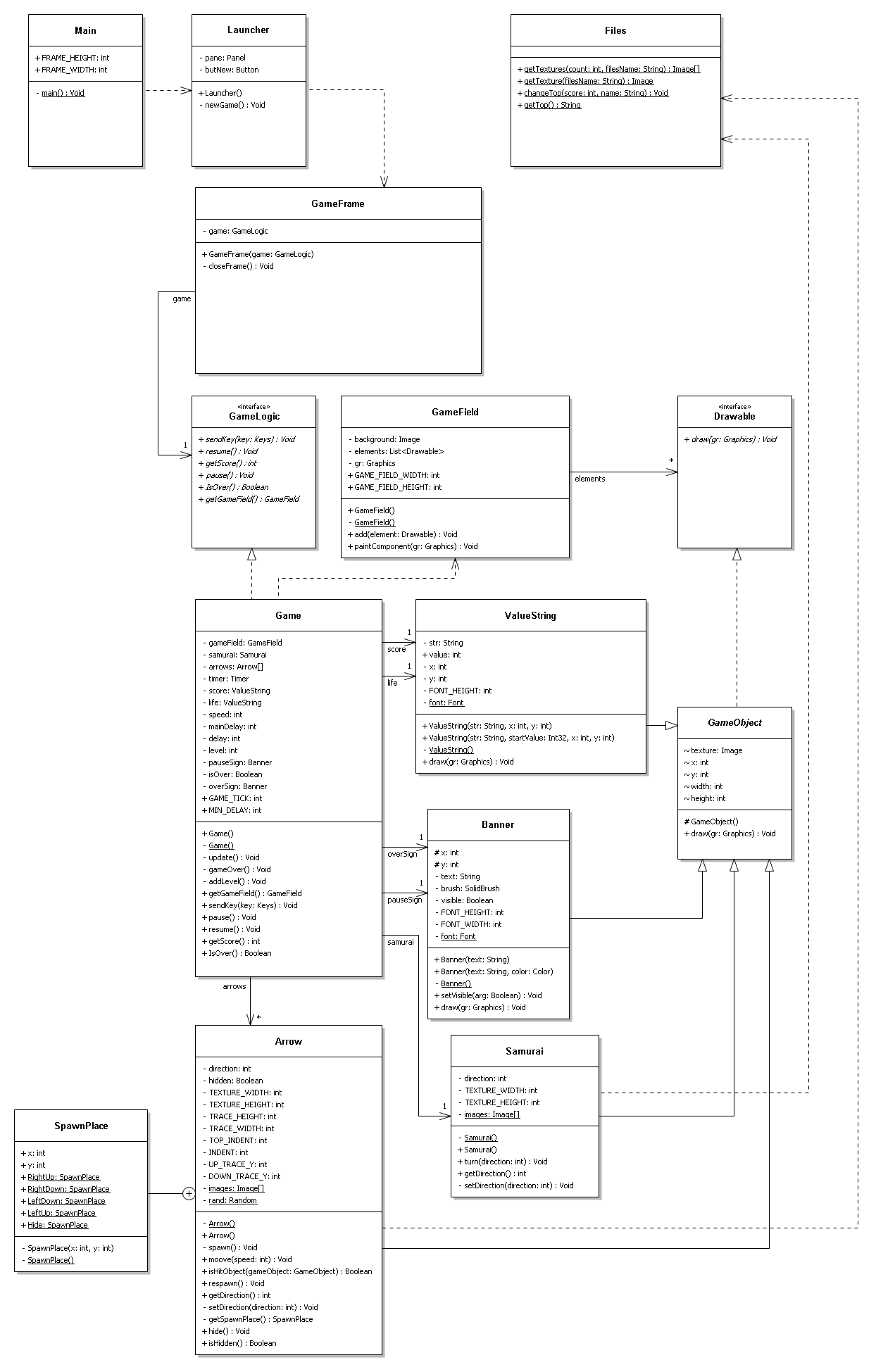


Рисунок 1.1 - Схема иерархии классов Java

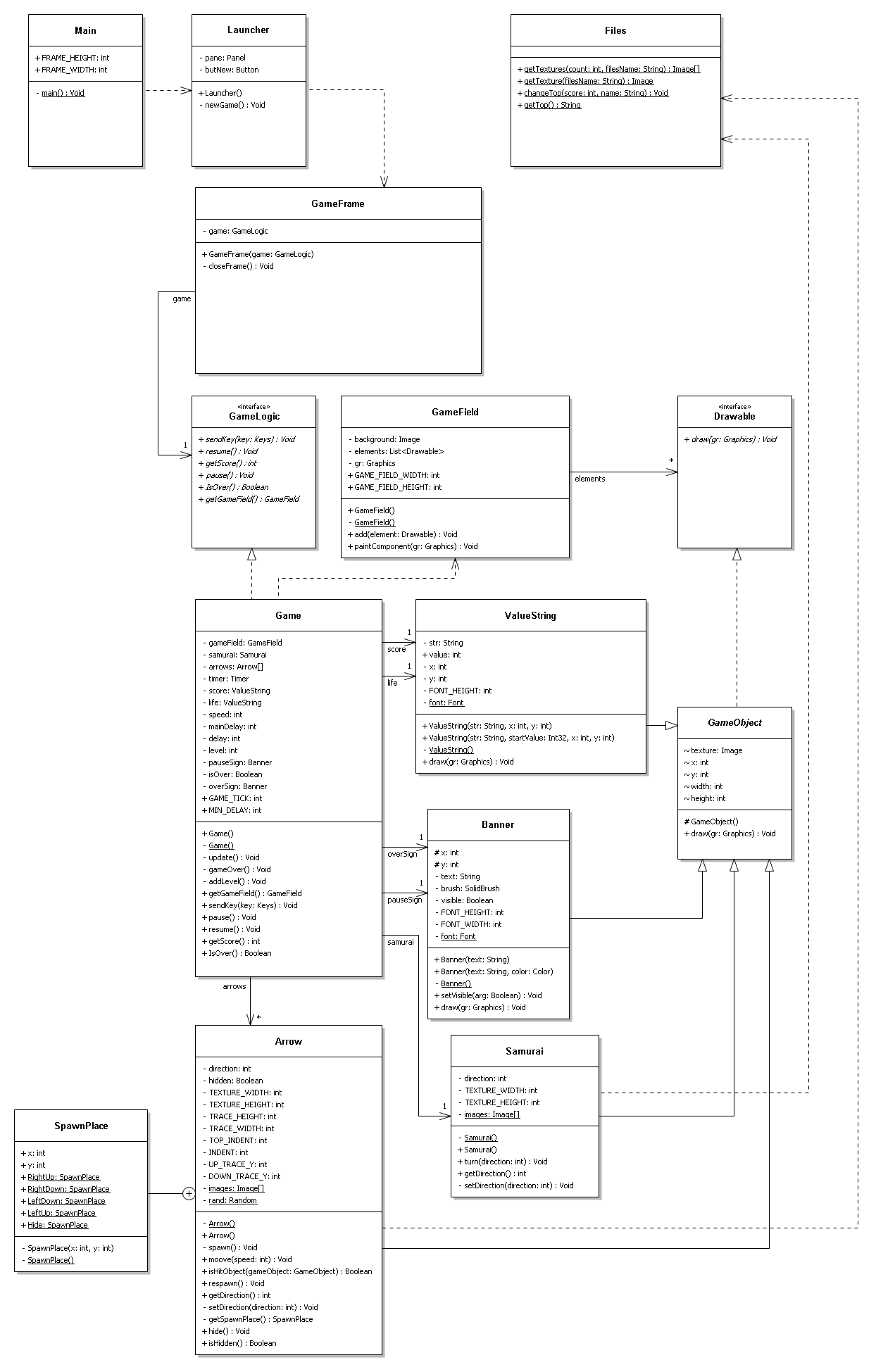


Рисунок 1.2 - Схема иерархии классов Java (продолжение)

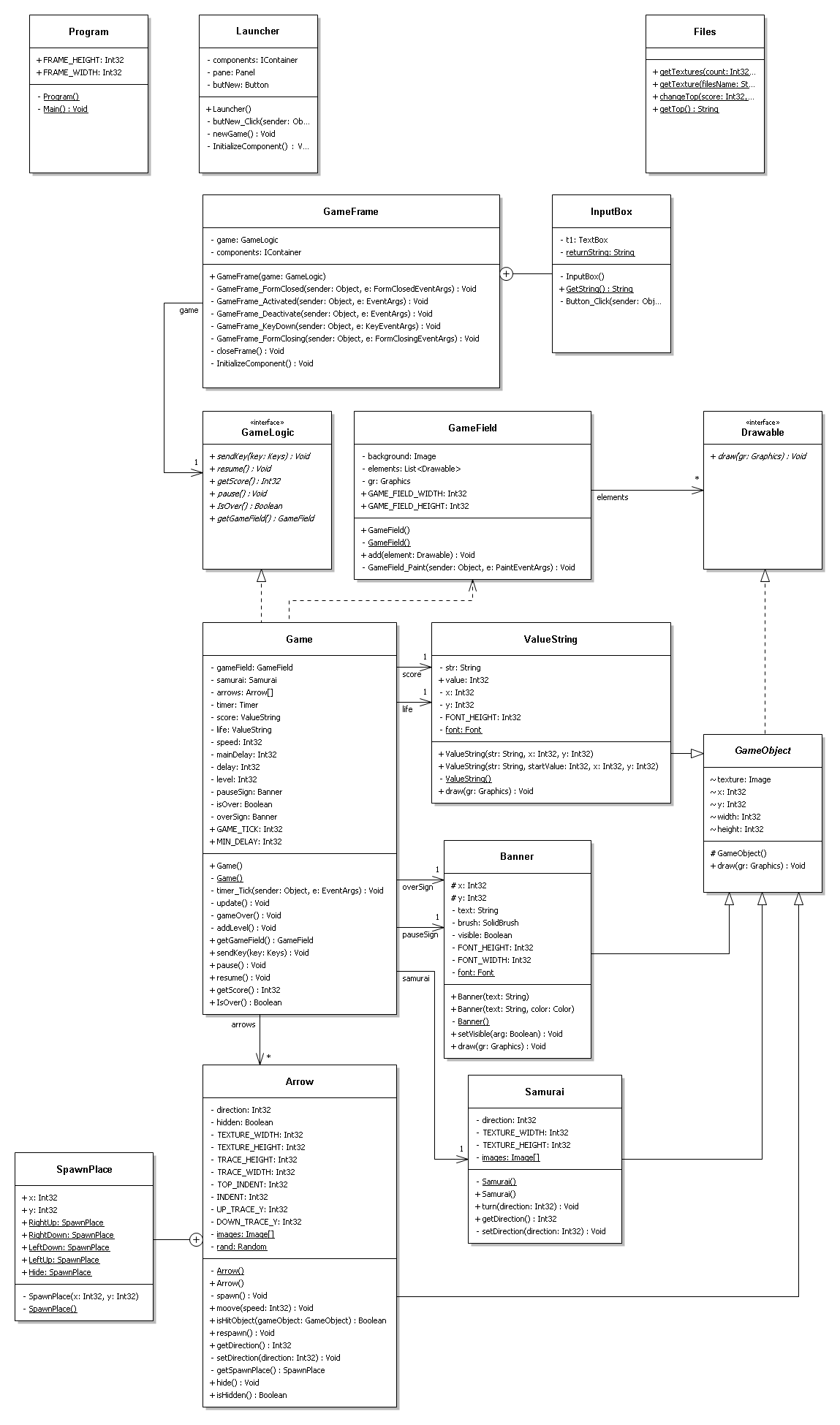


Рисунок 2.1 - Схема иерархии классов C#

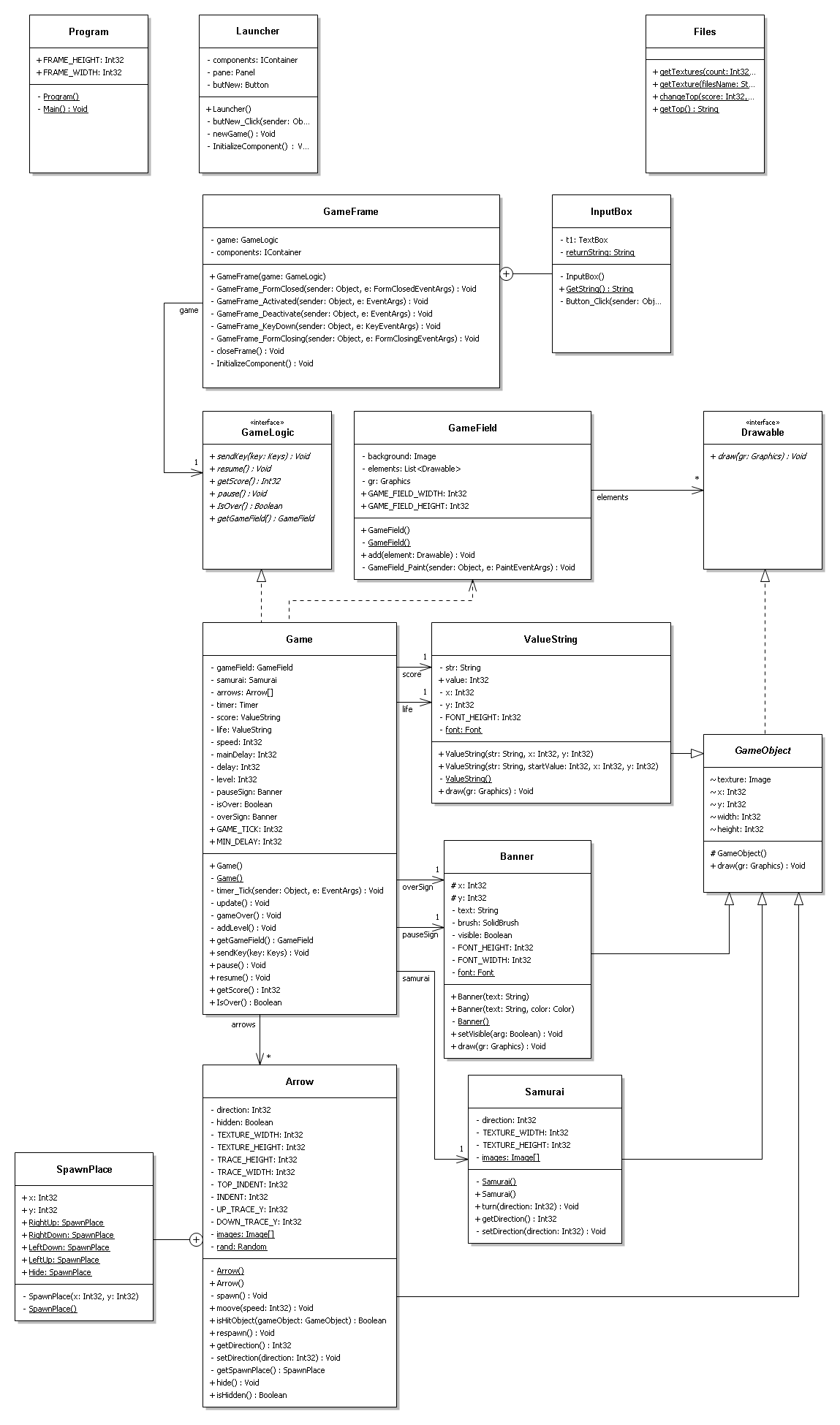


Рисунок 2.2 - Схема иерархии классов C# (продолжение)

**2.4 Проектирование окна приложения**

Так как была поставлена задача реализовать игру, то для удобства пользователя всё игровое управление реализовано через нажатие клавиш клавиатуры. Сделана возможность двух видов управления: для правой и левой руки. Управление правой рукой осуществляется через NumPad, а управление левой - через буквенные клавиши. Все информация о клавиша управления указана в правилах игры. Так как управление игрой происходит через клавиатуру, то окно с игрой не содержит элементов взаимодействия с пользователем. Игровое

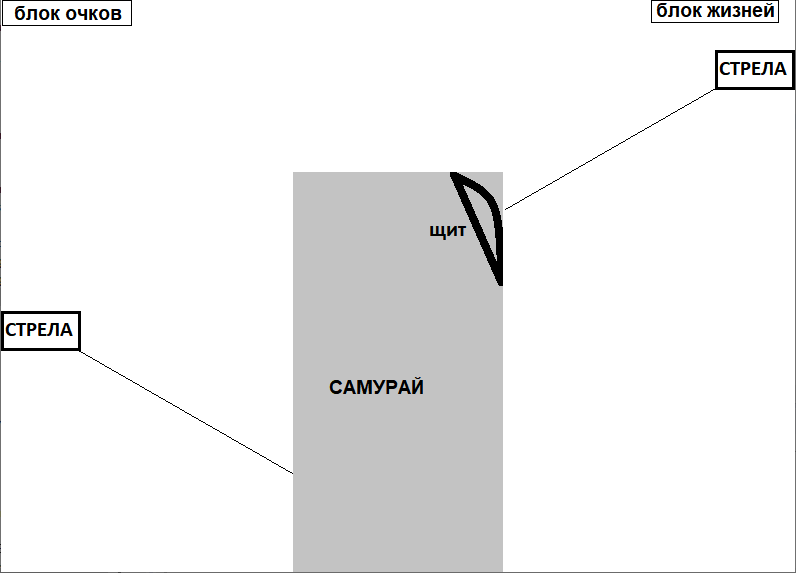


Рисунок 3 – Игра, проект

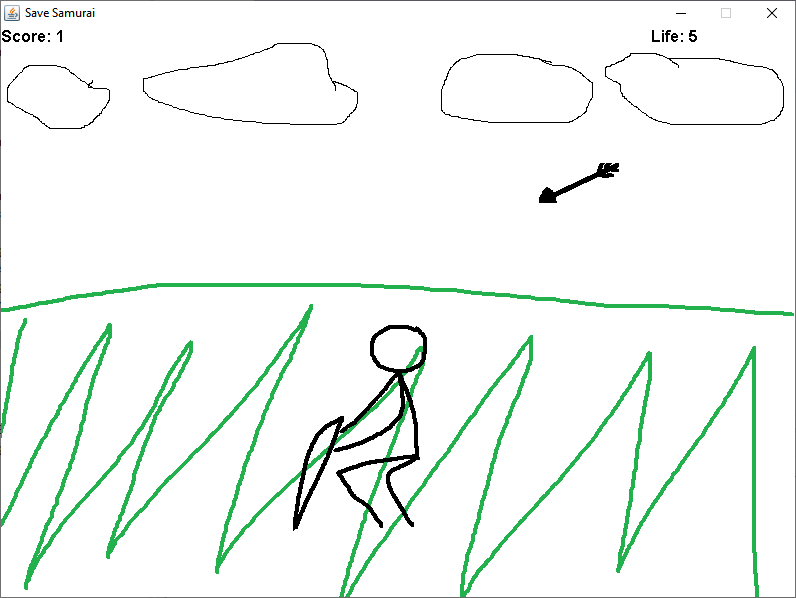


Рисунок 4 – Игра, результат

Так же было разработано окно меню, где пользователь мог ознакомится с правилами и лучшим результатом игры. Меню предполагалось в минималистичном формате, без элементов графики.

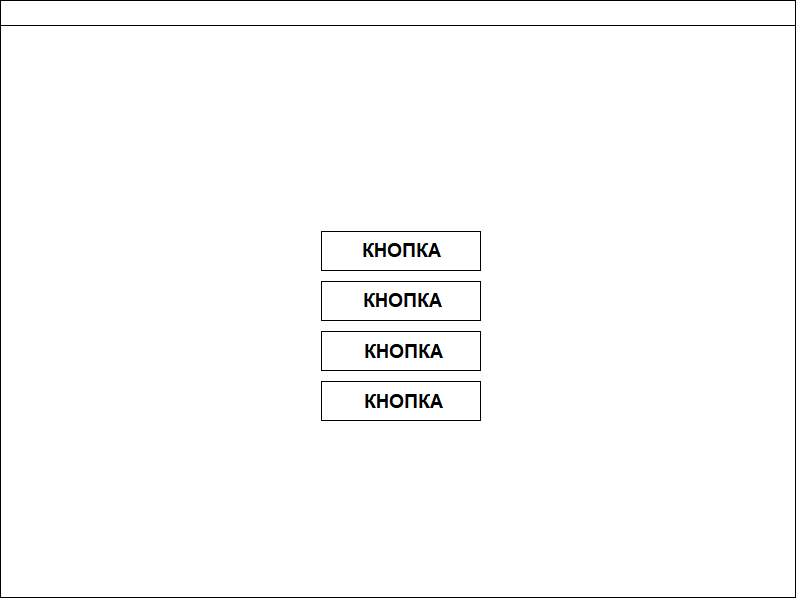


Рисунок 5 – Меню, проект

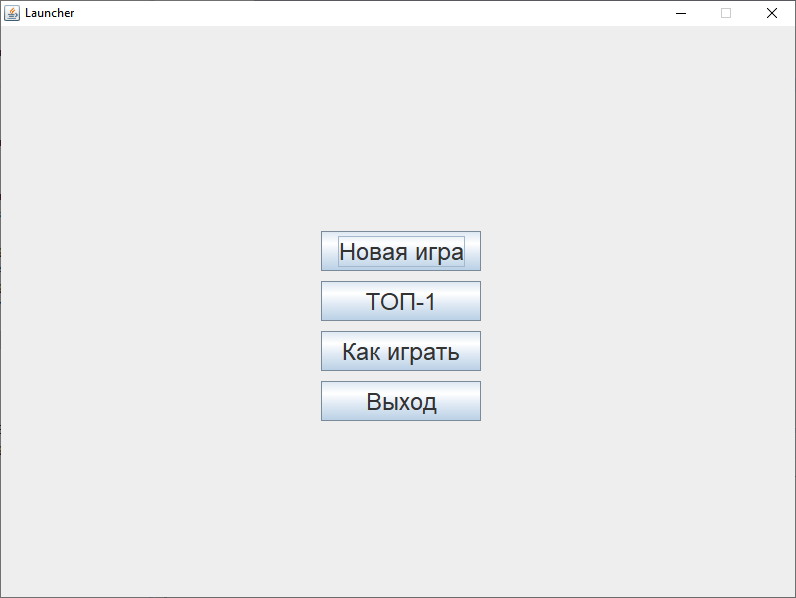


Рисунок 6 – Меню, результат

Все диалоговые окна планировалось реализовать встроенными библиотеками языков. Так, например окно с правилами игры: окно с информационным сообщением для пользователя, сообщение в котором и есть правила.

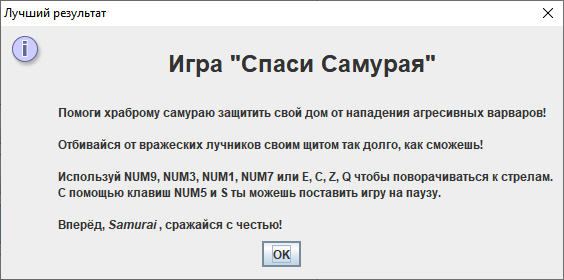


Рисунок 7 – Правила

В ходе разработки приложения на языке C# возникла проблема: во встроенных библиотеках не оказалось возможности показать пользователю диалоговое окно с возможностью текстового ввода, поэтому на языке C# была реализована форма диалогового окна с полем ввода.



Рисунок 8 - Диалоговое окно с вводом

**2.5 Тестовые наборы**

Во время тестирования будут произведены все возможные попытки вывести программу из строя. Варианты попыток представлены ниже, в   
Таблице 3.

Таблица 3 – Варианты тестов возможных ошибок

|  |  |
| --- | --- |
| № | Название испытания |
| 1 | Запуск игры без файлов текстур |
| 2 | Закрытие окна после победы с новым рекордом |
| 3 | Достижение максимальной сложности |
| 4 | Управление во время паузы |

**2.6 Блок-схемы алгоритмов**

Ниже представлены блок-схемы основных алгоритмов механики игры.

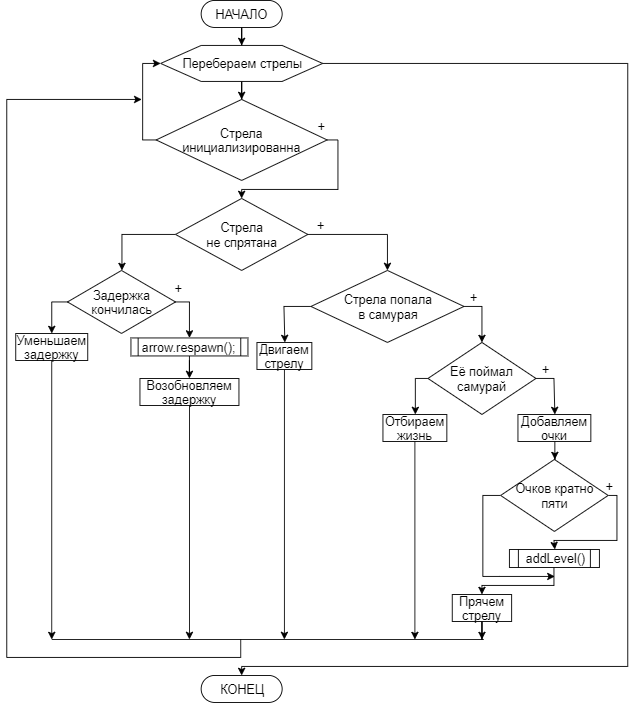


Рисунок 9 – Игровой цикл

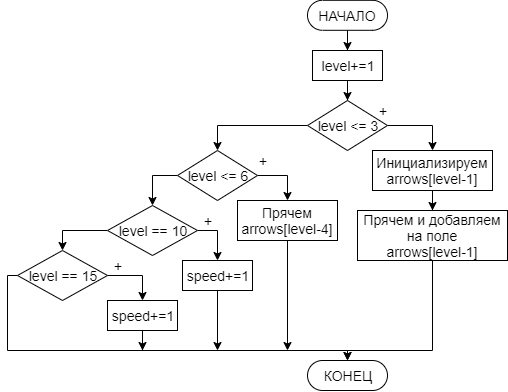


Рисунок 9 – addLevel()

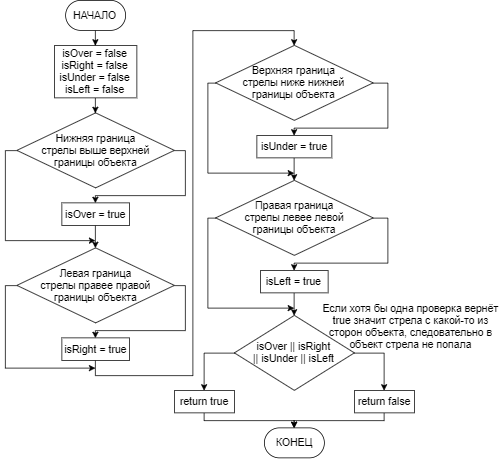


Рисунок 9 – проверка касания стрелой объекта

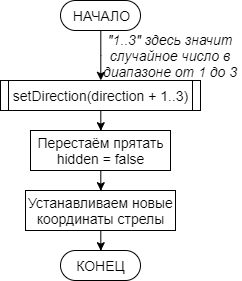


Рисунок 9 – Arrow.respawn()

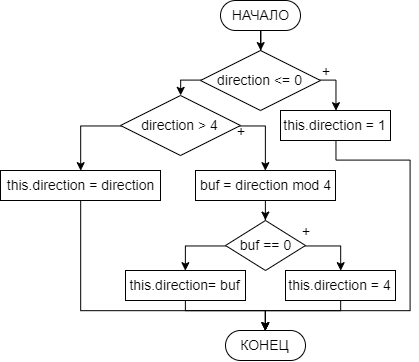


Рисунок 9 – Arrow.setDirection()

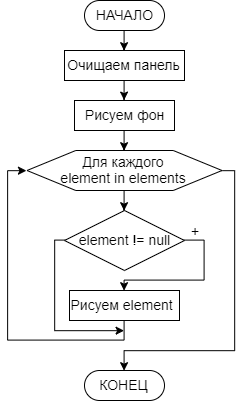


Рисунок 9 – Прорисовка игрового поля

**2.7 Протоколы тестирования**

Ещё на этапе проектирования, я избежал всяческой возможности появления ошибок, при помощи некоторых ограничений действия пользователя. Результаты тестов из Таблицы 3 представлены в следующей Таблице 4.

Таблица 4 – Тестирование

|  |  |  |
| --- | --- | --- |
| **№** | **Результат** | **Пояснение** |
| 1 | Выдача окна об ошибке, в котором содержится путь к файлу отсутствующей текстуры, и завершение приложения | Приложение завершается, так как его работа без визуальной составляющей не имеет смысла. |
| 2 | Приложение запрашивает имя игрока | Не важно, как пользователь выйдет из игры, ему всё равно будет предложено сохранить его результат. |
| 3 | Максимальной сложности добиться возможно | Сложность перестаёт увеличиваться после набранных 75 очков, согласно алгоритму. |
| 4 | “Самурай” меняет своё положение после выхода из режима паузы | Клавиши управления считываются даже в режиме паузы. Это было продуманно с целью того, что внимательный игрок может использовать эту возможность, и набирать максимальное количество очков, не смотря на скорость летящих “стрел”. |

**2.8 Листинги**

Ниже предоставлено кодирование программы на двух языках (Java, C#).

Java:

package com.kislov.java\_game;

import java.awt.\*; // Формы

import java.awt.event.\*; // Нажатия на кнопки

import javax.swing.\*; // swing

/\*\*

\*

\* @author realg

\*/

public class Main {

private static Gui game;

private static Menu menu;

private static Gamefield gamefield;

private static JFrame j = new JFrame("Rock-n-roll Gems");

private static JPanel p;

public static void close(){

//game.dispose();

//menu.dispose();

j.dispose();

System.exit(0);

};

public static void switchToMenu(){

//game.setVisible(false);

//menu.setVisible(true);

Container c = j.getContentPane();

c.removeAll();

menu.addReplayButton();

p = menu;

c.add(p);

j.validate();

j.repaint();

}

public static void switchToGame(){

//menu.setVisible(false);

//game.setVisible(true);

Container c = j.getContentPane();

c.removeAll();

p = game;

c.add(p);

j.validate();

j.repaint();

}

public static void restart(){

Container c = j.getContentPane();

c.removeAll();

gamefield = new Gamefield();

game = new Gui(gamefield);

menu = new Menu();

gamefield.addGui(game);

gamefield.prepareField();

p = menu;

c.add(p);

j.validate();

j.repaint();

switchToGame();

}

public static void main(String[] args) {

gamefield = new Gamefield();

game = new Gui(gamefield);

menu = new Menu();

gamefield.addGui(game);

gamefield.prepareField();

p = menu;

j.add(p);

Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

j.setBounds(dim.width/2-285, dim.height/2-225, 570, 450);

j.setResizable(false);

//menu.setVisible(true);

j.setVisible(true);

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.kislov.java\_game;

import java.awt.\*; // Формы

import java.awt.event.\*; // Нажатия на кнопки

import javax.swing.\*; // swing

/\*\*

\*

\* @author realg

\*/

//public class Menu extends JFrame {

public class Menu extends JPanel {

//private static Menu menu;

private JButton play = new JButton("Играть");

private JButton rules = new JButton("Правила");

private JButton exit = new JButton("Выйти");

private JPanel panel = new JPanel();

private GridBagConstraints gbc;

private JPanel buttons;

public Menu(){

//super("Rock-n-roll gems");

//Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

//this.setBounds(dim.width/2-285, dim.height/2-225, 570, 450);

//this.setResizable(false);

setLayout(new GridBagLayout());

gbc = new GridBagConstraints();

gbc.anchor = GridBagConstraints.CENTER;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.gridwidth = GridBagConstraints.REMAINDER;

buttons = new JPanel(new GridBagLayout());

buttons.add(play, gbc);

buttons.add(rules, gbc);

buttons.add(exit, gbc);

gbc.weighty = 1;

add(buttons, gbc);

play.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Играть

Main.switchToGame();

}

});

rules.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFrame j = new JFrame("Правила");

Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

j.setBounds(dim.width/2-150, dim.height/2-75, 300, 150);

j.setResizable(false);

JTextArea text = new JTextArea(" Игра три в ряд:\n"

+ " Перемещаясь по полю при помощи кнопок\n"

+ " выберите желаемый квадрат.\n"

+ " затем переместите его при помощи кнопок\n"

+ " если в результате действия сложится 3\n"

+ " квадрата в ряд, вы получите очки");

text.setEditable(false);

j.add(text);

j.setVisible(true);

}

});

exit.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Main.close();

}

});

}

public void addReplayButton(){

JButton replay = new JButton("Заново");

replay.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Main.restart();

}

});

buttons.add(replay, gbc);

}

}

package com.kislov.java\_game;

import java.awt.\*; // Формы

import java.awt.event.\*; // Нажатия на кнопки

import javax.swing.\*; // swing

/\*\*

\*

\* @author realg

\*/

//public class Gui extends JFrame {

public class Gui extends JPanel {

//private static Gui gui = null;

private Gamefield gamefield;

private boolean allowActions;

private JButton left = new JButton("←");

private JButton right = new JButton("→");

private JButton up = new JButton("↑");

private JButton down = new JButton("↓");

private JButton pick = new JButton("Выбор");

private JButton menu = new JButton("Пауза");

private JButton exit = new JButton("Выйти");

private JLabel score = new JLabel("Счёт: 0");

private Panel panel;

private JPanel btnPanel = new JPanel();

public Gui(Gamefield gamefield){

allowActions = true;

this.gamefield = gamefield;

panel = new Panel(gamefield.getMap());

this.add(btnPanel, BorderLayout.WEST); // панель для кнопок

this.add(panel, BorderLayout.CENTER); // панель с игрой

panel.setPreferredSize(new Dimension(400,400));

btnPanel.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

c.insets = new Insets(2, 2, 2, 2);

c.gridx = 1;

c.gridy = 0;

btnPanel.add(up, c);

c.gridx = 0;

c.gridy = 1;

btnPanel.add(left, c);

c.gridx = 1;

c.gridy = 1;

btnPanel.add(down, c);

c.gridx = 2;

c.gridy = 1;

btnPanel.add(right, c);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridwidth = 3;

c.gridx = 0;

c.gridy = 2;

btnPanel.add(pick, c);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 0;

c.gridy = 3;

btnPanel.add(menu, c);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 0;

c.gridy = 4;

btnPanel.add(score, c);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 0;

c.gridy = 5;

btnPanel.add(exit, c);

left.addActionListener(new ActionListener(){ // abstract class

@Override

public void actionPerformed(ActionEvent e){

if (allowActions)

gamefield.movePickerLeft();

updatePanel();

}

});

right.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

if (allowActions)

gamefield.movePickerRight();

updatePanel();

}

});

up.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

if (allowActions)

gamefield.movePickerUp();

updatePanel();

}

});

down.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

if (allowActions)

gamefield.movePickerDown();

updatePanel();

}

});

pick.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

if (allowActions)

gamefield.pick();

updatePanel();

}

});

exit.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

Main.close();

}

});

menu.addActionListener(new ActionListener(){ // абстрактный класс

@Override

public void actionPerformed(ActionEvent e){

// Пауза

Main.switchToMenu();

}

});

}

private void updatePanel(){

panel.repaint();

}

public void deactivate(){

allowActions = false;

}

public void activate(){

allowActions = true;

}

public void updateScore(){

score.setText("Счёт: "+Integer.toString(gamefield.getScore()));

}

}

package com.kislov.java\_game;

import java.awt.\*;

import javax.swing.\*;

public class Panel extends JPanel {

private Gem[][] map;

public Panel(Gem[][] map){

setSize(400,400);

this.map = map;

}

@Override

public void paintComponent(Graphics g){

super.paintComponent(g);

drawTable(g);

}

private void setBrush(Graphics2D g2, Gem gem){

switch (gem.getType()){

case 1:

g2.setColor(Color.yellow);

break;

case 2:

g2.setColor(Color.red);

break;

case 3:

g2.setColor(Color.green);

break;

case 4:

g2.setColor(Color.blue);

break;

case 5:

g2.setColor(Color.magenta);

break;

}

}

private void drawTable(Graphics g){

Graphics2D g2 = (Graphics2D) g;

for (int i = 0; i < map[0].length; i++){

for (int j = 0; j < map[0].length; j++){

setBrush(g2, map[i][j]);

g.drawRect(j\*50+2, i\*50+2, 46, 46);

g.fillRect(j\*50+2, i\*50+2, 46, 46);

if (map[i][j].isSelected()){

g2.setColor(Color.black);

g.drawRect(j\*50+1, i\*50+1, 48, 48);

}

if (map[i][j].isPicked()){

g2.setColor(Color.black);

g.drawRect(j\*50+4, i\*50+4, 42, 42);

g2.setColor(Color.white);

g.drawRect(j\*50+5, i\*50+5, 40, 40);

g2.setColor(Color.black);

g.drawRect(j\*50+6, i\*50+6, 38, 38);

}

}

}

}

}

package com.kislov.java\_game;

public class Gamefield {

private int size;

private Gem[][] map;

private int pickX;

private int pickY;

private int score;

private int[] pair;

private Gui gui;

public Gamefield(){

size = 8;

map = new Gem[8][8];

generate();

pickX = 0;

pickY = 0;

score = 0;

pair = new int[] {-1, -1, -1, -1};

setGemAsPicked();

}

public void addGui(Gui gui){

this.gui = gui;

}

public void movePickerUp(){

if(pickY > 0){

setGemAsUnpicked();

pickY = pickY - 1;

setGemAsPicked();

doSecondPick();

}

}

public void movePickerDown(){

if(pickY < size - 1){

setGemAsUnpicked();

pickY = pickY + 1;

setGemAsPicked();

doSecondPick();

}

}

public void movePickerLeft(){

if(pickX > 0){

setGemAsUnpicked();

pickX = pickX - 1;

setGemAsPicked();

doSecondPick();

}

}

public void movePickerRight(){

if(pickX < size - 1){

setGemAsUnpicked();

pickX = pickX + 1;

setGemAsPicked();

doSecondPick();

}

}

public void pick(){

if (pair[0] == -1){

pair[0] = pickY;

pair[1] = pickX;

setGemAsSelected();

} else {

pair[0] = pair[1] = pair[2] = pair[3] = -1;

setGemAsUnselected();

}

}

private void setGemAsPicked(){

map[pickY][pickX].setPicked();

}

private void setGemAsUnpicked(){

map[pickY][pickX].setUnpicked();

}

private void setGemAsSelected(){

map[pickY][pickX].setSelected();

}

private void setGemAsUnselected(){

map[pickY][pickX].setUnselected();

}

private void generate(){// Generates the game field

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

for (int j = 0; j < size; j++){

this.map[i][j] = new Gem();

}

}

}

private void doSecondPick(){

if(pair[0] != -1){

startCollectingOperation();

pair[2] = pickY;

pair[3] = pickX;

swap();

findRemovable(false);

}

}

private void startCollectingOperation(){

gui.deactivate();

}

private void endCollectingOperation(){

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

for (int j = 0; j < size; j++){

if(map[i][j].isSelected()){

map[i][j].setUnselected();

}

if(map[i][j].isPicked()){

map[i][j].setUnpicked();

}

}

}

setGemAsPicked();

pair[0] = pair[1] = pair[2] = pair[3] = -1;

gui.activate();

}

private void swap(){

Gem temp;

temp = map[ pair[0] ][ pair[1] ];

map[ pair[0] ][ pair[1] ] = map[ pair[2] ][ pair[3] ];

map[ pair[2] ][ pair[3] ] = temp;

}

private void findRemovable(boolean recursion){

boolean removableFound = false;

/\* rows \*/

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

for (int j = 0; j < size-2; j++){

if(this.map[i][j].getType() == this.map[i][j+1].getType() && this.map[i][j+1].getType() == this.map[i][j+2].getType()){

this.map[i][j].destroy();

this.map[i][j+1].destroy();

this.map[i][j+2].destroy();

addScore();

removableFound = true;

}

}

}

/\* columns \*/

for (int i = 0; i < size-2; i++){

for (int j = 0; j < size; j++){

if(this.map[i][j].getType() == this.map[i+1][j].getType() && this.map[i+1][j].getType() == this.map[i+2][j].getType()){

this.map[i][j].destroy();

this.map[i+1][j].destroy();

this.map[i+2][j].destroy();

addScore();

removableFound = true;

}

}

}

if (!removableFound && !recursion){

if (pair[0] != -1 ) swap();

endCollectingOperation();

}

else if (!removableFound && recursion) endCollectingOperation();

else collectGems();

}

public void prepareField(){

startCollectingOperation();

while (checkIfNoMoreMoves()) generate();

findRemovable(false);

score = 0;

gui.updateScore();

}

private void collectGems(){

popup();

dropdown();

while (checkIfNoMoreMoves()) generate();

findRemovable(true);

}

private void popup(){

Gem temp;

for(int k = 0; k < size; k++){

for(int i = 1; i < size; i++){

for(int j = 0; j < size; j++){

if (map[i][j].getType() == 0){

temp = map[i][j];

map[i][j] = map[i-1][j];

map[i-1][j] = temp;

}

}

}

}

}

private void dropdown(){

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

for (int j = 0; j < size; j++){

if(map[i][j].getType() == 0){

map[i][j] = new Gem();

}

}

}

}

private void addScore(){

score += 100;

gui.updateScore();

}

private boolean checkIfNoMoreMoves(){

for (int i = 0; i < size-1; i++){ // iterations for [ ### ] blocks

for (int j = 0; j < size-2; j++){ // [ ### ]

if(

(this.map[i][j].getType() == this.map[i][j+1].getType() && this.map[i][j+1].getType() == this.map[i+1][j+2].getType()) ||

(this.map[i+1][j].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i][j+2].getType()) ||

(this.map[i+1][j].getType() == this.map[i][j+1].getType() && this.map[i][j+1].getType() == this.map[i][j+2].getType()) ||

(this.map[i][j].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i+1][j+2].getType()) ||

(this.map[i][j].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i][j+2].getType()) ||

(this.map[i+1][j].getType() == this.map[i][j+1].getType() && this.map[i][j+1].getType() == this.map[i+1][j+2].getType())

) return false;

}

}

// [ ## ]

for (int i = 0; i < size-2; i++){ // iterations for [ ## ] blocks

for (int j = 0; j < size-1; j++){ // [ ## ]

if(

(this.map[i][j+1].getType() == this.map[i+1][j].getType() && this.map[i+1][j].getType() == this.map[i+2][j+1].getType()) ||

(this.map[i][j+1].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i+2][j].getType()) ||

(this.map[i][j+1].getType() == this.map[i+1][j].getType() && this.map[i+1][j].getType() == this.map[i+2][j].getType()) ||

(this.map[i][j].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i+2][j+1].getType()) ||

(this.map[i][j+1].getType() == this.map[i+1][j+1].getType() && this.map[i+1][j+1].getType() == this.map[i+2][j].getType()) ||

(this.map[i][j].getType() == this.map[i+1][j].getType() && this.map[i+1][j].getType() == this.map[i+2][j+1].getType())

) return false;

}

}

return true;

}

public int getScore(){

return score;

}

public Gem[][] getMap(){

return map;

}

}

package com.kislov.java\_game;

public class Gem {

private boolean picked = false;

private boolean selected = false;

private int type; /\* 0 - Destroyed

1,2,3,4,5 - Color \*/

/\*public Gem(int type){

this.type = type;

}\*/

public Gem(){

type = (int) (Math.random() \* 5) + 1;

}

public void destroy(){

this.type = 0;

}

public int getType(){

return type;

}

public boolean isSelected(){

return selected;

}

public void setPicked(){

picked = true;

}

public void setUnpicked(){

picked = false;

}

public void setSelected(){

selected = true;

}

public void setUnselected(){

selected = false;

}

public boolean isPicked(){

return picked;

}

}

C#:

namespace Rockgems

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Rockgems

{

public partial class Form1 : Form

{

private Gamefield gamefield;

private static bool allowActions = true;

public Form1()

{

InitializeComponent();

bRestart.Visible = false;

gamefield = new Gamefield();

gamefield.PrepareField();

UpdateScore();

panel1.Visible = true;

panel2.Visible = false;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)//rules

{

Правила form2 = new Правила();

form2.Show();

}

private void button3\_Click(object sender, EventArgs e)//exit1

{

this.Dispose();

Application.Exit();

}

private void button1\_Click(object sender, EventArgs e)//play

{

panel1.Visible = false;

panel2.Visible = true;

}

private void button4\_Click(object sender, EventArgs e)//exit

{

this.Dispose();

Application.Exit();

}

private void button11\_Click(object sender, EventArgs e)//restart

{

gamefield = new Gamefield();

gamefield.PrepareField();

UpdateScore();

panel1.Visible = false;

panel2.Visible = true;

Refresh();

}

private void button10\_Click(object sender, EventArgs e)//up

{

if (allowActions) gamefield.MovePickerUp();

UpdateScore();

RepaintPanel();

}

private void button8\_Click(object sender, EventArgs e)//down

{

if (allowActions) gamefield.MovePickerDown();

UpdateScore();

RepaintPanel();

}

private void button7\_Click(object sender, EventArgs e)//left

{

if (allowActions) gamefield.MovePickerLeft();

UpdateScore();

RepaintPanel();

}

private void button9\_Click(object sender, EventArgs e)//right

{

if (allowActions) gamefield.MovePickerRight();

UpdateScore();

RepaintPanel();

}

private void button6\_Click(object sender, EventArgs e)//pick

{

if (allowActions) gamefield.Pick();

RepaintPanel();

}

private void button5\_Click(object sender, EventArgs e)//pause

{

bRestart.Visible = true;

panel2.Visible = false;

panel1.Visible = true;

}

private void RepaintPanel()

{

pGame.Refresh();

}

public void UpdateScore()

{

lScore.Text = "Счёт: " + gamefield.GetScore().ToString();

}

public static void ActivateButtons()

{

allowActions = true;

}

public static void DeactivateButtons()

{

allowActions = false;

}

private void pGame\_Paint(object sender, PaintEventArgs e)

{

Pen pen;

SolidBrush brush;

Gem[,] m = gamefield.GetMap();

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

pen = SetColor(m[i, j]);

brush = SetBrush(m[i, j]);

Rectangle rectangle = new Rectangle(j \* 50 + 2, i \* 50 + 2, 46, 46);

e.Graphics.DrawRectangle(pen, rectangle);

e.Graphics.FillRectangle(brush, rectangle);

if (m[i,j].IsSelected())

{

e.Graphics.DrawRectangle(new Pen(Brushes.Black), new Rectangle(j \* 50 + 1, i \* 50 + 1, 48, 48) );

}

if (m[i,j].IsPicked())

{

e.Graphics.DrawRectangle(new Pen(Brushes.Black), new Rectangle(j\* 50 + 4, i \* 50 + 4, 42, 42));

e.Graphics.DrawRectangle(new Pen(Brushes.White), new Rectangle(j\* 50 + 5, i \* 50 + 5, 40, 40));

e.Graphics.DrawRectangle(new Pen(Brushes.Black), new Rectangle(j\* 50 + 6, i \* 50 + 6, 38, 38));

}

}

}

}

private Pen SetColor(Gem gem)

{

Pen pen = new Pen(Brushes.Black);

if (gem.GetGemType() == 1) pen = new Pen(Brushes.Yellow);

else if (gem.GetGemType() == 2) pen = new Pen(Brushes.Red);

else if (gem.GetGemType() == 3) pen = new Pen(Brushes.Green);

else if (gem.GetGemType() == 4) pen = new Pen(Brushes.Blue);

else if (gem.GetGemType() == 5) pen = new Pen(Brushes.Magenta);

return pen;

}

private SolidBrush SetBrush(Gem gem)

{

SolidBrush brush = new SolidBrush(Color.Black);

if (gem.GetGemType() == 1) brush = new SolidBrush(Color.Yellow);

else if (gem.GetGemType() == 2) brush = new SolidBrush(Color.Red);

else if (gem.GetGemType() == 3) brush = new SolidBrush(Color.Green);

else if (gem.GetGemType() == 4) brush = new SolidBrush(Color.Blue);

else if (gem.GetGemType() == 5) brush = new SolidBrush(Color.Magenta);

return brush;

}

}

}

namespace Rockgems

{

public partial class Правила : Form

{

public Правила()

{

InitializeComponent();

}

}

}

using System;

namespace Rockgems

{

public class Gamefield

{

private int size;

private Gem[,] map;

private int pickX;

private int pickY;

private int score;

private int[] pair;

private Random random = new Random();

public Gamefield()

{

size = 8;

map = new Gem[8,8];

Generate();

pickX = 0;

pickY = 0;

score = 0;

pair = new int[] { -1, -1, -1, -1 };

SetGemAsPicked();

}

public void MovePickerUp()

{

if (pickY > 0)

{

SetGemAsUnpicked();

pickY = pickY - 1;

SetGemAsPicked();

DoSecondPick();

}

}

public void MovePickerDown()

{

if (pickY < size - 1)

{

SetGemAsUnpicked();

pickY = pickY + 1;

SetGemAsPicked();

DoSecondPick();

}

}

public void MovePickerLeft()

{

if (pickX > 0)

{

SetGemAsUnpicked();

pickX = pickX - 1;

SetGemAsPicked();

DoSecondPick();

}

}

public void MovePickerRight()

{

if (pickX < size - 1)

{

SetGemAsUnpicked();

pickX = pickX + 1;

SetGemAsPicked();

DoSecondPick();

}

}

public void Pick()

{

if (pair[0] == -1)

{

pair[0] = pickY;

pair[1] = pickX;

SetGemAsSelected();

}

else

{

pair[0] = pair[1] = pair[2] = pair[3] = -1;

SetGemAsUnselected();

}

}

private void SetGemAsPicked()

{

map[pickY,pickX].SetPicked();

}

private void SetGemAsUnpicked()

{

map[pickY,pickX].SetUnpicked();

}

private void SetGemAsSelected()

{

map[pickY,pickX].SetSelected();

}

private void SetGemAsUnselected()

{

map[pickY,pickX].SetUnselected();

}

private void Generate()

{

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

map[i, j] = new Gem(random);

}

}

}

private void DoSecondPick()

{

if (pair[0] != -1)

{

StartCollectingOperation();

pair[2] = pickY;

pair[3] = pickX;

Swap();

FindRemovable(false);

}

}

public void StartCollectingOperation()

{

Form1.DeactivateButtons();

}

public void EndCollectingOperation()

{

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

if (map[i,j].IsSelected())

{

map[i,j].SetUnselected();

}

if (map[i,j].IsPicked())

{

map[i,j].SetUnpicked();

}

}

}

SetGemAsPicked();

pair[0] = pair[1] = pair[2] = pair[3] = -1;

Form1.ActivateButtons();

}

private void Swap()

{

Gem temp;

temp = map[pair[0],pair[1]];

map[pair[0],pair[1]] = map[pair[2],pair[3]];

map[pair[2],pair[3]] = temp;

}

private void FindRemovable(bool recursion)

{

bool removableFound = false;

/\* rows \*/

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size - 2; j++)

{

if (this.map[i,j].GetGemType() == this.map[i,j + 1].GetGemType() && this.map[i,j + 1].GetGemType() == this.map[i,j + 2].GetGemType())

{

this.map[i,j].Destroy();

this.map[i,j + 1].Destroy();

this.map[i,j + 2].Destroy();

AddScore();

removableFound = true;

}

}

}

/\* columns \*/

for (int i = 0; i < size - 2; i++)

{

for (int j = 0; j < size; j++)

{

if (this.map[i,j].GetGemType() == this.map[i + 1,j].GetGemType() && this.map[i + 1,j].GetGemType() == this.map[i + 2,j].GetGemType())

{

this.map[i,j].Destroy();

this.map[i + 1,j].Destroy();

this.map[i + 2,j].Destroy();

AddScore();

removableFound = true;

}

}

}

if (!removableFound && !recursion)

{

if (pair[0] != -1) Swap();

EndCollectingOperation();

}

else if (!removableFound && recursion) EndCollectingOperation();

else CollectGems();

}

public void PrepareField()

{

StartCollectingOperation();

while (CheckIfNoMoreMoves()) Generate();

FindRemovable(false);

score = 0;

}

private void CollectGems()

{

Popup();

Dropdown();

while (CheckIfNoMoreMoves()) Generate();

FindRemovable(true);

}

private void Popup()

{

Gem temp;

for (int k = 0; k < size; k++)

{

for (int i = 1; i < size; i++)

{

for (int j = 0; j < size; j++)

{

if (map[i,j].GetGemType() == 0)

{

temp = map[i,j];

map[i,j] = map[i - 1,j];

map[i - 1,j] = temp;

}

}

}

}

}

private void Dropdown()

{

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

if (map[i, j].GetGemType() == 0)

{

map[i, j] = new Gem(random);

}

}

}

}

private void AddScore()

{

score += 100;

}

private bool CheckIfNoMoreMoves()

{

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - 2; j++)

{

if (

(this.map[i,j].GetGemType() == this.map[i,j + 1].GetGemType() && this.map[i,j + 1].GetGemType() == this.map[i + 1,j + 2].GetGemType()) ||

(this.map[i + 1,j].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i,j + 2].GetGemType()) ||

(this.map[i + 1,j].GetGemType() == this.map[i,j + 1].GetGemType() && this.map[i,j + 1].GetGemType() == this.map[i,j + 2].GetGemType()) ||

(this.map[i,j].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i + 1,j + 2].GetGemType()) ||

(this.map[i,j].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i,j + 2].GetGemType()) ||

(this.map[i + 1,j].GetGemType() == this.map[i,j + 1].GetGemType() && this.map[i,j + 1].GetGemType() == this.map[i + 1,j + 2].GetGemType())

) return false;

}

}

for (int i = 0; i < size - 2; i++)

{

for (int j = 0; j < size - 1; j++)

{

if (

(this.map[i,j + 1].GetGemType() == this.map[i + 1,j].GetGemType() && this.map[i + 1,j].GetGemType() == this.map[i + 2,j + 1].GetGemType()) ||

(this.map[i,j + 1].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i + 2,j].GetGemType()) ||

(this.map[i,j + 1].GetGemType() == this.map[i + 1,j].GetGemType() && this.map[i + 1,j].GetGemType() == this.map[i + 2,j].GetGemType()) ||

(this.map[i,j].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i + 2,j + 1].GetGemType()) ||

(this.map[i,j + 1].GetGemType() == this.map[i + 1,j + 1].GetGemType() && this.map[i + 1,j + 1].GetGemType() == this.map[i + 2,j].GetGemType()) ||

(this.map[i,j].GetGemType() == this.map[i + 1,j].GetGemType() && this.map[i + 1,j].GetGemType() == this.map[i + 2,j + 1].GetGemType())

) return false;

}

}

return true;

}

public int GetScore()

{

return score;

}

public Gem[,] GetMap()

{

return map;

}

}

}

using System;

namespace Rockgems

{

public class Gem

{

private bool picked = false;

private bool selected = false;

private int type; /\* 0 - Destroyed

1,2,3,4,5 - Color \*/

public Gem(Random random)

{

type = random.Next(1, 6);

}

public void Destroy()

{

this.type = 0;

}

public int GetGemType()

{

return type;

}

public bool IsSelected()

{

return selected;

}

public void SetPicked()

{

picked = true;

}

public void SetUnpicked()

{

picked = false;

}

public void SetSelected()

{

selected = true;

}

public void SetUnselected()

{

selected = false;

}

public bool IsPicked()

{

return picked;

}

}

}

**Заключение**

В процессе выполнения курсовой работы было разработано приложение-игра «Спасти Самурая». На этом примере было произведено сравнение двух языков программирования Java и C#. Хочется отметить, что на базовом уровне у них мало отличий. Схожий синтаксис, схожая концепция компиляции и работы программ. Главным отличаем на примере оконного приложения становится разная обработка событий. В Java для этого существуют слушатели событий, а в C# используется своя система событий.

Так же, в процессе работы был усвоен ряд новых знаний и выполнены следующие пункты:

* закреплены все основные элементы программирования на двух объектно-ориентированных языках программирования;
* закреплены навыки представления информации в программах на языках высокого уровня;
* продемонстрированы приобретённые навыки и умения;
* приобретены профессиональные компетенции.

Все поставленные цели и задачи работы выполнены.

**Список использованных источников**

1. Учебная практика. Руководство и методические указания для студентов направления 09.03.02 «Информационные системы и технологии», профили подготовки «Информационные системы и технологии в административном управлении», «Информационные системы и технологии в промышленности/машиностроении». Составители: Аршинский В.Л., Серышева И.А – Иркутск, Изд-во ИРНИТУ, 2016, 49с.
2. docs.oracle.com [официальный сайт Oracle help] URL: <https://docs.oracle.com/en/> (дата последнего обращения: 10.12.2019)
3. docs.microsoft.com [официальный сайт Microsoft Docs] URL: [https://docs.microsoft.com/](https://docs.microsoft.com/en/) (дата последнего обращения: 14.12.2019)