Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

09.03.04 – Программная инженерия

Профиль направления подготовки бакалавриата

Системное и прикладное программное обеспечение

Отчет по учебному курсу 'Верификация программного обеспечения'

Выполнил:
студент 4 курса группы 22407

Н.Н. Анисимов

Петрозаводск — 2023

Содержание

1 Объект тестирования **3**

2 Высокоуровненная архитектура **4**

3 Интерфейс **5**

4 Стратегия тестирования **6**

4.1 Отношение к тестированию 6

4.2 Стратегия блочного тестирования 7

4.3 Стратегия интеграционного тестирования 7

4.4 Стратегия аттестационного тестирования 7

4.5 Стратегия нагрузочного тестирования 7

5 Детальный план тестирования **8**

5.1 Блочное тестирование 8

5.2 Интеграционное тестирование 9

5.3 Аттестационные тесты 10

5.4 Нагрузочные тесты 11

6 Примеры тестов **12**

6.1 Б3 12

6.2 Н1 12

6.3 И1 13

7 Журнал тестирования **14**

7.1 Журнал блочного тестирования 14

7.2 Журнал аттестационного тестирования 15

7.3 Журнал интеграционного тестирования 15

7.4 Журнал нагрузочного тестирования 16

8 Журнал ошибок **17**

9 Результаты **17**

1 Объект тестирования

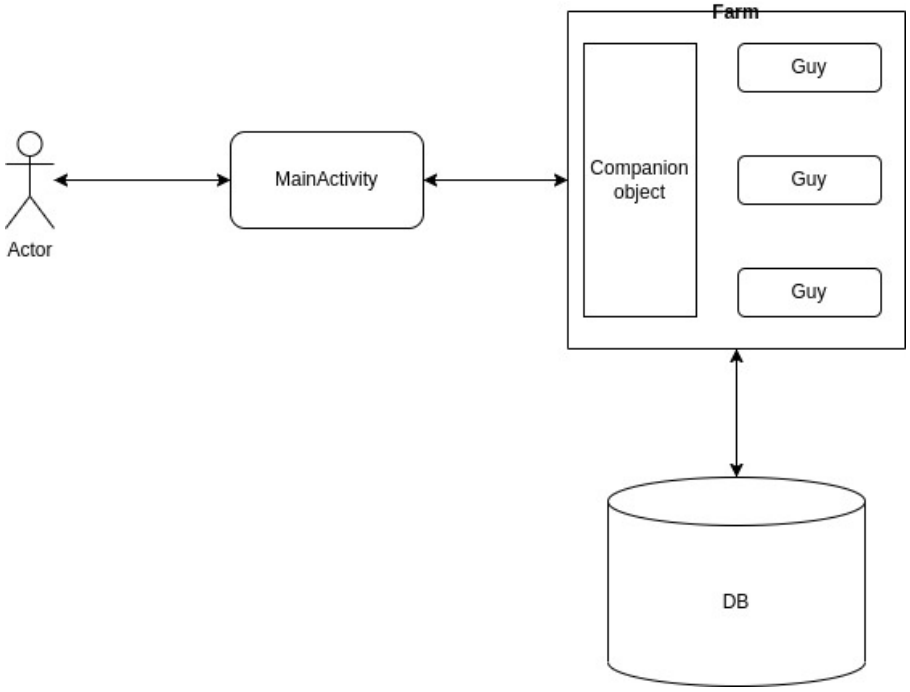
Тестируемое приложение разработано под Android на Kotlin и является игрой-фермой. В ней игроку предстоит покупать игровые объекты, приносящие деньги, а так же поддерживать их жизнедеятельность периодическими заходами в игру. У игрока есть следующие возможности:

- покупать игровой объект, для того, чтобы он приносил доход;
- кормить игровой объект, для того, чтобы он не умирал;
- включать/выключать музыку;
- начинать новую игру.

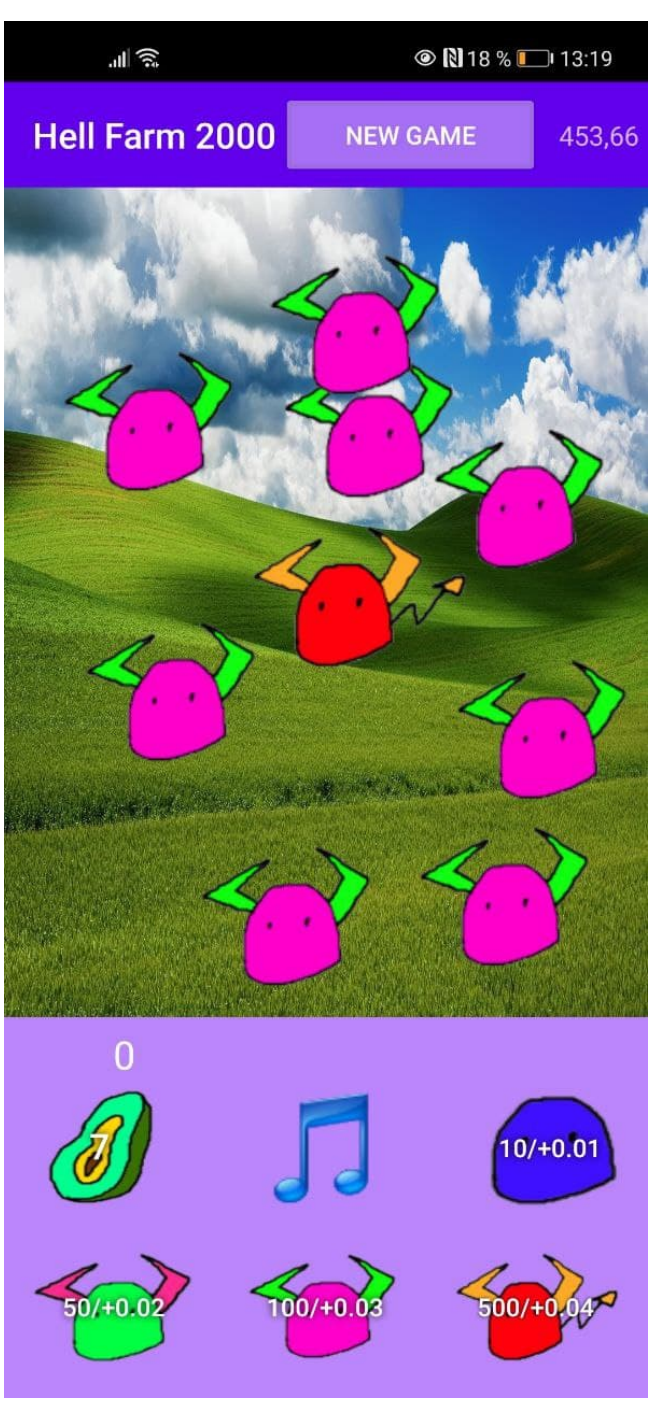
После покупки / кормежки игровой объект начинает хотеть есть через X минут, если его не кормить в течении Y минут, то он исчезает.

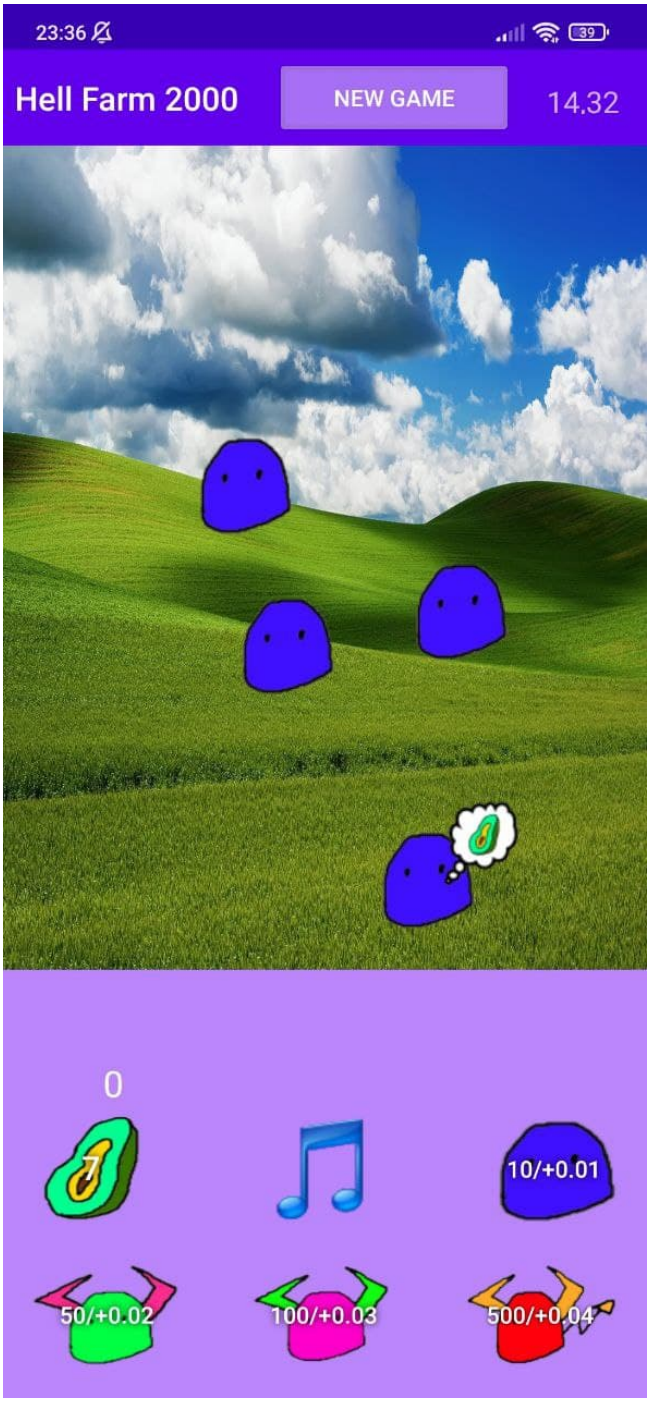
2 Высокоуровненная архитектура

Класс MainActivity реагирует на события, которые провоцирует пользователь и вызывает функции класса Farm. Этот класс содержит весь механизм игры, взаимодействует с базой данных, для сохранения информации перед выходом пользователя из игры. Так же он содержит и управляет набором элементов класса Guy, который и является игровым объектом.



3 Интерфейс


18 % 13:19
Hell Farm 2000
NEW GAME
453,66
0
7
10/+0.01
50/+0.02
100/+0.03
500/+0.04


23:36
Hell Farm 2000
NEW GAME
14,32
0
10/+0.01
50/+0.02
100/+0.03
500/+0.04

4 Стратегия тестирования

4.1 Отношение к тестированию

- MainActivity - не подлежит блочному тестированию, так как содержит перегруженные методы встроенного класса и музыкальный плеер, который тестироваться не будет. MainActivity необходим для проведения аттестационного и интеграционного тестирования;
  - onCreate(savedInstanceState: Bundle?) - вызывается при начале работы программы, вызывает функцию start у companion object Farm;
  - onPause() - вызывается при выходе из программы, останавливается музыка, меняется флаг для остановки игрового потока в Farm;
  - onResume() - вызывается при входе в программу, запускает игровой поток в Farm, включает музыку;
  - play(view: View) - метода для остановки/включения музыкального плеера;
  - createMaleGuy(view: View), createMaleGuy(view: View), createDefaultGuy(view: View), createPredatorGuy(view: View) - функции для создания игровых объектов, вызов addGuy в Farm;
  - newGame(view: View) - начало новой игры, вызов newGame в Farm;
  - buyFood(view: View) - покупка еды для игровых объектов.
- Farm - подлежит блочному тестированию и интеграционному тестированию связи с классом Guy и Main Activity;
  - addGuy(name: String, context: Context, cost: Int) - добавить игровой объект типа name и цены cost на поле, вызов конструктора Guy;
  - go(context: Context) - функция, которая запускается в игровом потоке для внесения изменений в запас денег игрока, обновления статуса игровых объектов(голоден/не сохранения игры и удаления объектов, вызывает функции delete и showCloud у Guy и функцию save;
  - newGame(context: Context) - начинает новую игру;
  - start(context: Context) - запускает вспомогательный поток для обновления отображения денеги количества купленной еды;

  - save(context: Context) - сохранение игры;
  - load(context: Context) - загрузка игры, вызов delete у guy для очистки поля.

- Guy - подлежит блочному тестированию и интеграционному стестированию связи с классом Farm и между объектами этого класса.
  - Guy(typeTo: String, context: Context, x: Int = -1, y:Int = -1) - конструктор, создающий игровой объект. При размещении на поле, должен быть в отдалении от других объектов;
  - showCloud(context: Context) - отображение облачка, когда объект голоден;
  - removeCloud(context: Context) - убрать облачко;
  - delete(context: Context) - удаляет связанные с объектом отображения на поле при его удалении.

4.2 Стратегия блочного тестирования

Тестирование будет проводиться с помощью платформы JUnit 5 для Kotlin. В блочных тестах тестируются отдельные функции, вне контекста программы. Тестированию подлежат все функции объектов Farm.

4.3 Стратегия интеграционного тестирования

Тестирование будет проводиться с помощью JUnit 5 для Kotlin. В интеграционных тестах тестируется взаимодействие программных модулей. Будут тестироваться взаимодействие Farm и Guy, а именно функции addGuy, go в модуле Farm и конструктор Guy.

4.4 Стратегия аттестационного тестирования

Этот вид тестирования предпологает запуск программы и ручной поиск ошибок, путем выполнения заданных сценариев.

4.5 Стратегия нагрузочного тестирования

Этот вид тестирование проверяет корректность работы программы под нагрузкой. Тестироваться будет быстрый вход и выход из программы с проверкой корректности работы всех потоков.

5 Детальный план тестирования

5.1 Блочное тестирование

ID	Б1
Описание теста	Начало игры
Тип теста	позитивный
Объект тестирования	Класс Farm, функция start
Входные данные	money = 500, food = 600
Ожидаемый результат	mny.text = 500, fdd.text = 600

ID	Б2
Описание теста	Новая игра
Тип теста	позитивный
Объект тестирования	Класс Farm, функция newGame
Входные данные	нет
Ожидаемый результат	mny.text = 0, fdd.text = 0

ID	Б3
Описание теста	Сохранение и загрузка
Тип теста	позитивный
Объект тестирования	Класс Farm, функция load и save
Входные данные	money = 500, food = 600
Ожидаемый результат	money = 500, food = 600

ID	Б4
Описание теста	Добавить персонажа
Тип теста	позитивный
Объект тестирования	Класс Farm, функция addGuy
Входные данные	name = ”maleGuy”, cost = 10
Ожидаемый результат	guys.size() == 1

ID	Б5
Описание теста	Добавить 11 персонажа
Тип теста	негативный
Объект тестирования	Класс Farm, функция addGuy
Входные данные	name = ”maleGuy”, cost = 10, guys.size() == 10
Ожидаемый результат	guys.size() == 10

5.2 Интеграционное тестирование

ID	И1
Описание теста	Добавление игрового объекта
Тип теста	позитивный
Объект тестирования	Функция Farm.addGuy и конструктор Guy
Действия	money = 10 Farm.addGuy(”DefaultGuy”, context, 10) Из функции вызывается конструктор Guy, которые добавляет игровой объект.
Ожидаемый результат	Farm.guys[0].button.x и Farm.guys[0].button.y должны быть определенны

ID	И2
Описание теста	Голодание
Тип теста	позитивный
Объект тестирования	Функция Farm.addGuy, Farm.go и конструктор Guy
Действия	addGuy(”MaleGuy”, 10) Farm.go()
Ожидаемый результат	Farm.guys[0].hungry > 0

ID	И3
Описание теста	Остановка голодания
Тип теста	негативный
Объект тестирования	Функция Farm.addGuy, Farm.go и конструктор Guy
Действия	addGuy(”MaleGuy”, 10) Farm.go() isStopped = 1
Ожидаемый результат	Farm.guys[0].hungry не меняется после проведения действий

5.3 Аттестационные тесты

ID	A1
Описание теста	Проверка музыкального плеера
Тип теста	позитивный
Объект тестирования	Модуль MainActivity, функции play(), onPause(), onResume()
Действия	Зайти в приложение
Ожидаемый результат	Играет музыка

ID	A2
Описание теста	Покупка игрового объекта при отсутствии денег
Тип теста	негативный
Объект тестирования	Модуль Farm функция addGuy, Модуль MainActivity, функция addMaleGuy
Действия	Нажать на игровой объект в магазине при отсутствии денег
Ожидаемый результат	Деньги не тратятся, объект не появляется

ID	A3
Описание теста	Отсутствие дохода при закрытом приложении
Тип теста	негативный
Объект тестирования	Модуль Farm функция go, MainActivity функция onPause
Действия	Купить игровой объект Выйти из приложения Зайти снова
Ожидаемый результат	Количество денег не изменилось после выхода

ID	A4
Описание теста	Начало новой игры
Тип теста	позитивный
Объект тестирования	Модуль Farm функция newGame
Действия	Купить игровой объект Нажать на кнопку ”Новая игра”
Ожидаемый результат	Игровой объект пропадает, денег становится 20

ID	A5
Описание теста	Начало новой игры
Тип теста	позитивный
Объект тестирования	Модуль Guy функция onClickListener()
Действия	Купить игровой объект Подождать пока не появится облако над игровым объектом Купить еду Нажать на игровой объект
Ожидаемый результат	Облако над игровым объектом пропадает

5.4 Нагрузочные тесты

ID	H1
Описание теста	Проверка правильного подсчета денег и статуса процессов при большом количестве включений и выключений программы
Тип теста	общий
Объект тестирования	Все модули
Действия	Добавляется игровой объект DefaultGuy. Последовательно 1000 раз запускается функции onPause() и onResume()
Ожидаемый результат	isRunning = true, money = 0.01

6 Примеры тестов

6.1 Б3

```
@Test
void testB3(context: Context)
{
   Farm.money = 500
   Farm.food = 600
   Farm.save(context)
   Farm.load(context)
   Assertions.assertEquals(Farm.money, 500);
   Assertions.assertEquals(Farm.food, 600);
}
```

6.2 H1

```
@Test
void testH1(context: Context)
{
   createDefaultGuy(context)
   for(i in 0 <= until < 1000)
   {
```

```
        onPause(context)
        onResume(context)
    }
    Assertions.assertEquals(Farm.money, 0.01);
    Assertions.assertEquals(Farm.isRunning, True);
}
```

6.3 И1

```
@Test
void testI1(context: Context)
{
    money = 10
    Farm.addGuy(''DefaultGuy'', context, 10)
    Assertions.assertNotSame(Farm.guys[0].button.x, -1);
    Assertions.assertNotSame(Farm.guys[0].button.y, -1);
}
```

7 Журнал тестирования

7.1 Журнал блочного тестирования

ID	Дата	Результат	Отчёт
Б1	30.11.23	Пройден	
Б2	30.11.23	Пройден	
Б3	30.11.23	Пройден	
Б4	30.11.23	Пройден	
Б5	30.11.23	Не пройден	Ошибка в функции addGuy

7.2 Журнал аттестационного тестирования

ID	Дата	Результат	Отчёт
А1	30.11.23	Пройден	
А2	30.11.23	Пройден	
А3	30.11.23	Пройден	
А4	25.12.23	Пройден	
А5	25.12.23	Пройден	

7.3 Журнал интеграционного тестирования

ID	Дата	Результат	Отчёт
И1	30.11.23	Пройден	
И2	30.11.23	Пройден	
И3	30.11.23	Пройден	

7.4 Журнал нагрузочного тестирования

ID	Дата	Результат	Отчёт
H1	25.12.23	Не пройден	Ошибка в функции start

8 Журнал ошибок

1. Место возникновения ошибки: Функция добавления игрового объект Farm.addGuy(), тест Б5.

   Ошибка: guys.size() == 11

   Причина ошибки: Отсутствие проверки в addGuy()

   Способ устранения: Добавление проверки перед добавлением объекта, что guys.size() != 10

   Статус: исправлена

2. Функции onResume() и onPause(), тест Н1.

   Ошибка: isRunning == False Причина ошибки: Поток в функции start не успевает завершиться, когда уже вызываются следующие функции

   Статус: не исправлена

9 Результаты

Были проведены блочные, аттестационные и интеграционные тестирования системы, а также тестирование системы на нагрузку.

Проведённое тестирование показало, что код программы нуждается в отладке и доработке. Не все выявленные в ходе тестирования ошибки были устранены. Покрытие тестами составляет 60%, следовательно, программа нуждается в дополнительном покрытии тестами.