Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Курсовая работа**

**Дисциплина**: Разработка мобильных приложений

**Тема**: Игра «Lines»

Выполнил студенты гр. 33531/2 П.В.Никонов

(подпись)

Преподаватель Кузнецов А.Н.

(подпись)

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Оглавление

[1. Введение 3](#_Toc533729571)

[2. Описание игры 3](#_Toc533729572)

[3. Реализация 3](#_Toc533729573)

[4. Тесты 5](#_Toc533729574)

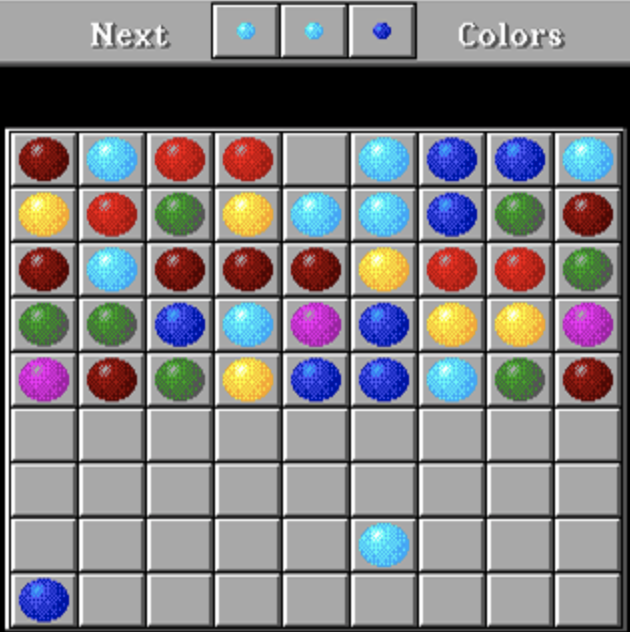
[5. Выводы 8](#_Toc533729575)

1. **Введение**

В данном курсовом проекте с помощью среды разработки Android Studio реализован проект в формате игры на телефон. Тема проекта – игра *Lines*. Основной язык разработки – *Java.* Тесты написаны с помощью фреймворка *Junit.*

1. **Описание игры**

Игра *Lines* представляет собой поле 9 на 9. В каждой клетке этого поля может находится шар определенного цвета: коричневый, красный, бирюзовый, желтый, зеленый, синий, фиолетовый. Примерный дизайн игры взят из *Lines98* и показан на *рисунке. 2.1.*



*Рис.2.1*

Задача игрока – собрать в ряд (по-вертикали, по-горизонтали, по-диагонали) шары одного цвета. В случае с определенным количеством собранных шаров начисляются очки по следующей схеме:

5 шаров - 5 очка

6 шаров - 12 очков (5 + 1) \* 2

7 шаров - 21 очко (5  + 2) \* 3

8 шаров - 32 очка (5 + 3) \* 4

и. т. д.

Очки записываются в таблицу рекордов и строчке в таблице присваивается имя, которое задаст пользователь.

1. **Реализация**

Приложение состоит из 3 активити:

1. Меню
2. Игровое активити
3. Рекорды

Меню является главным активити, в котором пользователь может выбрать свой путь по приложению: рекорды, игра или на выход. Соответственно это активити содержит три кнопки с листнерами, которые обрабатывают их нажатие.

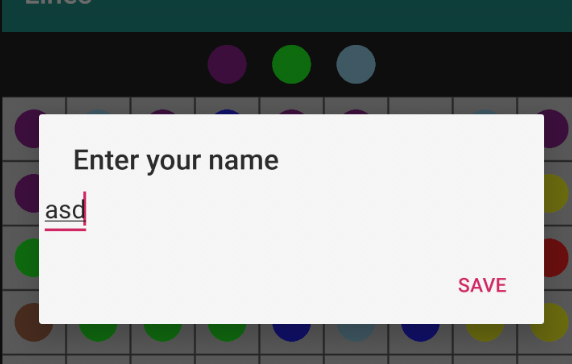
Игровое активити является самым сложным активити. Оно содержит в себе три компонента: *scoreView, gameView и textView*. Пример игрового активити приведен на *рисунке 3.1.*



*Рис. 3.1*

*scoreView* – самое верхнее view, означающее какого цвета будут сгенерированные шары на следующем шаге. В *gameView* пользователь выбирает какой шар куда переместить. Выбор шара поддерживается анимацией. *TextView* соответственно показывает текущее количество очков, которое пользователь набрал в данный момент.

По завершении игры пользователю предлагается ввести свое имя. Данный процесс представлен на *рисунке 3.2.*



*Рис. 3.2*

После чего данные заносятся в базу данных и после этого отображаются в *listView* в рекордах.

Подразумевается, что архитектура приложения выбрана как *MVP*.То есть, я описал интерфейс, который определят правила между взаимодействиями *view* (активити в котором есть *ScoreView* и *GameView*) и *Presenter*. *Presenter* и *Model* (Внутреннее представление игрового поля). То есть теперь связующим звеном между логикой и графикой является класс, реализующий интерфейс *Presenter*. В этом классе обрабатываются касания *View*, если нужно идет обращение к *Model* и этот же класс говорит *view* в какой клетке какого цвета рисовать шар.

Но данный паттерн не совсем хорошо получилось реализовать в связи с тем, что было выбрано другой способ отображения компонентов-мячей на *gameView*. Хорошим бы решением было использование *ViewGroup* и шаров, каждый из которых являлся view, и сам бы умел себя отрисовывать.

В *RecordsActivity* прописано взаимодействие с базой данных*.* В качестве бызы бла использована *SQLite,* был описан интерфейс взаимодействия с данными из базы и реализован *helper*. Затем использовался курсор, помогающий запихнуть все данные из базы в *listView.*

1. **Тесты**

Пример теста на бизнес-логику приведен ниже

|  |
| --- |
| package ru.polytech.course.pashnik.lines;  import org.junit.Test;  import ru.polytech.course.pashnik.lines.Core.Board; import ru.polytech.course.pashnik.lines.Core.Cell; import ru.polytech.course.pashnik.lines.Core.ColorType; import ru.polytech.course.pashnik.lines.Core.Intellect;  import static org.junit.Assert.\*;  public class BusinessLogicTests {   private static final int MIN\_WIN\_NUMBER = 5;   @Test  public void doArithmeticWithCells() {  Cell start = new Cell(2, 1);  final Cell[] direction = {  new Cell(1, 0), // x-axis  new Cell(0, 1), // y-axis  new Cell(-1, 1), // main diagonal  new Cell(1, 1)}; // secondary diagonal   assertEquals(new Cell(3, 1), start.plus(direction[0]));  assertEquals(new Cell(2, 0), start.minus(direction[1]));  assertEquals(new Cell(1, 2), start.plus(direction[2]));  assertEquals(new Cell(1, 0), start.minus(direction[3]));  }   @Test  public void isWinnerCell() {  final Board trueBoard = new Board();  final Board falseBoard = new Board();  final Board secondFalseBoard = new Board();  final Board thirdFalseBoard = new Board();   final Intellect intellect = new Intellect(secondFalseBoard);   // testing one line  for (int i = 0; i < MIN\_WIN\_NUMBER; i++) {  trueBoard.addCell(new Cell(i, i), ColorType.GREEN);  }  assertTrue(trueBoard.isWin(new Cell(4, 4)));  assertTrue(trueBoard.isWin(new Cell(5, 5)));   // testing different colors  for (int i = 0; i < MIN\_WIN\_NUMBER; i++) {  falseBoard.addCell(new Cell(i, i), ColorType.getColorType(i));  }  assertFalse(falseBoard.isWin(new Cell(4, 4)));   // testing five random cells  for (int i = 0; i < MIN\_WIN\_NUMBER; i++) {  secondFalseBoard.addCell(intellect.generateNextCell(),  intellect.generateNextColor());  }  assertFalse(secondFalseBoard.isWin(new Cell(4, 1)));   // testing four this situation: x x x x \_ x  // where x - is a ball, \_ - is a empty cell   for (int i = 0; i < MIN\_WIN\_NUMBER - 1; i++) {  thirdFalseBoard.addCell(new Cell(i, 1), ColorType.BLUE);  }  thirdFalseBoard.addCell(new Cell(5, 1), ColorType.BLUE);  assertFalse(thirdFalseBoard.isWin(new Cell(5, 1)));   } } |

Использовались следующие проверки: арифметические операции над клетками и главная проверка над методом, определяющим победный отрезок из шаров. Все тесты пройдены успешно.

Инструментальные тесты использовались для проверки с взаимодействием с базой данных. Тесты представлены ниже:

|  |
| --- |
| package ru.polytech.course.pashnik.lines;  import android.support.test.InstrumentationRegistry; import android.support.test.runner.AndroidJUnit4;  import org.junit.After; import org.junit.Before; import org.junit.Test; import org.junit.runner.RunWith;  import java.util.List;  import ru.polytech.course.pashnik.lines.DataBase.Contact; import ru.polytech.course.pashnik.lines.DataBase.DataBaseHandler;  import static org.junit.Assert.\*;  @RunWith(AndroidJUnit4.class) public class DataBaseTests {   private final static int TEST\_COUNTER = 3;  private final static int TEST\_DEFAULT\_SCORE = 246;  private final static Contact DEFAULT\_CONTACT = new Contact("Pavel", TEST\_DEFAULT\_SCORE);  private final static Contact DEFAULT\_CONTACT\_2 = new Contact("Alex", TEST\_DEFAULT\_SCORE);   private DataBaseHandler dataBaseHandler;   @Before  public void setUpDataBase() {  dataBaseHandler = new DataBaseHandler(InstrumentationRegistry.getTargetContext());  }   @After  public void finish() {  deleteAll();  dataBaseHandler.close();  }   @Test  public void preConditions() {  assertNotNull(dataBaseHandler);  }   @Test  public void addContact() {  // adding default contact 3 times  for (int i = 0; i < TEST\_COUNTER; i++) {  dataBaseHandler.addContact(DEFAULT\_CONTACT);  }  List<Contact> list = dataBaseHandler.getAllContacts();  assertEquals(TEST\_COUNTER, list.size());   // checking the content of added contact  for (int i = 0; i < TEST\_COUNTER; i++) {  assertEquals(DEFAULT\_CONTACT, list.get(i));  }  }   @Test  public void deleteContact() {  // testing of deleting first Contact from db  dataBaseHandler.addContact(DEFAULT\_CONTACT\_2);  dataBaseHandler.addContact(DEFAULT\_CONTACT);   dataBaseHandler.deleteContactById(1);  List<Contact> list = dataBaseHandler.getAllContacts();   assertEquals(1, list.size());  assertEquals(DEFAULT\_CONTACT, list.get(0));  }   @Test  public void deleteAll() {  // testing of deleting all contacts from db  dataBaseHandler.addContact(DEFAULT\_CONTACT);  dataBaseHandler.addContact(DEFAULT\_CONTACT);  dataBaseHandler.deleteAll();  assertTrue(dataBaseHandler.getAllContacts().isEmpty());  }  } |

Все тесты проверяют методы, которые описывает интерфейс *Idbhelper:* добавление, удаление, инициализация и поиск данных в базе.

UI тесты проверяют взаимодействие с пользователем кнопок. Данные тесты приведены ниже:

|  |
| --- |
| package ru.polytech.course.pashnik.lines;   import android.support.test.filters.LargeTest; import android.support.test.rule.ActivityTestRule; import android.support.test.runner.AndroidJUnit4;  import org.junit.Rule; import org.junit.Test; import org.junit.runner.RunWith;  import ru.polytech.course.pashnik.lines.Activities.MainActivity;  import static android.support.test.espresso.Espresso.onView; import static android.support.test.espresso.assertion.ViewAssertions.matches; import static android.support.test.espresso.matcher.ViewMatchers.isDisplayed; import static android.support.test.espresso.matcher.ViewMatchers.withId;  @RunWith(AndroidJUnit4.class) @LargeTest public class UITests {   @Rule  public ActivityTestRule<MainActivity> activityActivityTestRule  = new ActivityTestRule<>(MainActivity.class);   @Test  public void clicks() {  onView(withId(R.id.exit))  .check(matches(isDisplayed()));  onView(withId(R.id.recordds))  .check(matches(isDisplayed()));  onView(withId(R.id.startBtn))  .check(matches(isDisplayed()));  }  } |

1. **Выводы**

В результате работы над проектом были получены базовые знания использования фреймворков ОС Andoid. Было частично реализовано приложение Lines и написаны Unit, UI и Instrumental тесты.