**Министерство образования и науки Российской Федерации**

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

Оценка проекта

Члены комиссии

**РАЗРАБОТКА ПРОГРАММЫ НА JAVAFX «БЫКИ И КОРОВЫ»**

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Пояснительная записка

09.03.04 58.29.29 006 ПЗ

|  |  |
| --- | --- |
| Руководитель | В.И. Суханов |
| Студент гр. РИВ-270027у (662) | Е.Л. Бусыгин |

Екатеринбург 2018

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](#_Toc525045275) 3

[1 Постановка задачи 4](#_Toc525045276)

[2 Анализ поставленной задачи 5](#_Toc525045277)

[2.1 Задача и взаимодействие 5](#_Toc525045278)

[2.2 Структуры данных](#_Toc525045290) 5

[2.3 Схема работы системы](#_Toc525045291) 6

[3 Описание результатов разработки](#_Toc525045292) 7

[3.1 Основной класс](#_Toc525045293) 7

[3.2 Дополнительные классы](#_Toc525045294) 12

[3.3 Дополнительные](#_Toc525045295) ресурсы14

[4 Руководство пользователя](#_Toc525045296) 15

[4.1 Минимальные системные требования](#_Toc525045297) 15

[4.2 Установка и удаление программы](#_Toc525045298) 15

[4.3 Работа с программой](#_Toc525045299) 15

[4.4 Результат работы программы](#_Toc525045300) 15

[ЗАКЛЮЧЕНИЕ](#_Toc525045301) 19

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc525045302) 20

[ПРИЛОЖЕНИЕ А (обязательное) Исходный текст](#_Toc525045303) 21

[А.1 Основной класс](#_Toc525045304) 21

[А.2 Дополнительные классы](#_Toc525045305) 34

# ВВЕДЕНИЕ

Цель курсовой работы – используя графический фреймворк, разработать программу, которая осуществляет поиск, загаданной пользователем, последовательности цветов. Для разработки программы будем использовать язык программирования Java, а в частности платформу на основе java для создания приложений с графическим интерфейсом javaFX. При разработке программы будем руководствоваться описанием задачи в книге Окулова С.М. Программирование в алгоритмах. Требуется выбрать структуру данных, которая в последствии будет заполнена всеми возможными последовательностями вариантов и с использованием метода решета, отсеивать последовательности неудовлетворяющие условиям. Неудовлетворяющие последовательности должны быть получены из ответов пользователя на предложенные компьютером варианты последовательностей. Для нахождения загаданной последовательности за минимальное количество ходов необходимо, чтобы компьютер грамотно находил оптимальные предложенные пользователю последовательности [3]. Интерфейс программы должен позволять контактировать пользователю с программой, через обработчики нажатий на кнопки с помощью мыши или комбинаций клавиш на клавиатуре, а также предоставлять информацию пользователю в виде изменяющихся сцен или всплывающих окон. При разработки графической части программы будем руководствоваться учебным пособием [4].

Правила оформления пояснительной записки соответствуют приведенным в пособии [2].

Пояснительная записка имеет следующую структуру:

а) введение, где описывается краткая постановка задачи и содержание разделов;

б) основная часть, которая состоит из нескольких разделов:

1) постановка задачи;

2) анализ поставленной задачи;

3) описание результатов разработки;

4) руководство пользователя.

в) заключение, где проводится анализ проделанной работы;

г) список использованных источников, который содержит перечень использованной при разработке литературы;

д) приложение, где содержатся исходные коды.

## 

## 1 Постановка задачи

Для решения задачи в программе «Быки и коровы» воспользуемся методом решета. Метод комбинаторного программирования, который рассматривает конечное множество и исключает все элементы этого множества, не представляющие интереса. Мы будем генерировать полное множество и пойдем от обратного – от всего множества решений к единственному решению задачи, путем отсеивания. Для наилучшего результата, необходимо правильно выбрать первые вариации последовательностей, предложенные компьютером. В этом нам поможет специальный метод, который будет генерировать «правильные» первые ходы, наличие которых позволит нам найти загаданное число не более чем за 6 ходов [3].

При разработке не обязательно использование JUnit-тестов.

Для создания графического интерфейса прежде всего нам нужны различные визуальные элементы, например, кнопки, текстовые поля, списки, изображения и так далее. JavaFX предоставляет большой набор встроенных визуальных элементов. Далее нам необходимы компоновщики, которые позволяют управлять расположением вложенных в них элементов, для придания графическому интерфейсу упорядоченности, создавая дизайн окна. Для приданию интерфейсу динамичности будем использовать обработчики событий, которые будут, например, реагировать на нажатие по кнопке и изменять интерфейс окна пользователя. В целом интерфейс программы будет представлять из себя оконные сцены, которые будут сменять друг друга в зависимости от действий пользователя [4].

При описании программы использовать схемы работы системы (ГОСТ 19.701-90) и диаграммы классов по правилам UML.

## 2 Анализ поставленной задачи

#### 2.1 Задача и взаимодействие

Для решения поставленной задачи необходимо в разработанной программе выполнить следующий алгоритм (последовательность действий):

а) сгенерировать все возможные последовательности комбинаций в выбранной структуре данных;

б) сгенерировать последовательность для вопроса пользователю, с помощью специального метода преобразовать её в последовательность цветов и показать пользователю;

в) создать окно графического интерфейса для общения с пользователем и дать пользователю возможность ответить, его ли это последовательность или нет;

г) если пользователь ответит «ДА», то переходим к пункту 'ж', если «НЕТ», то к пункту 'д';

д) дать пользователю возможность выбрать сколько в предложенной последовательности «БЫКОВ» и «КОРОВ»;

е) отсортировать от всех возможных последовательностей те что не подходят;

ё) если в массиве всех последовательностей осталось одна последовательность, то переходим к пункту 'ж', если больше одной, то к пункту 'б';

ж) выводим найденную последовательность пользователя на экран, а также даем возможность начать заново или закрыть программу с помощью кнопок.

Также нужно реализовать дополнительные функции для улучшения интерфейса. Реализуем, на старте программы, возможность выбрать секретную последовательность цветов, чтобы в последствии она отображалась в окне пользователя, для более легкого сравнения его с предложенными компьютером последовательностями. Там же нужно реализовать кнопку с возможностью вызвать всплывающее окно с правилами игры. Ещё, правила игры, функцию начать заново и выход из программы, реализуем в контекстном меню. А также вызов этих трех функций реализуем с помощью комбинаций клавиш. Реализуем лог ходов, с подсветкой ошибок, чтобы пользователь мог наглядно увидеть где он просчитался.

#### 2.2 Структуры данных

Как мы выяснили при реализации программы нам требуется хранить данные в виде массива, чтобы в последствии преобразовывать его, для получения конечного результата, в виде единственного значения. Для хранения данных нам вполне подошли бы простые массивы, но для большего удобства преобразования нужно использовать более сложную структуру. Поэтому был выбран Collections Framework. В частности, такая реализация интерфейса List как ArrayList. Структура данных максимально приближена к обычному массиву и содержит готовые методы add() и remove(), которые помогут нам при работе с коллекцией наших значений [1].

#### 2.3 Схема работы системы

Опишем программу в виде схемы работы системы по ГОСТ 19.701-90 (см. рисунок 1).



Рисунок 1 – Схема работы системы

## 3 Описание результатов разработки

#### 3.1 Основной класс

Полный текст исходного кода находиться в приложении. На рисунке 2 приведена полная диаграмма классов на которой изображены все основные классы.

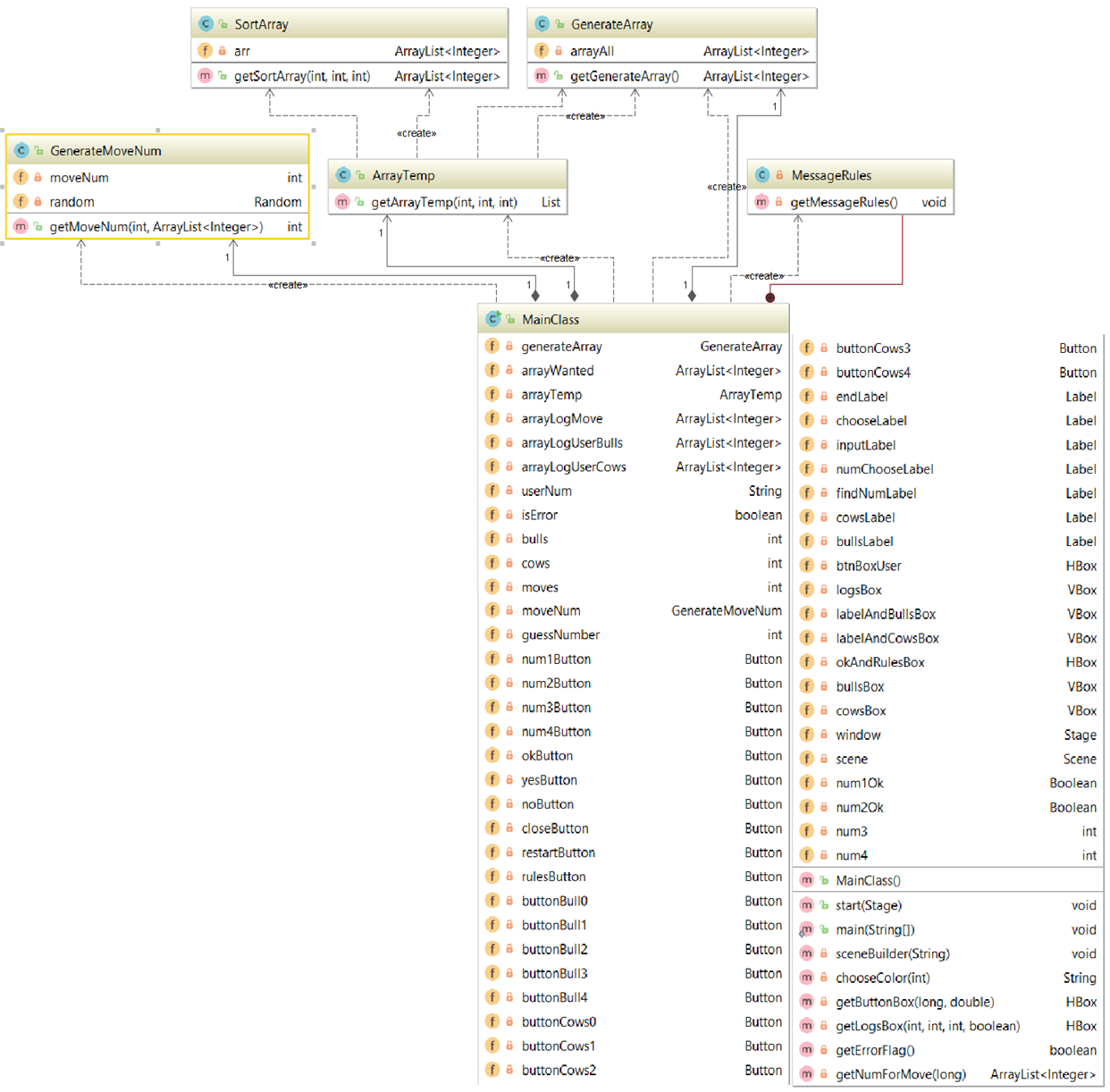


Рисунок 2 – Общая диаграмма классов

Основной класс MainClass, включает в себя конструктор, восемь методов и 53 поля (см. рисунок 3):

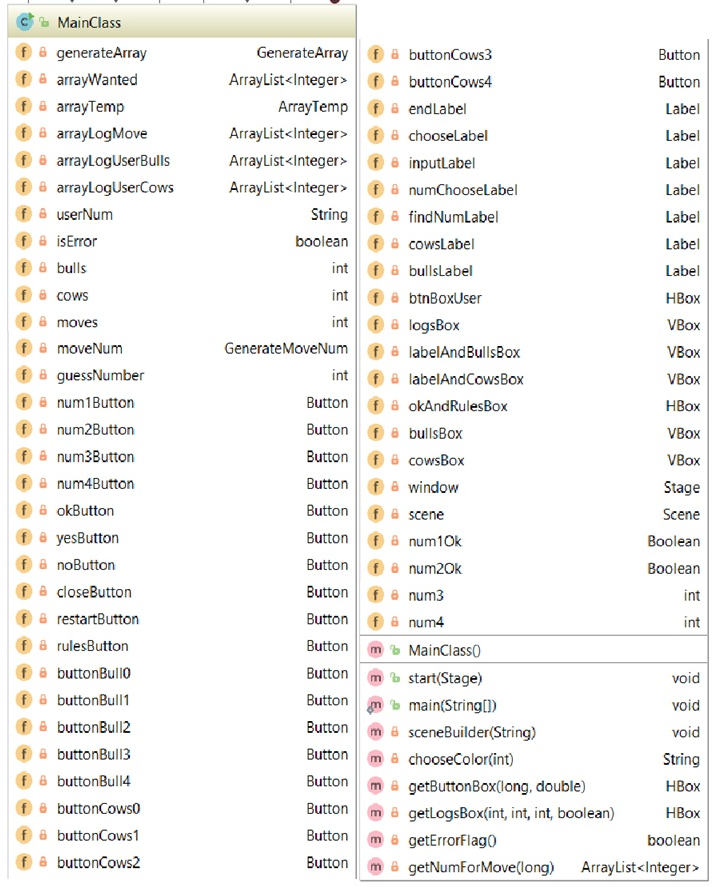


Рисунок 3 – Диаграмма класса MainClass

В конструкторе класса *MainClass()* генерируется начальный массив всех последовательностей с помощью getGenerateArray().

Поле *generateArray* является экземпляром класса GenerateArray – необходим для вызова через него метода getGenerateArray().

Поле *arrayWanted* имеет тип Array<Integer> – необходимо для сохранения в нем всех последовательностей, удовлетворяющих условиям. Когда после просеивания в нем останется одно значение, то это будет искомая секретная последовательность.

Поле *arrayTemp* является экземпляром класса ArrayTemp – необходим для вызова через него метода getArrayTemp().

Поля *arrayLogMove*, *arrayLogUserBulls* и *arrayLogUserCows* имеют тип Array<Integer> – необходимы для сохранения в них guessNumber, bulls и cows, соответственно. Используются для ведения лога.

Поле *userNum* имеет тип String – необходимо для сложения num1, num2, num3 и num4 в строку, чтобы получить общую последовательность.

Поле *isError* имеет тип boolean – является меткой наличия ошибки и определяет в окне завершения значение поля endLabel.

Поля *bulls*, *cows* и *moves* имеют тип int – в этих переменных сохраняются значения быков, коров и текущий ход, соответственно.

Поле *moveNum* является экземпляром класса GenerateMoveNum – необходим для вызова через него метода getMoveNum(int, ArrayList<Integer>).

Поле *guessNumber* имеет тип int – необходимо для хранения варианта секретной последовательности, варианты сохраняются в логе и выводятся на экран пользователю.

Поля *num1Button*, *num2Button*, *num3Button* и *num4Button* имеют типы Button – это кнопки, они лежат все вместе в btnBoxUser, на них установлены клик-эвенты. Помогают сохранить секретную последовательность пользователя в окне старта, для последующего вывода его в подсказках в каждом окне.

Поля *okButton*, *yesButton*, *noButton*, *closeButton*, *restartButton* и *rulesButton* тоже имеют тип Button и являются кнопками, которые имеют свои клик-эвенты на которых возложено управление окнами во время работы программы.

Поля *buttonBull0*, *buttonBull1*, *buttonBull2*, *buttonBull3* и *buttonBull4* имеют тип Button и являются кнопками, на них установлены клик-эвенты, они определяют количество быков, которые указал пользователь в ответ на заданный вариант последовательности компьютера.

Поля *buttonCows0*, *buttonCows1*, *buttonCows2*, *buttonCows3* и *buttonCows4* имеют тип Button и являются кнопками, на них установлены клик-эвенты, они определяют количество коров, которые указал пользователь в ответ на заданный вариант последовательности компьютера.

Поля *endLabel*, *chooseLabel*, *inputLabel*, *numChooseLabel* и *findNumLabel* имеют тип Label и являются этикетками. Внутри содержат форматированные строки, для вывода информации пользователю в соответствующих окнах, таких как окно завершения, окно выбора, окно ввода и окно старта, соответственно.

Поля *cowsLabel* и *bullsLabel* имеют тип Label и хранят в себе иконки устанавливаемые с помощью setGraphic(), которые находятся в папке resources.

Поле *btnBoxUser* имеет тип HBox и хранит в себе набор из 4 последовательностей выбранных пользователем для вывода в подсказках.

Поля *logBox*, *labelAndBullsBox*, *labelAndCowsBox* имеют тип VBox, в первом хранятся логи, во втором и третьем bullsLabel, bullsBox и cowsLabel, сowsBox, соответственно.

Поле *okAndRulesBox* имеет тип HBox и содержит в себе кнопку okButtton и rulesButton.

Поля *bullsBox* и *сowsBox* имеют тип VBox и содержат кнопки buttonBull0, buttonBull1, buttonBull2, buttonBull3, buttonBull4 и buttonCows0, buttonCows1, buttonCows2, buttonCows3, buttonCows4, соответственно.

Поле *window* имеет тип Stage является контейнером, в который помещаются все остальные компоненты интерфейса.

Поле *scene* имеет тип Scene представляет контейнер для всех графических элементов внутри объекта Stage в виде графа.

Поля *num1Ok, num2Ok, num3Ok* и *num4Ok* являются метками и имеют тип boolean. Используются в логических операторах, в клик-эвентах на кнопках num1Button, num2Button, num3Button и num4Button, чтобы сделать активной кнопку «ok», через okButton.setDisable(false).

Поля *num1*, *num2*, *num3* и *num4* имеют тип int и содержат в себе значение выбранного пользователем цвета в секретной последовательности. Находятся в клик-эвентах на кнопках num1Button, num2Button, num3Button и num4Button.

Метод *MainClass()* является конструктором класса MainClass, генерирует начальный массив всех последовательностей.

Метод *public static void main()* – главный метод, который является входной точкой в приложение. В этом методе вызывается стандартный статический метод javaFX Application.launch(args).

Метод *public void start(Stage)* – метод получает в аргументы объект типа Stage и дальше выполняется тело метода. Метод является обязательным в классе Application и переопределяется.

Метод *public void sceneBuilder(String)* – метод получает в аргументы объект типа String и определяет какую часть кода выполнить в теле метода, затем выполняет её. Метод выполняет построение сцены, которая следует следующей по логике программы.

Метод *private String chooseColor(int)* – метод получает в аргументы объект типа int, одно из чисел секретной последовательности и возвращает строку со стилем, который должна принимать кнопка в ответ на действие пользователя. Метод вызывается в стартовом окне в клик-эвентах на кнопках num1Button, num2Button, num3Button и num4Button.

Метод *private HBox getButtonBox(long, double)* – метод получает в аргументы объект типа long, четырехзначное число определяющее последовательность цветов и объект типа double, а также размер четырех кнопок, которые будут помещены в Hbox и возвращены методом в ответ на вызов. Цвет определяется методом javaFX setStyle(), которому в аргументах приходит стиль из метода chooseColor(int). Четырехзначное число разбивается на числовую последовательность методом getNumForMove(long).

Метод *private HBox getLogsBox(int, int, int, boolean)* – метод получает в аргументы четыре значения. Сгенерированную компьютером последовательность цветов в виде четырёхзначного числа, выбранное пользователем число быков и число коров, а также переменную с меткой наличия или отсутствия ошибки. Метод возвращает лог в виде Hbox, в котором генерируется цветовая последовательность и строки с ответом пользователя, при чём в зависимости от наличии ошибки меняет цвет строк на красный.

Метод *private boolean getErrorFlag()* – метод анализирует есть ли в ответах пользователя ошибка, сравнивая значения с расчетными. Возвращает метку о наличии или отсутствии ошибки с помощью переменной типа boolean.

Метод *private ArrayList<Integer> getNumForMove(long)* – метод получает в аргументы переменную типа long, число, которое преобразовывает в виде последовательности чисел и помещает в массив значений типа ArrayList<Integer>. Затем массив возвращается.

Основной класс содержит внутренний класс MessageRules. Он включает в себя один приватный метод (см. рисунок 4):



Рисунок 4 – Диаграмма внутреннего класса MessageRules

Метод *private void getMessageRules()* стандартными средствами javaFX создаёт окно сообщения пользователю, которое содержит правила игры. В теле метода создаётся экземпляр класса Stage и класса Scene. Далее окно наполняется элементами, а сами правила загружаются из файла типа html, с помощью объекта класса WebView, встроенного в javaFX.

#### 3.2 Дополнительные классы

В коде программы присутствуют так же несколько дополнительных классов.

Класс GenerateMoveNum включает в себя один метод и два поля (см. рисунок 5):

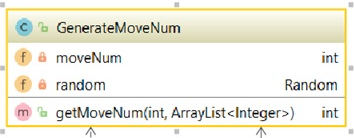


Рисунок 5 – Диаграмма класса GenerateMoveNum

Поле *moveNum* имеет тип int – хранит в себе генерированную последовательность для первых двух ходов компьютера, переменная возвращается в качестве результата работы метода *int* *getMoveNum(int, ArrayList<Integer.*

Поле *random* является объектом класса Random – экземпляр класса Random, необходим для получения случайного числа из последовательности, от 1 до 6.

Метод *public int* *getMoveNum(int, ArrayList<Integer>)* – метод получает в аргументы переменную типа long, которая хранит в себе текущий ход, а также переменную типа ArrayList<Integer>, в которой хранятся все последовательности удовлетворяющие условиям. Метод возвращает переменную типа int, которая является результатом генерации последовательности в теле метода. В зависимости от текущего хода метод генерирует особую случайную последовательность или возвращает её из массива.

Класс GenerateArray включает в себя метод и поле (см. рисунок 6):

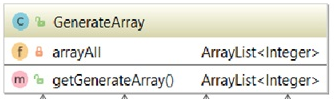


Рисунок 6 – Диаграмма класса GenerateArray

Поле *arrayAll* имеет тип ArrayList<Integer> – хранит в себе массив всех последовательностей, удовлетворяющих условиям.

Метод *public ArrayList<Integer> getGenerateArray(),* возвращает структуру данных ArrayList<Integer>, в ней хранятся все последовательности, удовлетворяющие условиям задачи, которые генерируются в теле метода с помощью цикла с условиями.

Класс SortArray включает в себя метод и поле (см. рисунок 7):

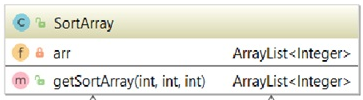


Рисунок 7 – Диаграмма класса SortArray

Поле *arr* имеет тип ArrayList<Integer> – хранит отсортированные последовательности, используется в качестве возвращаемого объекта.

Метод *public ArrayList<Integer> getSortArray(int, int, int)* принимает на вход три переменные типа int, первая хранят значение быков выбранное пользователем, второе значение коров, а третье последовательность определяющая эти значения. Метод возвращает отсортированный массив данных. Массив отсортировывается в теле метода исходя из входящих данных.

Класс ArrayTemp включает в себя один метод (см. рисунок 8):



Рисунок 8 – Диаграмма класса ArrayTemp

Метод *public List getArrayTemp(int, int, int)* принимает на вход три переменные типа int, первая хранят значение быков выбранное пользователем, второе значение коров, а третье последовательность определяющая эти значения. Метод возвращает отсортированный массив временных данных. Массив отсортировывается в теле метода исходя из входящих данных и возвращается.

#### 3.3 Дополнительные ресурсы

Некоторые части программы реализованы с использованием особых стилей и изображений. Эти дополнительные ресурсы находятся в папке 'resources'.

Изображения, которые используются на кнопках выбора количества «БЫКОВ» и «КОРОВ» лежат там под названиями: 'iconBull.png', 'iconCow.png', '0.png', '1.png', '2.png', '3.png' и '4.png'. Для установки изображений на кнопки используется стандартный метод javaFX setGraphic(). Два изображения под названиями 'imageStart.png' и 'imageWindow.jpg' используются в качестве фона в окнах. Для установки изображений в качестве фона используется стандартный метод javaFX setBackground(). Особые стили хранятся в файле 'style.css' и подключаются стандартным методом javaFX getStylesheets(). А также файл rules.html используется при появлении всплывающего окна с правилами игры. В всплывающем окне используется стандартный в javaFX класс WebView и стандартный метод getEngine(), который загружает html-страничку. Файл 'rules.html' имеет подключаемый файл, который задает стили с названием 'styleHTML.css'.

## 4 Руководство пользователя

#### 4.1 Минимальные системные требования

Для установки и запуска программы «BullsAndCows» необходимо выполнение следующих минимальных требований:

а) операционная система MS Windows (любая версия);

б) установленная виртуальная машина Java (Java Standard Edition), не ниже седьмой версии. Установочные файлы Java можно загрузить с сервера компании Oracle, по адресу: http://oracle.com.

#### 4.2 Установка и удаление программы

Для установки программы на ПК необходимо:

а) скопировать jar-архив с программой «BullsAndCows» в любую директорию вашей системы (например, диск «С:\»);

б) чтобы запустить программу необходимо просто щелкнуть два раза левой кнопкой мыши по jar-архиву или скопируйте его на рабочий стол (правой кнопкой на файл, далее пункт пеню «Отправить» на «Рабочий стол») и запускайте программу с рабочего стола;

в) чтобы удалить программу просто удалите папку с программой из вашей директории.

#### 4.3 Работа с программой

Для того чтобы сыграть в «BullsAndCows», в первом окне выберите свою секретную последовательность из четырех цветных шаров и нажмите кнопку «Готово». Дальше следуйте подсказкам в следующем окне.

В программе есть окно «Меню» с функциями «Начать сначала», «Правила» и «Выход», также эти функции вызываются комбинациями клавиш «Ctrl+N», «Ctrl+P» и «Ctrl+E», соответственно.

#### 4.4 Результат работы программы

После запуска программы мы видим начальное окно с выбором секретной последовательности пользователя (см. рисунок 9). После выбора жмем «Готово» и появляется окно выбора (см. рисунок 10). В этом окне сверху видим комбинацию цветов, сгенерированную компьютером и нашу комбинацию внизу экрана, также две кнопки с выбором дальнейших действий. В верхнем углу программы мы видим меню (см. рисунок 11). Если это комбинация цветов не угадана компьютером мы нажимаем кнопку «Нет» и попадаем в окно ввода количества «быков» и «коров» (см. рисунок 12). После ввода количества «быков» кнопками слева и количества «коров» кнопками справа нажимаем кнопку «Готово». Когда последовательность цветов компьютера будет угадана мы попадем в заключительное окно (см. рисунок 13). Из разных частей программы можно вызвать окно с правилами игры (см. рисунок 14).

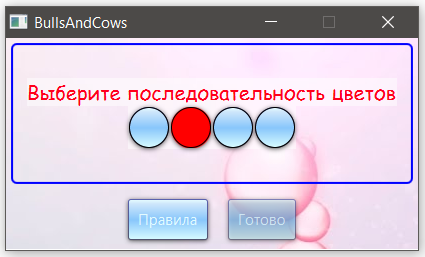


Рисунок 9 – Начальное окно

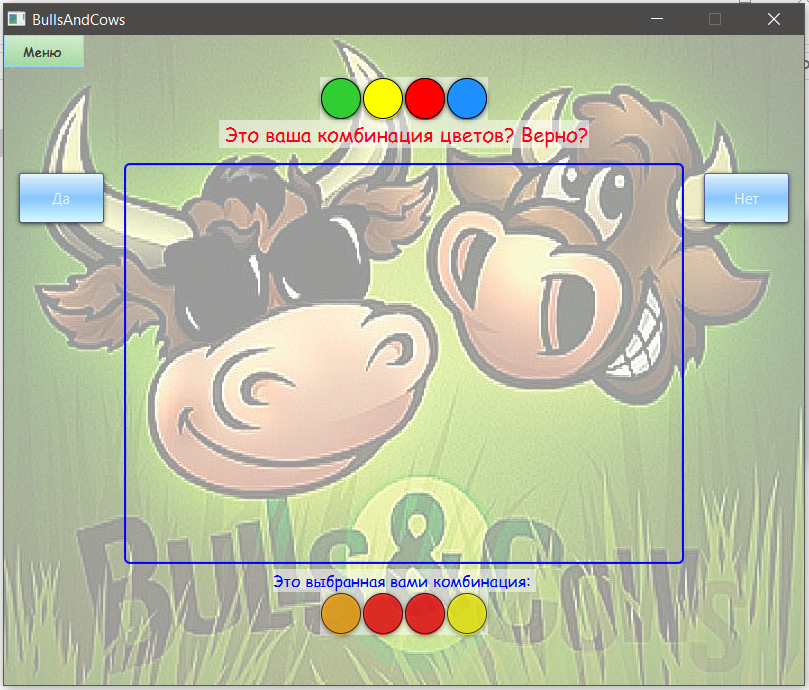


Рисунок 10 – Окно выбора

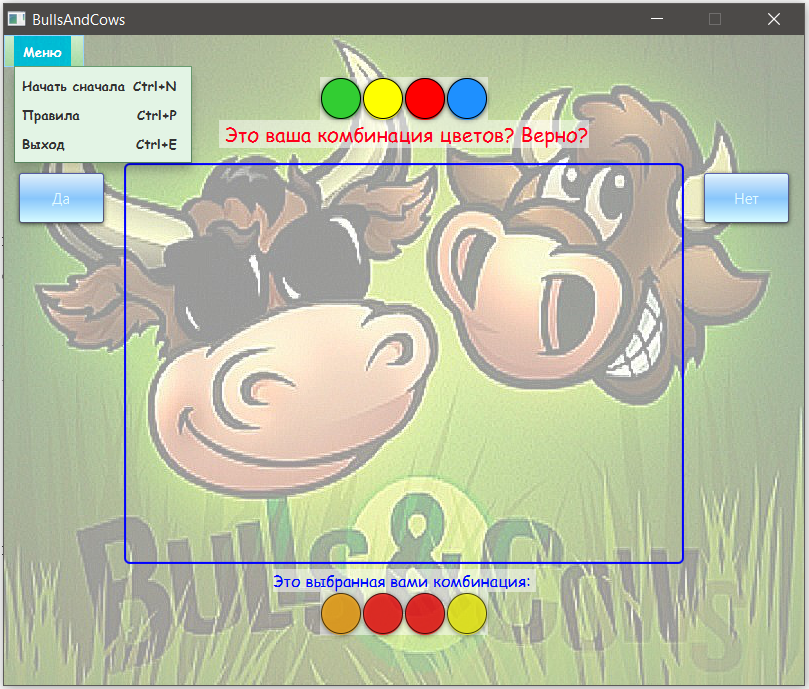


Рисунок 11 – Меню программы

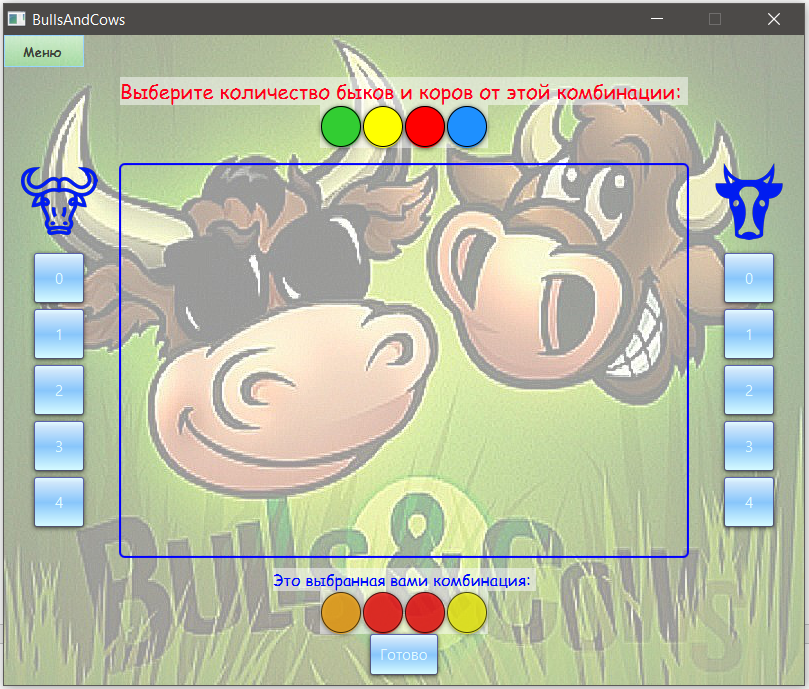


Рисунок 12 – Окно ввода

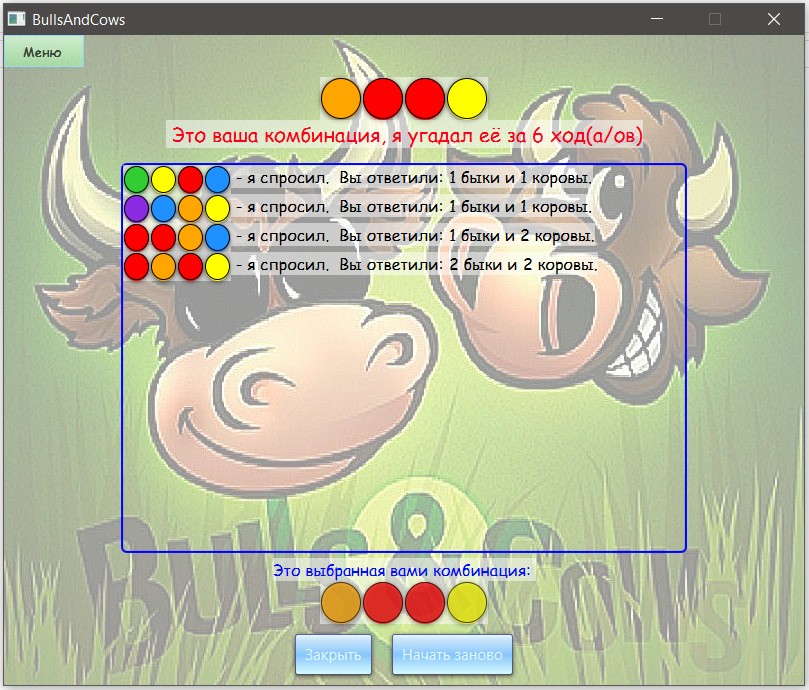


Рисунок 13 – Завершающее окно

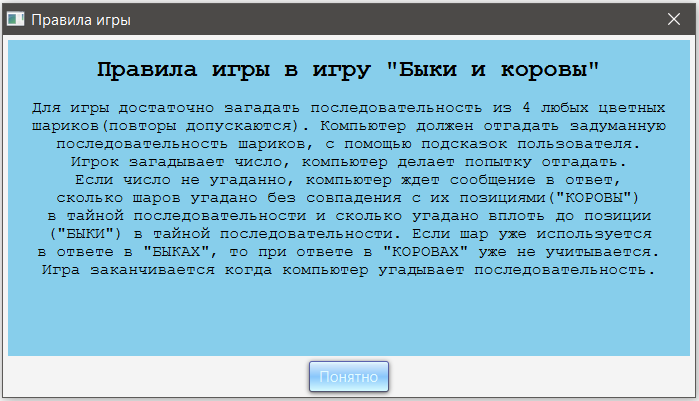


Рисунок 14 – Окно правил игры

# ЗАКЛЮЧЕНИЕ

В курсовой работе был описан процесс разработки программы «BullsAndCows». Исследованы методы поиска оптимального маршрута в задаче о коммивояжере. В ходе тестирования была проверена работа всех функций программы и установлено, что программа полностью работоспособна и способна угадывать загаданную последовательность максимум за шесть ходов. Пояснительная записка к курсовой работе включает в себя руководство пользователя, описывающее: системные требования, процессы установки, настройки и удаления программы, а также правила пользования. Исходный код программы приведен в приложении А. Окончательная версия программы – консольное Windows-приложение для Java платформы.

Допустимо дальнейшее усовершенствование программы, например, можно переделать программу под игру против компьютера, в которой пользователь с компьютером будут поочередно пытаться угадать секретную последовательность друг друга. Эту задачу легко реализовать, воспользовавшись и немного изменив некоторые используемые в программе методы. В частности, чтобы реализовать уровни сложности нужно поиграться с методом генерации ходов компьютера и ухудшить где это нужно логику выбора последовательности.

В целом поставленная в курсовой работе задача была решена.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лафоре Р. Структуры данных и алгоритмы в Java, 2–е изд. [Текст] : пер. с англ. / Р. Лафоре, Е. Матвеев. – СПб. : Питер, 2018. – 704 с.
2. Мясников И. Е. Государственная итоговая аттестация [Текст] : учеб.-метод. Пособие / И. Е. Мясников, Н. Р. Спиричева, С. И. Тимощенко. – Екатеринбург : Урал. ун-та, 2017. – 104 с.
3. Окулов С. М. Программирование в аглоритмах, 4–е изд. [Текст] / С. М. Окулов. – М. : Бином , 2007. – 384 с.
4. Суханов В. И. Программирование графических интерфейсов с использованием платформы JavaFX : учеб. пособие для студ. вузов [Текст] / В. И. Суханов. – Екатеринбург : УрФУ, 2017. – 78 с.

# ПРИЛОЖЕНИЕ А (обязательное) Исходный текст

### А.1 Основной класс

package main;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Cursor;

import javafx.scene.control.\*;

import javafx.scene.effect.BlendMode;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.KeyCombination;

import javafx.scene.layout.\*;

import javafx.scene.shape.Circle;

import javafx.scene.web.WebEngine;

import javafx.scene.web.WebView;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.scene.Scene;

import java.io.\*;

import java.net.URL;

import java.util.\*;

public class MainClass extends Application {

private GenerateArray generateArray = new GenerateArray();

private ArrayList<Integer> arrayWanted;

private ArrayTemp arrayTemp = new ArrayTemp();

private ArrayList<Integer> arrayLogMove = new ArrayList();

private ArrayList<Integer> arrayLogUserBulls;

private ArrayList<Integer> arrayLogUserCows;

private String userNum;

private boolean isError = false;

private int bulls, cows, moves;

private GenerateMoveNum moveNum = new GenerateMoveNum();

private int guessNumber;

private Button num1Button, num2Button, num3Button, num4Button;

private Button okButton, yesButton, noButton, closeButton, restartButton, rulesButton;

private Button buttonBull0, buttonBull1, buttonBull2, buttonBull3, buttonBull4;

private Button buttonCows0, buttonCows1, buttonCows2, buttonCows3, buttonCows4;

private Label endLabel, chooseLabel, inputLabel, numChooseLabel, findNumLabel, cowsLabel, bullsLabel;

private HBox btnBoxUser;

private VBox logsBox = new VBox();

private VBox labelAndBullsBox = new VBox();

private VBox labelAndCowsBox;

private HBox okAndRulesBox;

private VBox bullsBox = new VBox();

private VBox cowsBox = new VBox();

private Stage window;

private Scene scene;

private Boolean num1Ok = false, num2Ok = false, num3Ok = false, num4Ok = false;

private Boolean bullsChoose = false, cowsChoose = false;

private int num1 = 0, num2 = 0, num3 = 0, num4 = 0;

public MainClass() {

//Генерирует начальный массив всех последовательностей

arrayWanted = generateArray.getGenerateArray();

moves = 1;

arrayLogUserCows = new ArrayList();

arrayLogUserBulls = new ArrayList();

btnBoxUser = new HBox();

labelAndCowsBox = new VBox();

okAndRulesBox = new HBox();

}

@Override

public void start(Stage primaryStage) {

try {

//Инициализвация и построение окна СТАРТ

window = primaryStage;

window.setTitle("BullsAndCows");

BorderPane root = new BorderPane();

VBox fullBox = new VBox();

okButton = new Button("Готово");

yesButton = new Button("Да");

noButton = new Button("Нет");

closeButton = new Button("Закрыть");

restartButton = new Button("Начать заново");

rulesButton = new Button("Правила");

okButton.setMinHeight(40);

closeButton.setMinHeight(40);

restartButton.setMinHeight(40);

rulesButton.setMinHeight(40);

bullsLabel = new Label();

cowsLabel = new Label();

num1Button = new Button();

num2Button = new Button();

num3Button = new Button();

num4Button = new Button();

buttonBull0 = new Button("0");

buttonBull1 = new Button("1");

buttonBull2 = new Button("2");

buttonBull3 = new Button("3");

buttonBull4 = new Button("4");

buttonCows0 = new Button("0");

buttonCows1 = new Button("1");

buttonCows2 = new Button("2");

buttonCows3 = new Button("3");

buttonCows4 = new Button("4");

buttonBull0.setMinSize(50, 50);

buttonBull1.setMinSize(50, 50);

buttonBull2.setMinSize(50, 50);

buttonBull3.setMinSize(50, 50);

buttonBull4.setMinSize(50, 50);

buttonCows0.setMinSize(50, 50);

buttonCows1.setMinSize(50, 50);

buttonCows2.setMinSize(50, 50);

buttonCows3.setMinSize(50, 50);

buttonCows4.setMinSize(50, 50);

VBox.setMargin(buttonBull0, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonBull1, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonBull2, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonBull3, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonBull4, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonCows0, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonCows1, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonCows2, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonCows3, new Insets(3, 30, 3, 30));

VBox.setMargin(buttonCows4, new Insets(3, 30, 3, 30));

VBox.setMargin(bullsLabel, new Insets(5, 0, 10, 0));

VBox.setMargin(cowsLabel, new Insets(5, 0, 10, 0));

num1Button.setShape(new Circle(2));

num2Button.setShape(new Circle(2));

num3Button.setShape(new Circle(2));

num4Button.setShape(new Circle(2));

num1Button.setStyle("-fx-border-color: black;");

num2Button.setStyle("-fx-border-color: black;");

num3Button.setStyle("-fx-border-color: black;");

num4Button.setStyle("-fx-border-color: black;");

num1Button.setPadding(new Insets(10, 20, 10, 20));

num2Button.setPadding(new Insets(10, 20, 10, 20));

num3Button.setPadding(new Insets(10, 20, 10, 20));

num4Button.setPadding(new Insets(10, 20, 10, 20));

okButton.setPadding(new Insets(10, 10, 10, 10));

yesButton.setPadding(new Insets(10, 10, 10, 10));

noButton.setPadding(new Insets(10, 10, 10, 10));

closeButton.setPadding(new Insets(10, 10, 10, 10));

restartButton.setPadding(new Insets(10, 10, 10, 10));

rulesButton.setPadding(new Insets(10, 10, 10, 10));

yesButton.setMinHeight(50);

yesButton.setMinWidth(85);

noButton.setMinHeight(50);

noButton.setMinWidth(85);

BorderPane.setMargin(okButton, new Insets(15));

BorderPane.setMargin(yesButton, new Insets(15));

BorderPane.setMargin(noButton, new Insets(15));

BorderPane.setMargin(closeButton, new Insets(15));

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/iconBull.png"))));

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/iconCow.png"))));

bullsBox.getChildren().addAll(buttonBull0, buttonBull1, buttonBull2, buttonBull3, buttonBull4);

cowsBox.getChildren().addAll(buttonCows0, buttonCows1, buttonCows2, buttonCows3, buttonCows4);

labelAndBullsBox.setAlignment(Pos.TOP\_CENTER);

labelAndCowsBox.setAlignment(Pos.TOP\_CENTER);

labelAndBullsBox.getChildren().addAll(bullsLabel, bullsBox);

labelAndCowsBox.getChildren().addAll(cowsLabel, cowsBox);

okButton.setDisable(true);

numChooseLabel = new Label("Выберите последовательность цветов");

numChooseLabel.setId("chooseLabel");

//Эвенты устанавливают цвет кнопок num1 num2 num3 num4 в зависимости от значения

num1Button.setOnAction(event -> {

num1Ok = true;

if (num2Ok & num3Ok & num4Ok) {

okButton.setDisable(false);

}

num1++;

num1Button.setStyle(chooseColor(num1) + "-fx-border-color: black;");

if (num1 == 6) num1 = 0;

});

num2Button.setOnAction(event -> {

num2Ok = true;

if (num1Ok & num3Ok & num4Ok) {

okButton.setDisable(false);

}

num2++;

num2Button.setStyle(chooseColor(num2) + "-fx-border-color: black;");

if (num2 == 6) num2 = 0;

});

num3Button.setOnAction(event -> {

num3Ok = true;

if (num1Ok & num2Ok & num4Ok) {

okButton.setDisable(false);

}

num3++;

num3Button.setStyle(chooseColor(num3) + "-fx-border-color: black;");

if (num3 == 6) num3 = 0;

});

num4Button.setOnAction(event -> {

num4Ok = true;

if (num1Ok & num2Ok & num3Ok) {

okButton.setDisable(false);

}

num4++;

num4Button.setStyle(chooseColor(num4) + "-fx-border-color: black;");

if (num4 == 6) num4 = 0;

});

okButton.setOnAction(event -> {

if (num1 == 0) num1 = 6;

if (num2 == 0) num2 = 6;

if (num3 == 0) num3 = 6;

if (num4 == 0) num4 = 6;

userNum = "" + num1 + num2 + num3 + num4;

System.out.println("user num" + userNum);

num1Button.setDisable(true);

num2Button.setDisable(true);

num3Button.setDisable(true);

num4Button.setDisable(true);

sceneBuilder("choose");

});

rulesButton.setOnAction(event -> {

new MessageRules().getMessageRules();

});

btnBoxUser.getChildren().addAll(num1Button, num2Button, num3Button, num4Button);

btnBoxUser.setAlignment(Pos.CENTER);

btnBoxUser.setMaxWidth(50);

btnBoxUser.setId("btnBoxUser");

okAndRulesBox.getChildren().addAll(rulesButton, okButton);

fullBox.getChildren().add(numChooseLabel);

fullBox.getChildren().add(btnBoxUser);

fullBox.setAlignment(Pos.CENTER);

fullBox.setId("fullBox");

HBox.setMargin(num1Button, new Insets(1, 1, 1, 1));

HBox.setMargin(num2Button, new Insets(1, 1, 1, 1));

HBox.setMargin(num3Button, new Insets(1, 1, 1, 1));

HBox.setMargin(num4Button, new Insets(1, 1, 1, 1));

root.setCenter(fullBox);

okAndRulesBox.setSpacing(20);

okAndRulesBox.setPadding(new Insets(10, 0, 10, 0));

okAndRulesBox.setAlignment(Pos.CENTER);

root.setBottom(okAndRulesBox);

BackgroundImage myBI = new BackgroundImage(new Image("main/resources/imageStart.png"),

BackgroundRepeat.REPEAT, BackgroundRepeat.NO\_REPEAT, BackgroundPosition.DEFAULT, BackgroundSize.DEFAULT);

root.setBackground(new Background(myBI));

scene = new Scene(root, 400, 200);

scene.getStylesheets().addAll(this.getClass().getResource("resources/style.css").toExternalForm());

window.setResizable(false);

window.setScene(scene);

window.show();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

launch(args);

}

private void sceneBuilder(String nameScene) {

BorderPane root = new BorderPane();

VBox fullBox = new VBox();

VBox menuAndFullBox = new VBox();

VBox findNumBox = new VBox();

fullBox.setPadding(new Insets(10, 0, 10, 0));

findNumLabel = new Label("Это выбранная вами комбинация: ");

findNumLabel.setMaxWidth(263);

findNumLabel.setId("findNumLabel");

logsBox.setId("logsBox");

MenuBar menuBar = new MenuBar();

menuBar.setLayoutX(0);

menuBar.setLayoutY(0);

menuBar.setBlendMode(BlendMode.HARD\_LIGHT);

menuBar.setCursor(Cursor.CLOSED\_HAND);

menuBar.setMaxWidth(80);

Menu menuF = new Menu("Меню");

menuF.setId("menuF");

MenuItem itemNew = new MenuItem("Начать сначала");

itemNew.setAccelerator(KeyCombination.keyCombination("Ctrl+N"));

itemNew.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

MainClass mainClass = new MainClass();

window.close();

Platform.runLater(() -> mainClass.start(new Stage()));

}

});

MenuItem itemHelp = new MenuItem("Правила");

itemHelp.setAccelerator(KeyCombination.keyCombination("Ctrl+P"));

itemHelp.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

new MessageRules().getMessageRules();

}

});

MenuItem itemExit = new MenuItem("Выход");

itemExit.setAccelerator(KeyCombination.keyCombination("Ctrl+E"));

itemExit.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Platform.exit();

}

});

menuF.getItems().addAll(itemNew, itemHelp, itemExit);

menuBar.getMenus().add(menuF);

menuAndFullBox.getChildren().addAll(menuBar, fullBox);

root.setTop(menuAndFullBox);

okButton.setOnAction(event -> {

arrayLogUserBulls.add(bulls);

System.out.println("bulls set" + bulls);

arrayLogUserCows.add(cows);

System.out.println("cows set" + cows);

//Отсеивание последовательностей неудовлетворяющих условиям

arrayWanted.removeAll(arrayTemp.getArrayTemp(bulls, cows, guessNumber));

//Используя метод getErrorFlag() создает лог

if (getErrorFlag()) {

logsBox.getChildren().addAll(getLogsBox(guessNumber, bulls, cows, false));

} else {

logsBox.getChildren().addAll(getLogsBox(guessNumber, bulls, cows, true));

isError = true;

sceneBuilder("end");

}

if (!isError) sceneBuilder("choose");

if (arrayWanted.size() == 1) {

sceneBuilder("end");

}

});

yesButton.setOnAction(event -> sceneBuilder("end"));

noButton.setOnAction(event -> {

if (moves == 1) okButton.setDisable(true);

sceneBuilder("input");

});

closeButton.setOnAction(event -> sceneBuilder("close"));

//По метке "input" строит окно ввода быков и коров

if (nameScene.equals("input")) {

//Эвенты определяют количества быков и иконку кнопки их выбора

buttonBull0.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/0.png"))));

bulls = 0;

bullsChoose = true;

if (cowsChoose) {

okButton.setDisable(false);

}

}

});

buttonBull1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/1.png"))));

bulls = 1;

bullsChoose = true;

if (cowsChoose) {

okButton.setDisable(false);

}

}

});

buttonBull2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/2.png"))));

bulls = 2;

bullsChoose = true;

if (cowsChoose) {

okButton.setDisable(false);

}

}

});

buttonBull3.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/3.png"))));

bulls = 3;

bullsChoose = true;

if (cowsChoose) {

okButton.setDisable(false);

}

}

});

buttonBull4.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

bullsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/4.png"))));

bulls = 4;

bullsChoose = true;

if (cowsChoose) {

okButton.setDisable(false);

}

}

});

//Эвенты определяют количества коров и иконку кнопки их выбора

buttonCows0.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/0.png"))));

cows = 0;

cowsChoose = true;

if (bullsChoose) {

okButton.setDisable(false);

}

}

});

buttonCows1.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/1.png"))));

cows = 1;

cowsChoose = true;

if (bullsChoose) {

okButton.setDisable(false);

}

}

});

buttonCows2.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/2.png"))));

cows = 2;

cowsChoose = true;

if (bullsChoose) {

okButton.setDisable(false);

}

}

});

buttonCows3.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/3.png"))));

cows = 3;

cowsChoose = true;

if (bullsChoose) {

okButton.setDisable(false);

}

}

});

buttonCows4.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

cowsLabel.setGraphic(new ImageView(new Image(getClass().getResourceAsStream("resources/4.png"))));

cows = 4;

cowsChoose = true;

if (bullsChoose) {

okButton.setDisable(false);

}

}

});

inputLabel = new Label("Выберите количество быков и коров от этой комбинации: ");

inputLabel.setId("inputLabel");

BorderPane.setMargin(fullBox, new Insets(20, 120, 20, 114));

HBox btnBox = getButtonBox(guessNumber, 1);

btnBox.setAlignment(Pos.CENTER);

btnBox.setMaxWidth(50);

btnBox.setId("btnBox");

fullBox.getChildren().addAll(inputLabel, btnBox);

fullBox.setAlignment(Pos.CENTER);

findNumBox.getChildren().addAll(findNumLabel, btnBoxUser, okButton);

findNumBox.setAlignment(Pos.CENTER);

findNumBox.setPadding(new Insets(5, 0, 10, 0));

root.setTop(menuAndFullBox);

root.setBottom(findNumBox);

root.setLeft(labelAndBullsBox);

root.setRight(labelAndCowsBox);

root.setCenter(logsBox);

BorderPane.setMargin(logsBox, new Insets(0, 0, 0, 0));

}

//По метке "end" строит окно завершения

if (nameScene.equals("end")) {

if (!isError) {

endLabel = new Label(" Это ваша комбинация, я угадал её за " + moves + " ход(а/ов)");

}

if (isError) {

endLabel = new Label("Увы, вы допустили ошибку! ");

}

endLabel.setId("endLabel");

BorderPane.setMargin(fullBox, new Insets(20, 114, 20, 110));

HBox btnBox = getButtonBox(guessNumber, 1);

btnBox.setAlignment(Pos.CENTER);

btnBox.setMaxWidth(50);

btnBox.setId("btnBox");

fullBox.getChildren().addAll(btnBox, endLabel);

fullBox.setAlignment(Pos.CENTER);

HBox endHBox = new HBox();

endHBox.setSpacing(20);

endHBox.setPadding(new Insets(10, 0, 10, 0));

endHBox.getChildren().addAll(closeButton, restartButton);

endHBox.setAlignment(Pos.CENTER);

restartButton.setOnAction(event -> {

//Эвент создает еще один экземпляр программы, а текущй закрывает

MainClass mainClass = new MainClass();

window.close();

Platform.runLater(() -> mainClass.start(new Stage()));

});

findNumBox.getChildren().addAll(findNumLabel, btnBoxUser, endHBox);

findNumBox.setAlignment(Pos.CENTER);

root.setBottom(findNumBox);

root.setTop(menuAndFullBox);

root.setCenter(logsBox);

BorderPane.setMargin(logsBox, new Insets(0, 112, 0, 112));

}

//По метке "choose" строит окно выбора

if (nameScene.equals("choose")) {

try {

//Выбор варианта секретной последовательности через метод getMoveNum(moves, arrayWanted);

guessNumber = moveNum.getMoveNum(moves, arrayWanted);

arrayLogMove.add(guessNumber);

System.out.println("num move" + guessNumber);

moves++;

} catch (Exception e) {

isError = true;

sceneBuilder("end");

}

chooseLabel = new Label(" Это ваша комбинация цветов? Верно?");

chooseLabel.setId("chooseLabel");

BorderPane.setMargin(fullBox, new Insets(20, 114, 20, 114));

HBox btnBox = getButtonBox(guessNumber, 1);

btnBox.setAlignment(Pos.CENTER);

btnBox.setMaxWidth(50);

btnBox.setId("btnBox");

fullBox.getChildren().addAll(btnBox, chooseLabel);

fullBox.setAlignment(Pos.CENTER);

root.setTop(menuAndFullBox);

root.setLeft(yesButton);

root.setRight(noButton);

findNumBox.getChildren().addAll(findNumLabel, btnBoxUser);

root.setBottom(findNumBox);

root.setCenter(logsBox);

BorderPane.setMargin(findNumBox, new Insets(0, 0, 50, 0));

findNumBox.setAlignment(Pos.CENTER);

BorderPane.setAlignment(yesButton, Pos.TOP\_LEFT);

BorderPane.setAlignment(noButton, Pos.TOP\_RIGHT);

}

//По метке "close" закрывает программу

if (nameScene.equals("close")) {

System.exit(0);

}

root.setId("pane");

scene = new Scene(root, 800, 650);

scene.getStylesheets().addAll(this.getClass().getResource("resources/style.css").toExternalForm());

window.setScene(scene);

window.centerOnScreen();

window.show();

}

private String chooseColor(int num) {

if (num == 1) {

return "-fx-background-color: RED;";

}

if (num == 2) {

return "-fx-background-color: ORANGE;";

}

if (num == 3) {

return "-fx-background-color: YELLOW;";

}

if (num == 4) {

return "-fx-background-color: limegreen;";

}

if (num == 5) {

return "-fx-background-color: dodgerblue;";

}

if (num == 6) {

return "-fx-background-color: blueviolet;";

}

return "";

}

private HBox getButtonBox(long number, double sizeButton) {

HBox btnBox = new HBox();

Button num1Button = new Button();

Button num2Button = new Button();

Button num3Button = new Button();

Button num4Button = new Button();

num1Button.setShape(new Circle(2));

num2Button.setShape(new Circle(2));

num3Button.setShape(new Circle(2));

num4Button.setShape(new Circle(2));

num1Button.setPadding(new Insets(10 / sizeButton, 20, 10 / sizeButton, 20 / sizeButton));

num2Button.setPadding(new Insets(10 / sizeButton, 20, 10 / sizeButton, 20 / sizeButton));

num3Button.setPadding(new Insets(10 / sizeButton, 20, 10 / sizeButton, 20 / sizeButton));

num4Button.setPadding(new Insets(10 / sizeButton, 20, 10 / sizeButton, 20 / sizeButton));

HBox.setMargin(num1Button, new Insets(1));

HBox.setMargin(num2Button, new Insets(1));

HBox.setMargin(num3Button, new Insets(1));

HBox.setMargin(num4Button, new Insets(1));

ArrayList<Integer> arrayButton = getNumForMove(number);

num1Button.setStyle(chooseColor(arrayButton.get(0)) + "-fx-border-color: black;");

num2Button.setStyle(chooseColor(arrayButton.get(1)) + "-fx-border-color: black;");

num3Button.setStyle(chooseColor(arrayButton.get(2)) + "-fx-border-color: black;");

num4Button.setStyle(chooseColor(arrayButton.get(3)) + "-fx-border-color: black;");

btnBox.getChildren().addAll(num1Button, num2Button, num3Button, num4Button);

return btnBox;

}

private HBox getLogsBox(int num, int numBulls, int numCows, boolean errorFlag) {

Label logLabelBefore = new Label(" - я спросил. ");

logLabelBefore.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: black;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

Label logLabelAfter = new Label("Вы ответили: ");

logLabelAfter.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: black;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

HBox logBox = new HBox();

HBox buttonBoxTemp = getButtonBox(num, 4);

buttonBoxTemp.setId("buttonBoxTemp");

Label numBullsLabel = new Label(numBulls + " быки и ");

if (!errorFlag) {

numBullsLabel.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: black;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

} else {

numBullsLabel.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: red;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

}

Label numCowsLabel = new Label(numCows + " коровы.");

if (!errorFlag) {

numCowsLabel.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: black;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

} else {

numCowsLabel.setStyle("-fx-font-size: 12pt;" + "-fx-text-fill: red;" + "-fx-background-color: rgba(255, 255, 255, 0.5);");

}

logBox.getChildren().addAll(buttonBoxTemp, logLabelBefore, logLabelAfter, numBullsLabel, numCowsLabel);

return logBox;

}

private boolean getErrorFlag() {

boolean errorFlag = false;

int bullsReal = 0;

int cowsReal = 0;

int lastMove = arrayLogMove.get(arrayLogMove.size() - 1);

ArrayList<Integer> numsForLastMove = getNumForMove(lastMove); // 1 2 3 4

ArrayList<Integer> numsForUserNum = getNumForMove(Integer.parseInt(userNum)); // 1 2 3 4

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

if (numsForUserNum.get(i).equals(numsForLastMove.get(i))) {

numsForUserNum.set(i, 0);

numsForLastMove.set(i, 0);

bullsReal++;

}

}

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

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

if (numsForUserNum.get(i) != 0 & numsForLastMove.get(j) != 0 & numsForUserNum.get(i).equals(numsForLastMove.get(j))) {

numsForUserNum.set(i, 0);

numsForLastMove.set(j, 0);

cowsReal++;

}

}

}

if (bullsReal == arrayLogUserBulls.get(arrayLogUserBulls.size() - 1) & cowsReal == arrayLogUserCows.get(arrayLogUserCows.size() - 1)) {

errorFlag = true;

}

return errorFlag;

}

private ArrayList<Integer> getNumForMove(long move) {

ArrayList<Integer> result = new ArrayList<>();

String s = Long.toString(move);

for (int i = 0; i < s.length(); i++) {

result.add(s.charAt(i) - '0');

}

return result;

}

private class MessageRules {

private void getMessageRules() {

Stage messageRulesWindow = new Stage();

messageRulesWindow.setResizable(false);

messageRulesWindow.initModality(Modality.APPLICATION\_MODAL);

messageRulesWindow.setTitle("Правила игры");

Button closeRulesButton = new Button("Понятно");

closeRulesButton.setOnAction(e -> messageRulesWindow.close());

WebView browser = new WebView();

WebEngine webEngine = browser.getEngine();

VBox root = new VBox();

File file = null;

String resource = "/main/resources/rules.html";

URL url = getClass().getResource(resource);

if (url.toString().startsWith("jar:")) {

try {

InputStream input = getClass().getResourceAsStream(resource);

file = File.createTempFile(new Date().getTime() + "", ".html");

OutputStream out = new FileOutputStream(file);

int read;

byte[] bytes = new byte[1024];

while ((read = input.read(bytes)) != -1) {

out.write(bytes, 0, read);

}

out.flush();

out.close();

input.close();

file.deleteOnExit();

} catch (IOException ex) {

ex.printStackTrace();

}

} else {

//this will probably work in your IDE, but not from a JAR

file = new File(url.getFile());

}

//URL url = MessageRules.class.getClassLoader().getResource("./main/resources/rules.html");

//assert url != null;

webEngine.load(url.toString());

root.setAlignment(Pos.CENTER);

root.setPadding(new Insets(5));

root.setSpacing(5);

root.getChildren().addAll(browser, closeRulesButton);

Scene scene = new Scene(root, 680, 350);

scene.getStylesheets().addAll(this.getClass().getResource("resources/style.css").toExternalForm());

messageRulesWindow.setScene(scene);

messageRulesWindow.show();

}

}

}

### А.2 Дополнительные классы

package main;

import java.util.ArrayList;

import java.util.Random;

public class GenerateMoveNum {

private int moveNum;

private Random random = new Random();

public int getMoveNum(int move, ArrayList<Integer> arrayWanted) {

if (move == 1) {

int num1;

int num2;

int num3;

int num4;

for (; ; ) {

num1 = random.nextInt(6) + 1;

if (num1 != 2 & num1 != 6) break;

}

for (; ; ) {

num2 = random.nextInt(6) + 1;

if (num1 != num2 & num2 != 2 & num2 != 6) break;

}

for (; ; ) {

num3 = random.nextInt(6) + 1;

if (num1 != num3 & num2 != num3 & num3 != 2 & num3 != 6) break;

}

for (; ; ) {

num4 = random.nextInt(6) + 1;

if (num1 != num4 & num2 != num4 & num3 != num4 & num4 != 2 & num4 != 6) break;

}

moveNum = Integer.parseInt(num1 + "" + num2 + num3 + num4);

}

if (move == 2) {

int num1;

int num2;

int num3;

int num4;

for (; ; ) {

num1 = random.nextInt(6) + 1;

if (num1 != 1 & num1 != 4) break;

}

for (; ; ) {

num2 = random.nextInt(6) + 1;

if (num1 != num2 & num2 != 1 & num2 != 4) break;

}

for (; ; ) {

num3 = random.nextInt(6) + 1;

if (num1 != num3 & num2 != num3 & num3 != 1 & num3 != 4) break;

}

for (; ; ) {

num4 = random.nextInt(6) + 1;

if (num1 != num4 & num2 != num4 & num3 != num4 & num4 != 1 & num4 != 4) break;

}

moveNum = Integer.parseInt(num1 + "" + num2 + num3 + num4);

}

if (move > 2) {

moveNum = arrayWanted.get(0);

}

return moveNum;

}

}

package main;

import java.util.ArrayList;

public class SortArray {

private ArrayList<Integer> arr = new ArrayList();

public ArrayList<Integer> getSortArray(int bulls, int cows, int guessNumber) {

for (int i = 1111; i <= 6666; i++) {

if (i % 10 != 0 && i % 10 != 7 && i % 10 != 8 && i % 10 != 9 &&

(i / 10) % 10 != 0 && (i / 10) % 10 != 7 && (i / 10) % 10 != 8 && (i / 10) % 10 != 9 &&

(i / 100) % 10 != 0 && (i / 100) % 10 != 7 && (i / 100) % 10 != 8 && (i / 100) % 10 != 9) {

int currentBulls = 0;

int currentCows = 0;

int currentNumber = i;

int currentD = currentNumber % 10;

currentNumber /= 10;

int currentC = currentNumber % 10;

currentNumber /= 10;

int currentB = currentNumber % 10;

currentNumber /= 10;

int currentA = currentNumber % 10;

currentNumber /= 10;

int tempGuessNumber = guessNumber;

int guessNumberD = tempGuessNumber % 10;

tempGuessNumber /= 10;

int guessNumberC = tempGuessNumber % 10;

tempGuessNumber /= 10;

int guessNumberB = tempGuessNumber % 10;

tempGuessNumber /= 10;

int guessNumberA = tempGuessNumber % 10;

// найти bulls

if (currentA == guessNumberA) {

currentBulls++;

currentA = -1;

guessNumberA = -2;

}

if (currentB == guessNumberB) {

currentBulls++;

currentB = -1;

guessNumberB = -2;

}

if (currentC == guessNumberC) {

currentBulls++;

currentC = -1;

guessNumberC = -2;

}

if (currentD == guessNumberD) {

currentBulls++;

currentD = -1;

guessNumberD = -2;

}

// выбор A

if (currentA == guessNumberB) {

currentCows++;

currentA = -1;

guessNumberB = -2;

}

if (currentA == guessNumberC) {

currentCows++;

currentA = -1;

guessNumberC = -2;

}

if (currentA == guessNumberD) {

currentCows++;

currentA = -1;

guessNumberD = -2;

}

// выбор B

if (currentB == guessNumberA) {

currentCows++;

currentB = -1;

guessNumberA = -2;

}

if (currentB == guessNumberC) {

currentCows++;

currentB = -1;

guessNumberC = -2;

}

if (currentB == guessNumberD) {

currentCows++;

currentB = -1;

guessNumberD = -2;

}

// выбор C

if (currentC == guessNumberA) {

currentCows++;

currentC = -1;

guessNumberA = -2;

}

if (currentC == guessNumberB) {

currentCows++;

currentC = -1;

guessNumberB = -2;

}

if (currentC == guessNumberD) {

currentCows++;

currentC = -1;

guessNumberD = -2;

}

// выбор D

if (currentD == guessNumberA) {

currentCows++;

currentD = -1;

guessNumberA = -2;

}

if (currentD == guessNumberB) {

currentCows++;

currentD = -1;

guessNumberB = -2;

}

if (currentD == guessNumberC) {

currentCows++;

currentD = -1;

guessNumberC = -2;

}

if (currentA == guessNumberB) {

currentCows++;

currentA = -1;

guessNumberB = -2;

}

if (currentA == guessNumberC) {

currentCows++;

currentA = -1;

guessNumberC = -2;

}

if (currentA == guessNumberD) {

currentCows++;

currentA = -1;

guessNumberD = -2;

}

if (currentBulls == bulls && currentCows == cows) {

arr.add(i);

}

}

}

return arr;

}

}

package main;

import java.util.ArrayList;

public class GenerateArray {

private ArrayList<Integer> arrayAll = new ArrayList();

public ArrayList<Integer> getGenerateArray() {

for (int i = 1111; i <= 6666; i++) {

if (i % 10 != 0 && i % 10 != 7 && i % 10 != 8 && i % 10 != 9 &&

(i / 10) % 10 != 0 && (i / 10) % 10 != 7 && (i / 10) % 10 != 8 && (i / 10) % 10 != 9 &&

(i / 100) % 10 != 0 && (i / 100) % 10 != 7 && (i / 100) % 10 != 8 && (i / 100) % 10 != 9) {

arrayAll.add(i);

}

}

return arrayAll;

}

}

package main;

import java.util.ArrayList;

import java.util.List;

public class ArrayTemp {

public List getArrayTemp(int bulls, int cows, int guessNumber) {

// ОТСОРТИРОВАЛИ И СОХРАНИЛИ В tempArray

SortArray arraySortTemp = new SortArray();

ArrayList<Integer> tempArray = arraySortTemp.getSortArray(bulls, cows, guessNumber);

// СГЕНЕРИРОВАЛИ ВСЕ ЗНАЧЕНИЯ ОТ 1111 ДО 6666, СОХРАНИЛИ В КОЛЛЕКЦИЮ arrayGeneration И УДАЛИЛИ ИЗ НЕЁ ОТСОРТИРОВАННЫЕ ЗНАЧЕНИЯ tempArray

GenerateArray arrayGen = new GenerateArray();

ArrayList<Integer> arrayGeneration = arrayGen.getGenerateArray();

arrayGeneration.removeAll(tempArray);

return arrayGeneration;

}

}