Факультет: Механико-Математический

Кафедра: Математическое и компьютерное моделирование

Выполнили: Раимжанов Д.Р.

Кадырова З.В.

Проверил: Кудайбергенов А.К.

**Оглавление**

**1.** [**Описание проекта**](#Описание_проекта)

**2.** [**Основные методы**](#Основные_методы)

**3.** [**Описание библиотек**](#Описание_библиотек)

**4.** [**Описание основных компонентов**](#Описание_основных_компонентов)

**5.** [**Реализация приложения**](#Реализация_приложения)

**6.** [**Описание достигнутых результатов**](#Описание_достигнутых_результатов)

**7.** [**Заключение**](#Заключение)

**8.** [**Ссылка на видео-защиту**](#Ссылка_на_видеозащиту)

**9.** [**Ссылки на видео-обсуждения**](#Ссылки_на_видеообсуждения)

**10.** [**Приложение с кодом**](#Приложение_с_кодом)

1. **Описание проекта**

По данному проекту нами было разработано приложение-викторина, где пользователю необходимо ответить на 20 вопросов, каждый из которых имеет по 4 варианта ответа. Вопросы, используемые в викторине, направлены на знание основных определений программного языка Java.

1. **Основные методы, используемые при разработке приложения**

Приложение было разработано с помощью методов библиотек **Swing** и **AWT**.

**Swing** отвечает за всю структуру приложения, а именно:

- Создание окна (JFrame)

- Создание панелей, добавление их в окно, их перерисовка (JPanel)

- Метки (JLabel) и кнопки (JButton, JRadioButton)

- Срисовка фона через JLabel и добавление элементов в нее

- Размер и расположение окна, текстов, кнопок

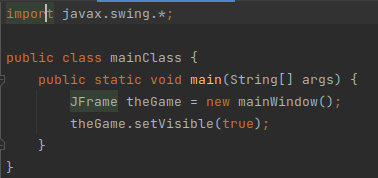
**AWT** играет роль слушателя событий, а именно:

- нажатие JButton – MouseListener

- нажатие (или точнее выбор) JRadioButton – ItemListener

1. **Описание библиотек и классов, используемых для реализации приложения**

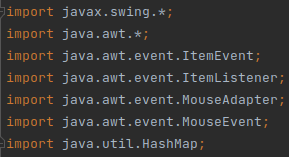
Код приложения разбит на два класса: *mainClass* и *mainWindow*.

*mainClass* создает окно и запускает его через *mainWindow*: **

*mainWindow* у нас по счету второй класс, через конструктор которого и выполняется работа самого приложения:



Как уже было сказано ранее, здесь используются две библиотеки, *Swing* и *AWT*, а также дополнительно была использована библиотека *HashMap*:



И у *AWT* (Abstract Window Toolkit), и у *Swing* почти одинаковое определение - обе являются библиотеками для создания графического интерфейса (окна).Но *AWT* является первой версией такой библиотеки, в то время как Swing была разработана как раз таки на базе библиотеки *AWT*, только уже более удобным в использовании. *Swing* заменила большинство компонентов *AWT* своими. В сравнении двух библиотек можно привести некоторые плюсы Swing:

* богатый набор интерфейсных примитивов;
* настраивающийся внешний вид на различных платформах (look and feel);
* раздельная архитектура модель-вид (model-view);
* встроенная поддержка HTML.

Также компоненты *Swing* можно отличить от компонентов *AWT* благодаря приставке “J” в начале названия компонента. Это можно было заметить в разделе под номером 2.

1. **Описание основных компонентов и событий, используемых при разработке** **графического интерфейса приложения**

Ниже поочередно будут приведены все компоненты, которые были использованы в процессе написания кода, а также описание того, для чего они были использованы:

- ***JFrame*** - для создания окна.

- ***JPanel*** - для добавления и управления всеми элементами приложения. Без панели это представляется невозможным. Из-за этого панель используется для каждой страницы приложения. Например, для главного меню - *mainPanel*, для страницы с правилами - *rulePanel*, для страницы с самой викториной - *gamePanel* и так далее. Из-за этого *JPanel* является самым часто используемым компонентом *Swing*.

- ***JTabbedPane*** - одно из важных составляющих приложения. Панель с вкладками использовалась в странице с игрой (*gamePanel*) для вывода вопросов. Одна вкладка - один вопрос.

- ***JLabel*** - метка для работы с текстом. Она использовалась для добавления вопросов викторины, их вариантов ответа и любых других текстов, которые должен увидеть пользователь. Здесь надо отметить один из плюсов *Swing*, который был сказан ранее, а именно поддержка HTML. Именно благодаря этому получилось регулировать размер текста и перенос на следующую строку в случае нехватки пространства на странице. HTML использовалась во всех текстах, которые встречаются в приложении.

- ***JButton*** - обычная кнопка для выполнения основных действий, таких как - “Играть”, “Правила”, “Выйти”, “Продолжить”, “Ответить”, “Назад”.

- ***JRadioButton*** - кнопка переключатель. Использовалась для выбора варианта ответа. Чтобы варианты ответа были взаимоисключающими (предполагается выбор лишь одного варианта ответа), был использован компонент *ButtonGroup*.

- ***ImageIcon*** - название говорит само за себя. Использовалась для добавления иконок в кнопки, а также вся работа с этими иконками и фотографиями велась через *ImageIcon*.

- ***MouseListener*** - для считывания событий от мышки. В нашем случае использовался только один из ее методов: *public void mouseClicked (MouseEvent event)* - выполнен щелчок мышкой на наблюдаемом объекте. Использовалась в таких кнопках, как - “Играть”, “Правила”, “Выйти”, “Продолжить”, “Ответить”, “Назад”. Например, при нажатии кнопки “Правила”, мы переходим в страницу с описанием правил викторины.

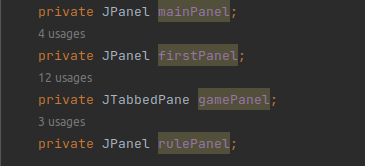
- ***ItemListener*** - для считывания изменений наблюдаемого элемента. В нашем случае состояние вариантов ответа (нажата ли кнопка), которые были созданы через *JRadioButton*. Этот слушатель имеет лишь один метод: *public void itemStateChanged (ItemEvent event).*

- ***HashMap*** - был использован для работы с вопросами и их ответами для викторины.

1. **Реализация приложения**

* Для начала мы создали главное окно с помощью *JFrame* и вызываем *mainWindow*, где происходит все основное действие.
* В *mainWindow* для начала мы создали все нужные глобальные переменные:

**Панели:**



Названия панелей соответствуют целям, для которых они были созданы.

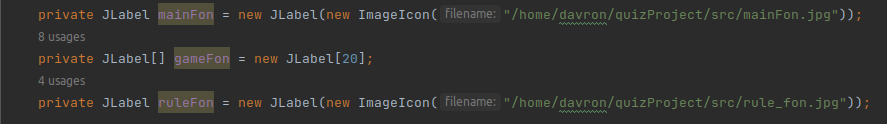
**mainPanel** — это главная панель, на которой рисуется все остальное. Поэтому в отношении *mainPanel* прописываем *setContentPane* (метод, с помощью которого мы указываем компьютеру, какую панель показать в окне (*JFrame*)).

**firstPanel** — это первая страница, которую видит пользователь. Здесь находятся 3 кнопки: “Играть”, “Правила”, “Выйти”.

**rulePanel** — это панель, где описываются правила викторины. Имеет кнопку “Назад” для выхода в *firstPanel*.

**gamePanel** — это панель с вкладками для прохождения викторины. Одна вкладка - один вопрос. Имеет две кнопки: “Продолжить” - чтобы перейти на следующий вопрос и “Ответить” - чтобы ответить на текущий вопрос.

**Метки для фона :**



Здесь также названия соответствуют целям, для которых они были созданы.

*firstPanel* - *mainFon*.

*rulePanel* - *ruleFon*.

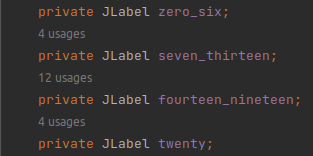
*gamePanel* - *gameFon*. Фонов для игры 20, так как вопросов в викторине 20. То есть, для каждого вопроса рисуется свой фон, со своим панелью. Об этом и о том, как рисуется фон будет чуть позже.

**Иконки:**



Также названия соответствуют целям, для которых они были созданы.

**Метки для текстов:**



При окончании викторины, в зависимости от того сколько баллов пользователь набрал, создается панель с результатом и разными текстами. Название соответствует набранному баллу - *zero\_six* для вывода текста с результатом, в случае если пользователь набрал от 0 до 6 баллов и т. д.

**Несколько важных переменных:**



*i, j* для итерационного цикла, *for, count* - для подсчета баллов, *lifе* - для счета количества жизни (вначале дается 3 жизни в соответствии с картинкой).

* Теперь заходим в конструктор *mainWindow* и делаем несколько первоначальных действий:

- даем название окну, устанавливаем размер окна, и чтобы окно закрывалось при нажатии на “х”.



- устанавливаем менеджер размещения для *mainPanel* *BorderLayout*\* и указываем ее как *setContentPane*.

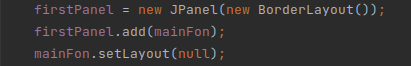




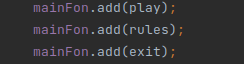
\* - Во всех 4 основных панелях установлен менеджер размещения *BorderLayout*. *BorderLayout* - менеджер размещения, который разделяет панель на 5 областей: верхняя, нижняя, левая, правая, центральная. Если при добавлении элементов в панель с таким менеджером размещения указать ЦЕНТР или не указывать совсем, то добавленный элемент займет все пространство панели. *Далее в реализации будут пропущены эти объяснения.*

* Первая страница (*firstPanel*) или страница входа в игру

Здесь сначала остановимся на том, как мы рисуем фон. Для того чтобы на фоне у нас была картинка, мы сначала создаем глобальную метку с нужной нам картинкой, как в метках для фона. После мы добавляем ее в нужную нам панель и указываем менеджер размещения как: *null*.



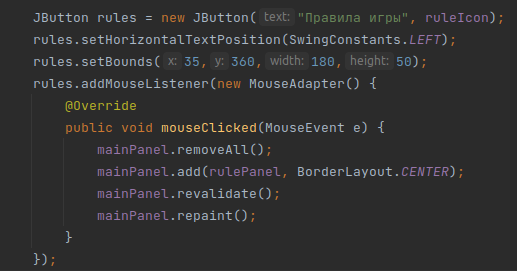
Далее, все что нужно показать на странице, мы добавляем в метку с фоновой картинкой, а не в панель:



**Срисовка фоновой картинки во всем приложении была реализована таким методом\***

**\****далее в реализации будут пропущены эти объяснения.*

В первой странице как уже ранее было описано имеются три кнопки. Покажем, как мы добавили одну из них (остальные кнопки будут оставлены ниже), к примеру кнопка - “Правила игры”:



Создаем кнопку, сразу задав название и иконку. После ставим текст слева от иконки. Задаем размер и расположение для кнопки, т. к. в метке с фоновой картинкой менеджер размещения у нас null, поэтому нужно указывать расположение. После добавляем для кнопки слушатель *MouseListener* с методом *MouseClicked*. Здесь после нажатия происходит одно из важных действий приложения:

- Сначала удаляем все элементы с *mainPanel*

- Добавляем панель с правилами, ведь мы нажимаем кнопку “Правила игры”

- Методы *revalidate* и *repaint* используются, когда мы добавляем в панель новый элемент, удалив перед этим старые.

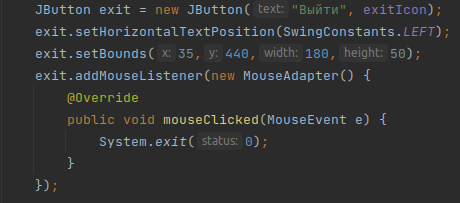
**Все кнопки в приложении созданы и добавлены по этому методу**

**В приложении, во всех случаях, когда нужно показать другую панель, используется такой метод срисовки\***

**\****далее в реализации будут пропущены эти объяснения.*

Остальные две кнопки:



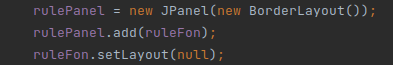


Добавляем наш *firstPanel* в *mainPanel*, так как *firstPanel* показывается сразу после запуска программы, остальные панели мы не добавляем в *mainPanel*, так как к ним переходим через кнопки: к панели с игрой через «Играть», к панели с правилами через «Правила игры:

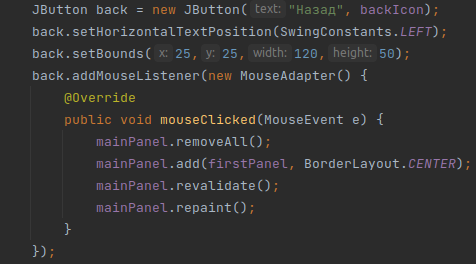


* Страница с правилами (*rulePanel*)

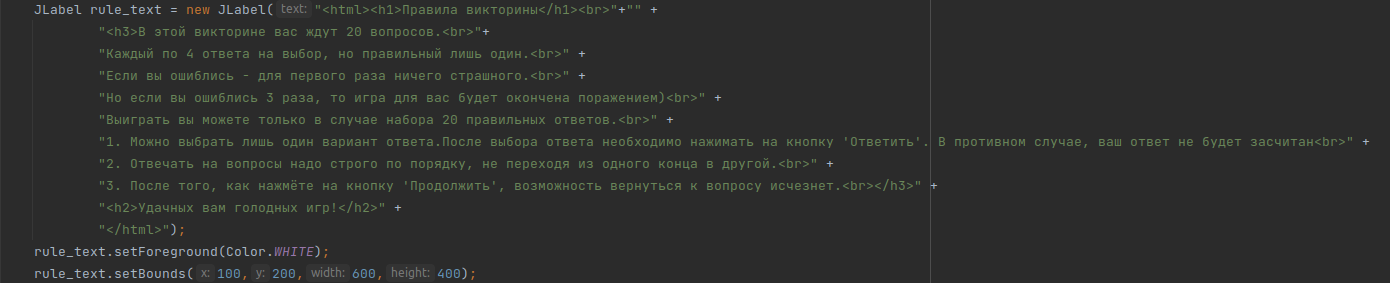
Устанавливаем для панели менеджер размещения, добавляем фон :



Кнопка “Назад”:



Метка с правилами:



Добавляем их в фон:



* Наша “База данных” вопросов и их ответов

Для вопросов и их правильных ответов был использован *HashMap*. Такой выбор объясняется легко – H­ashMap имеет некоторые удобные методы (put, get). Подробнее о них будет описано ниже, при объяснении страницы с игрой.





И т. д.

Для вариантов ответа использован обычный строковой двумерный массив:



И т. д.

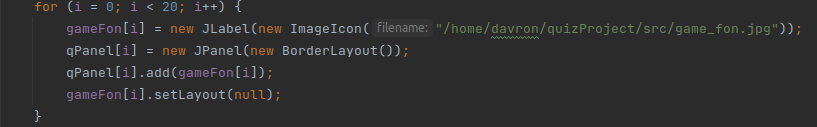
* Страница с игрой (*gamePanel*)

Сначала разбираемся с тем, где и как будут показываться наши вкладки с вопросами в *gamePanel*



То есть, вкладки будут показываться слева, без прокрутки. Также, мы создали массив панелей размером 20, по 1 - для каждого вопроса.

Теперь для каждого вопроса рисуем фон и добавляем его в наш массив панелей:

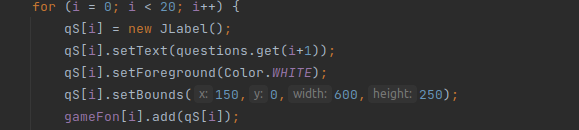


Проходим дальше, и теперь нам нужно создать массив меток размером 20. В каждом из них будет хранится текст вопроса:



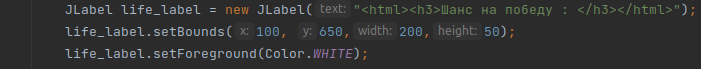
И здесь мы также заранее создали *ButtonGroup,* чтобы сделать варианты ответа взаимоисключающими.

**Теперь заходим в основной итерационный цикл:**



Сперва, мы добавляем текст вопроса в заранее созданные метки, через индексы. Вопросы берем с заранее созданного *HashMap* массива с помощью метода “get”. Меняем цвет текста на белый с помощью *setForeground*. Определяем размер и расположение, добавляем в наш фон.

Идем дальше, и мы создаем метку со счетчиком жизни:



Создаем кнопку “Ответить”, после которой происходят основные действия:

-считывание баллов;

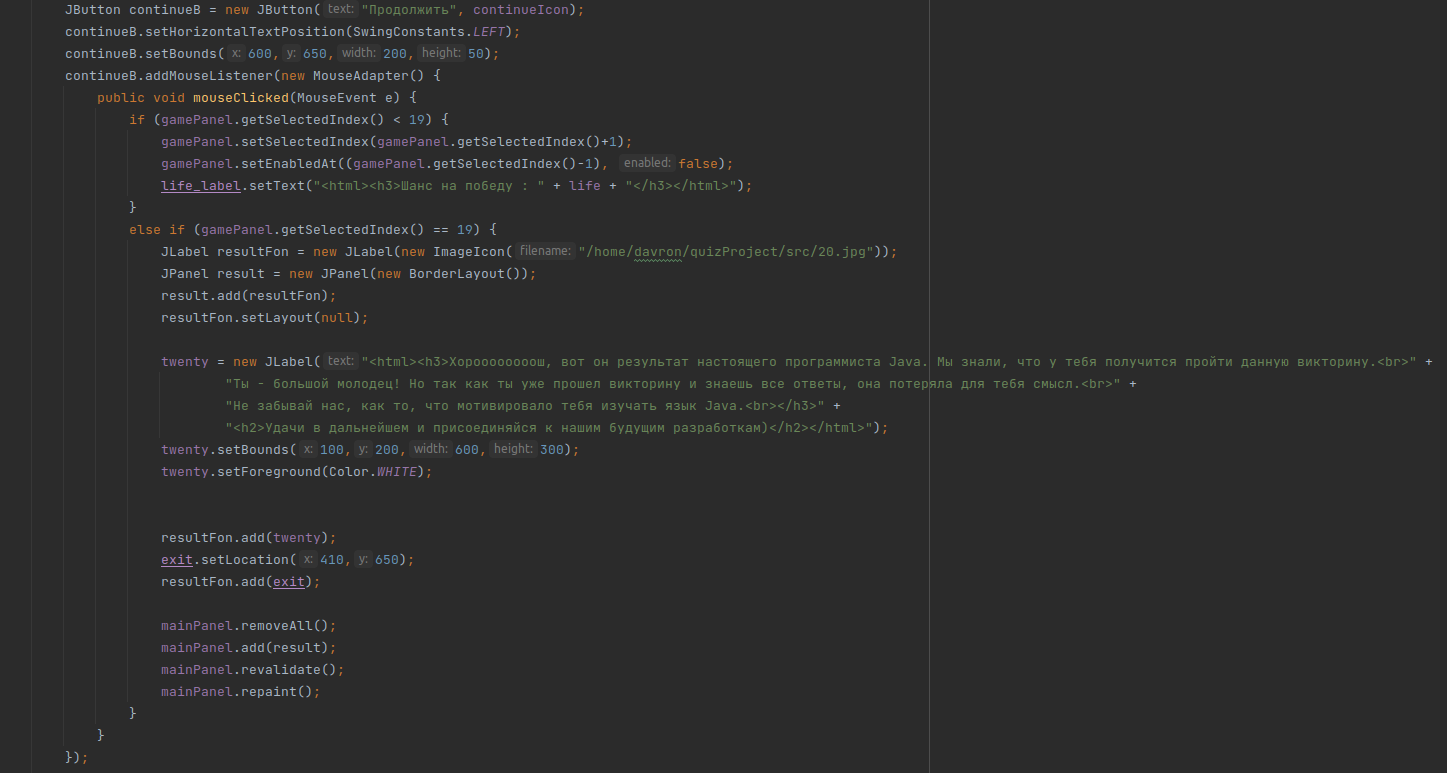
-блокирование кнопок;

-считывание количества оставшихся жизней, при обнулении счетчика, игра заканчивается.

**Но мы сейчас только создаем кнопку, указав для нее размер с расположением, слушатель с действием будет добавлен чуть позже.**



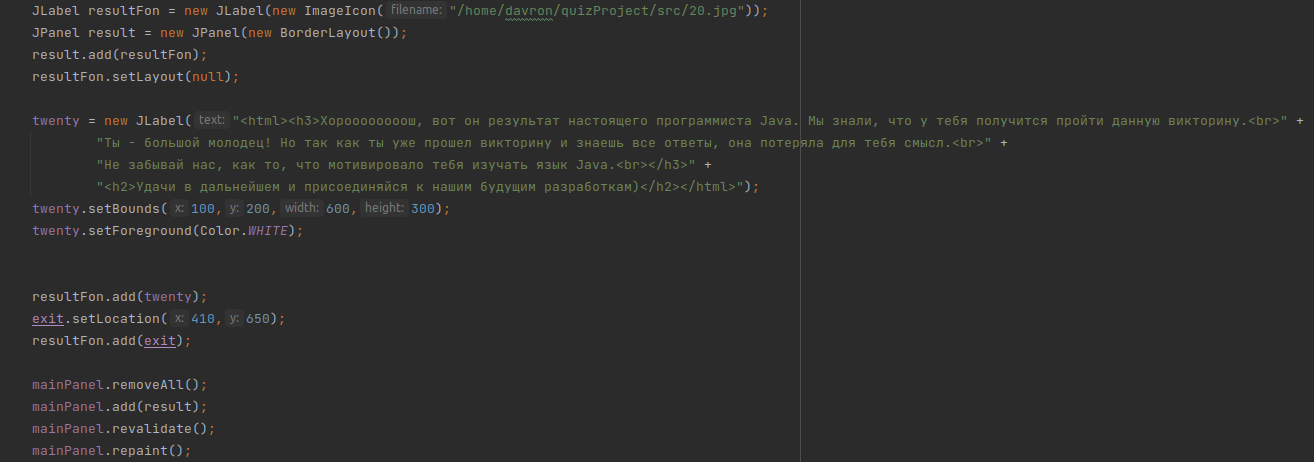
Создаем кнопку “Продолжить”, основная задача которой переходить на следующий вопрос и блокировать предыдущий:



Здесь проверка индекса нынешней вкладки нужна, чтобы не допустить переход на следующую вкладку, если эта последняя. Поэтому, первое условие - если индекс нынешней вкладки панели меньше 19 (индекс идет с 0, а не с 1), то переходим на следующий вопрос (вкладку) и блокируем предыдущий :

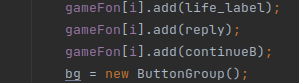


Ну а если мы уже на последней вкладке:



В случае нажатия кнопки «Продолжить» в последнем вопросе, мы удаляем нынешнюю панель, и создаем новую, в которой покажем результат, а именно тот, в котором пользователь выиграл викторину (ответил на все 20 вопросов правильно). Это происходит потому, что другие варианты исхода викторины будут описаны в слушателе кнопки “Ответить”, чуть ниже.

Теперь все эти элементы добавляем в наш фон и также определяем *ButtonGroup*, так как она вот-вот понадобится:



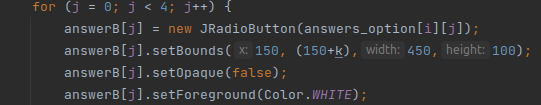
Дело осталось за вариантами ответа!

Мы все еще находимся в главной итерации от 0 до 20.

Для вариантов ответа нам понадобится массив кнопок *JRadioButton* размером 4, так как у нас 4 варианта ответа на каждый вопрос и одна целочисленная переменная для расположения.

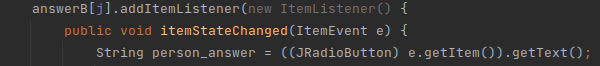


Создаем итерацию *for* от 0 до 4. Определяем *JRadioButton*:



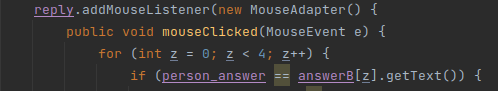
Здесь незнакомым является лишь *setOpaque*. Этот метод делает фон кнопки прозрачным. Текст для вариантов также берем с заранее созданного двумерного массива.

Создаем слушатель и входим в него:



Сначала мы переопределяем выбранный вариант в кнопку *JRadioButton*, затем методом *getText* получаем текст выбранного варианта, чтобы далее проверить правильный ли это вариант или нет.

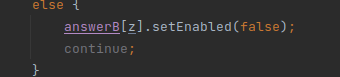
Вот теперь мы создаем для кнопки “Ответить” слушатель нажатия кнопки, чтобы именно при нажатии считывались баллы:



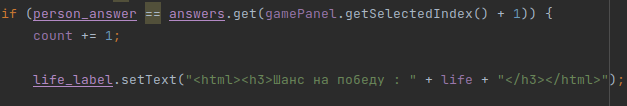
*Важно: Swing устроена так что, все наши действия с созданиями и определениями кнопок и каких-либо элементов в главной итерации полностью проходит, игнорируя слушатели вариантов ответа и слушатель кнопки “Ответить”. Так как внутрь слушателя наша программа заходит лишь тогда и только тогда, когда мы делаем эти действия. Соответственно, внутри слушателя мы уже не можем обратиться к индексу “і” главной итерации и к индексу “j”, так как главная итерация уже закончена. Поэтому попытка обратиться к массиву кнопок варианта ответа через “j” приведет к ошибке.*

А блокировать все кнопки нам все-таки нужно. Поэтому, мы поступим чуть хитрее:

Мы создаем новую итерацию *for* от 0 до 4 и ставим условие - если ранее полученный текст выбранного ответа совпадает с текстом кнопки с индексом “z”, то мы нашли вариант, который выбрал наш пользователь, а значит мы можем действовать дальше с этой кнопкой. А остальные кнопки мы блокируем:

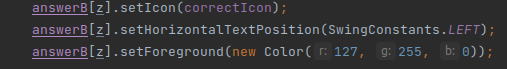


Идем дальше. Теперь ставим условие, чтобы понять правильно ли ответил пользователь, и в таком случае производим следующее действие:

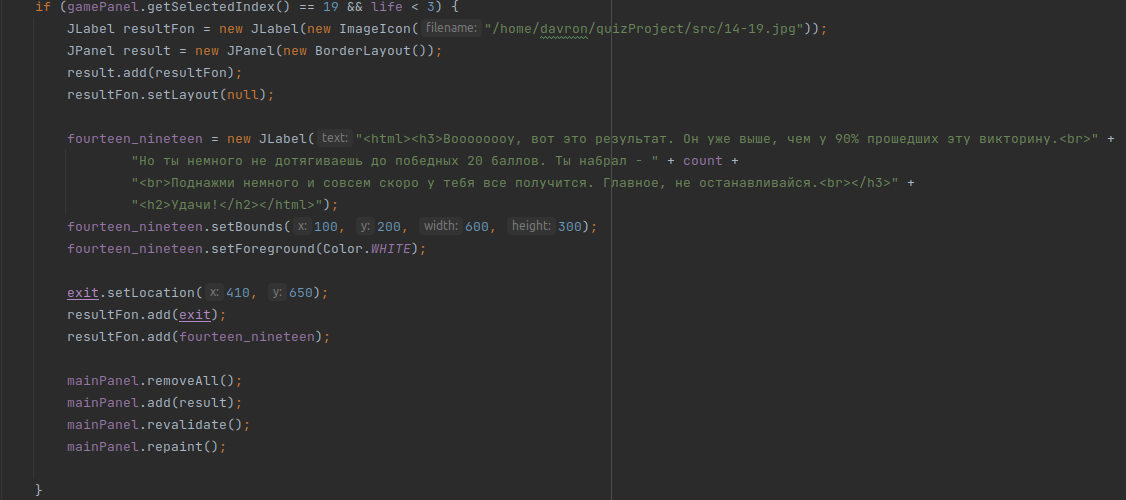


Считаем количество баллов в переменную *cont* и показываем сколько жизней осталось у пользователя.

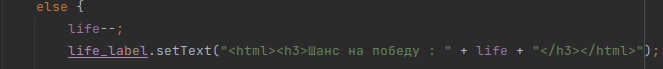
Меняем цвет текста выбранного ответа на зеленый и ставим иконку галочки, ведь пользователь ответил правильно:



И теперь одно важное условие: если мы находимся на последнем вопросе и пользователь ответил правильно, НО до этого он уже ошибался один или два раза, то это означает что он набрал от 14 до 19, поэтому в соответствии с этим выводим результат, не дав пользователю нажать кнопку “Продолжить”:

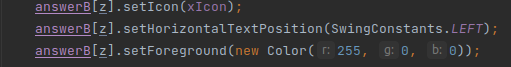


Далее. Случай если пользователь ответил неправильно:



Отнимаем одну жизнь и показываем сколько жизней у него осталось.

Так как пользователь ответил неправильно меняем цвет текста на красный и ставим иконку “х”:



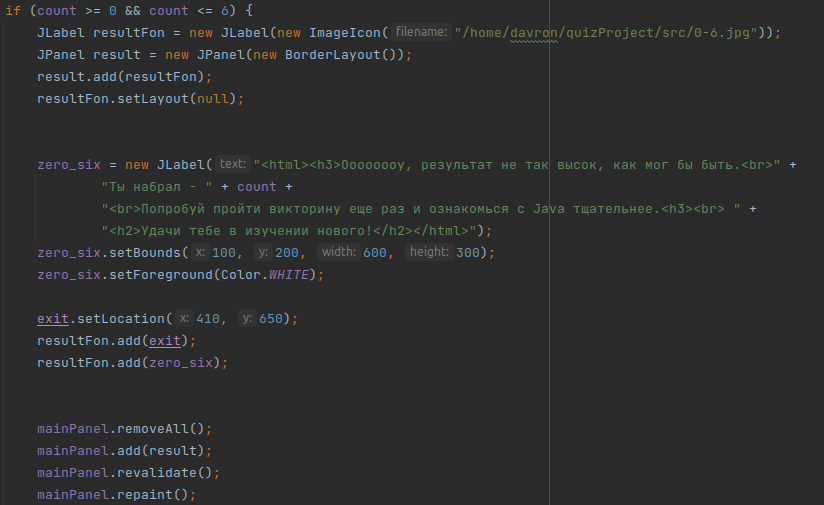
Идем далее и ставим условие, в случае, когда пользователь отвечает неправильно и у него уже не остаются жизни. Внутри нее мы еще раз ставим условие, чтобы понять сколько баллов пользователь набрал и, исходя из него, показываем разный результат:



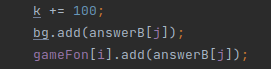




Почему здесь нет условия когда равно 20 мы ранее объясняли. А сейчас покажем лишь вариант от 0 до 6, так как вывод результата у всех вариантов исхода идентичны:



Теперь мы закрываем нашу итерацию с вариантами ответа:



*bg* – *ButtonGroup* которую мы создали и определили ранее.

И вот уже можно закрывать главную итерацию, сделав последнее действие:



Добавляем в нашу *TabbedPane* с названием наши панели с вопросами.

**На этом реализация викторины окончена!**

1. **Описание достигнутых результатов проекта**

В результате проектной работы у нас на руках есть работающее приложение-викторина с 20 вопросами и 4 вариантами ответа для каждого.

Возможности нашего приложения:

- считывание баллов;

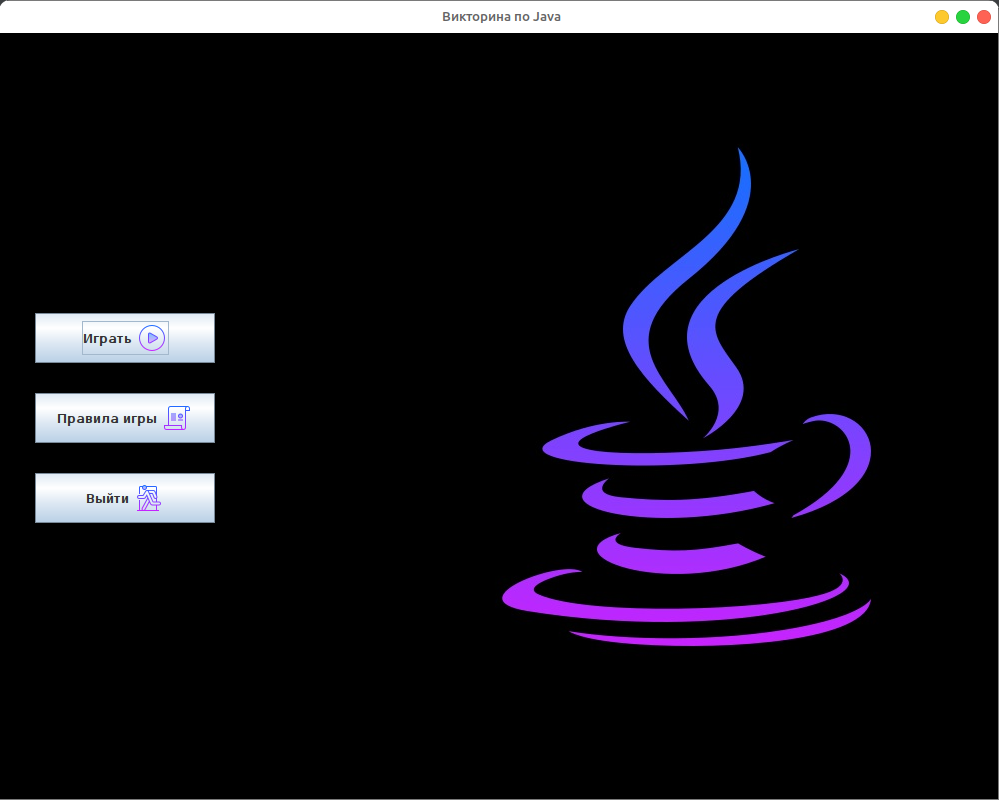
- считывание количества жизней (вначале дается 3);

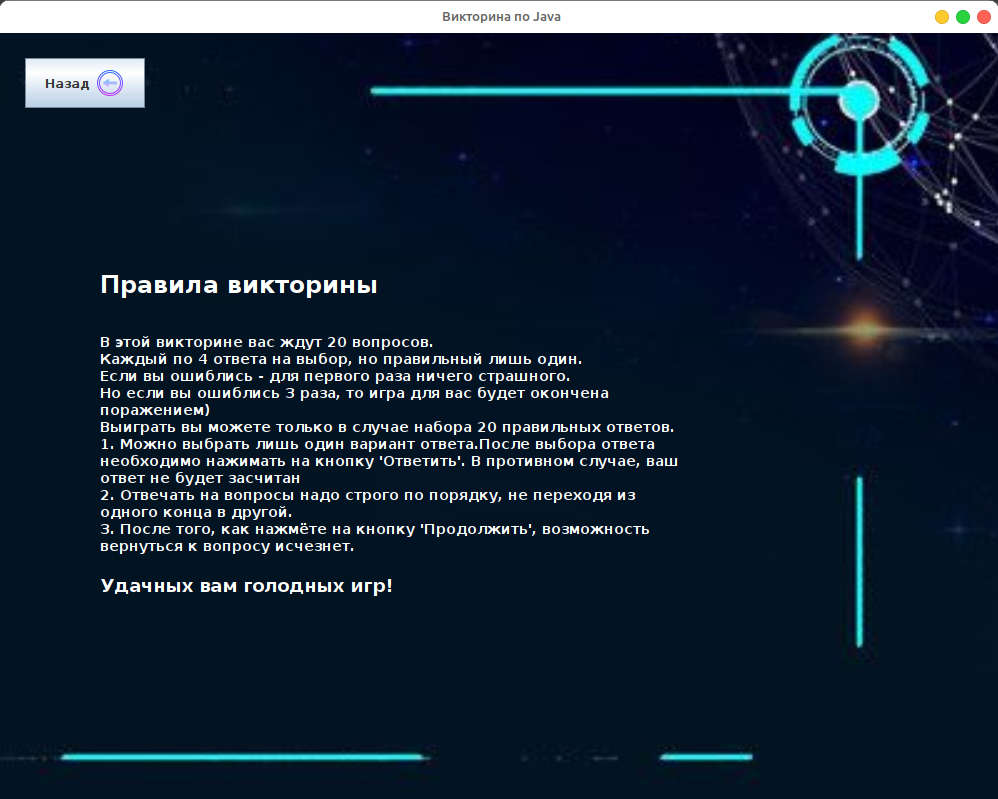
- показ разного результата в зависимости от того, как пользователь прошел викторину;

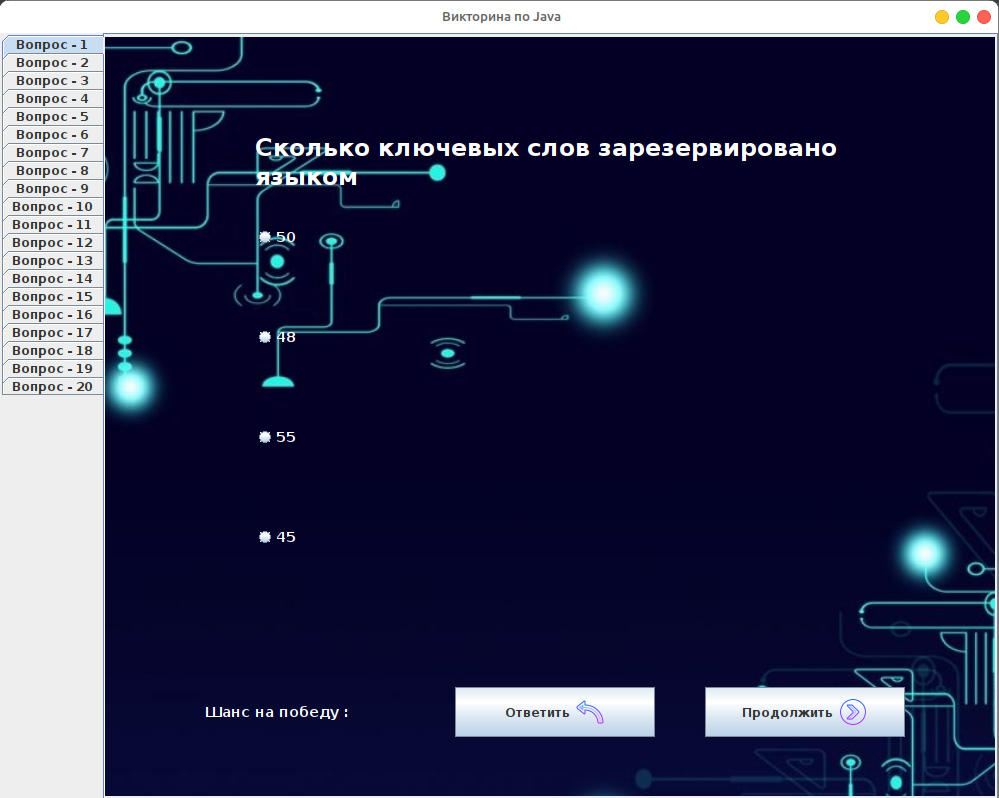
- после ответа блокирование остальных кнопок;

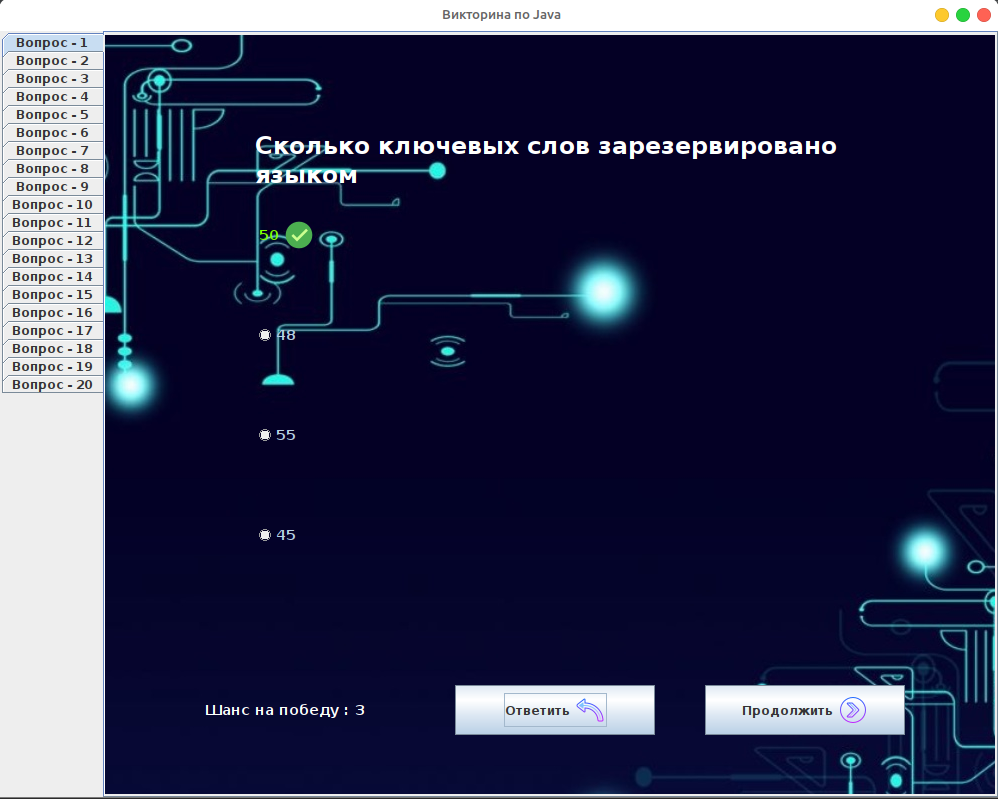
- после перехода на следующий вопрос блокирование предыдущего вопроса;

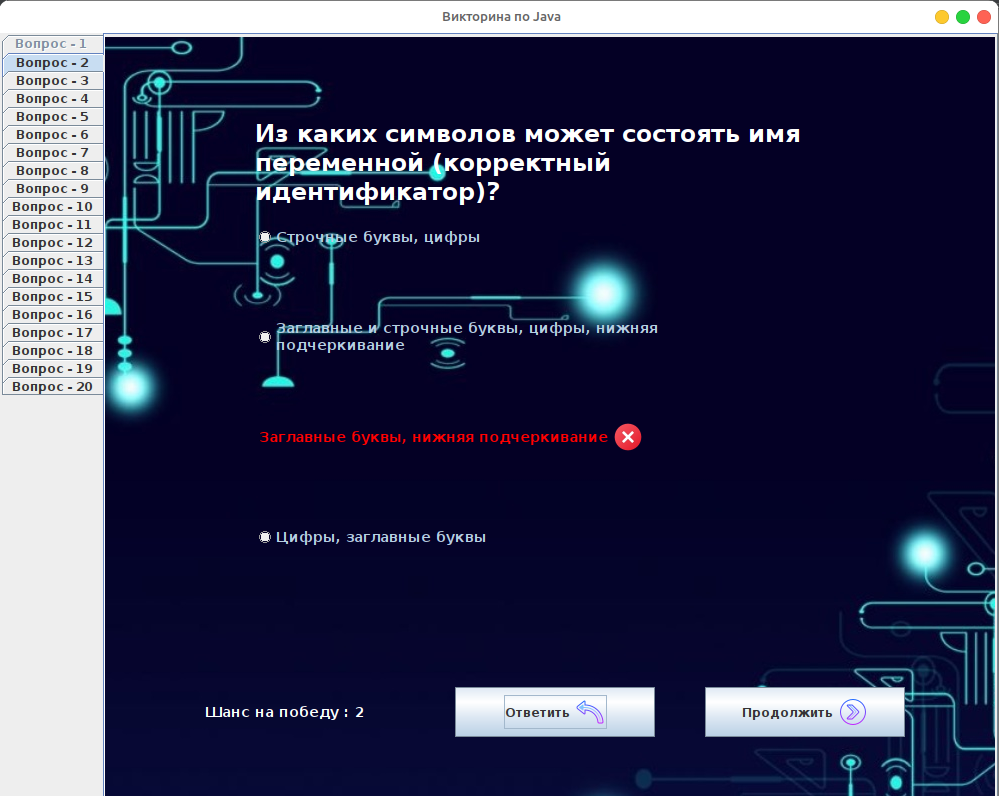
**Скрины самой викторины:**

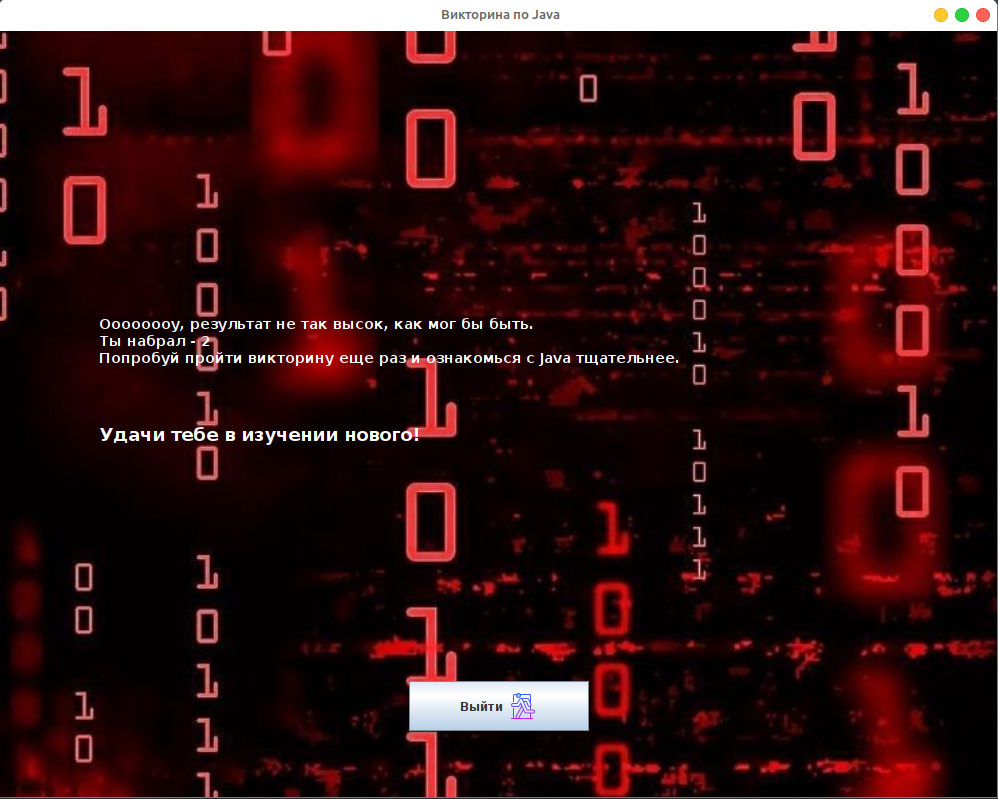




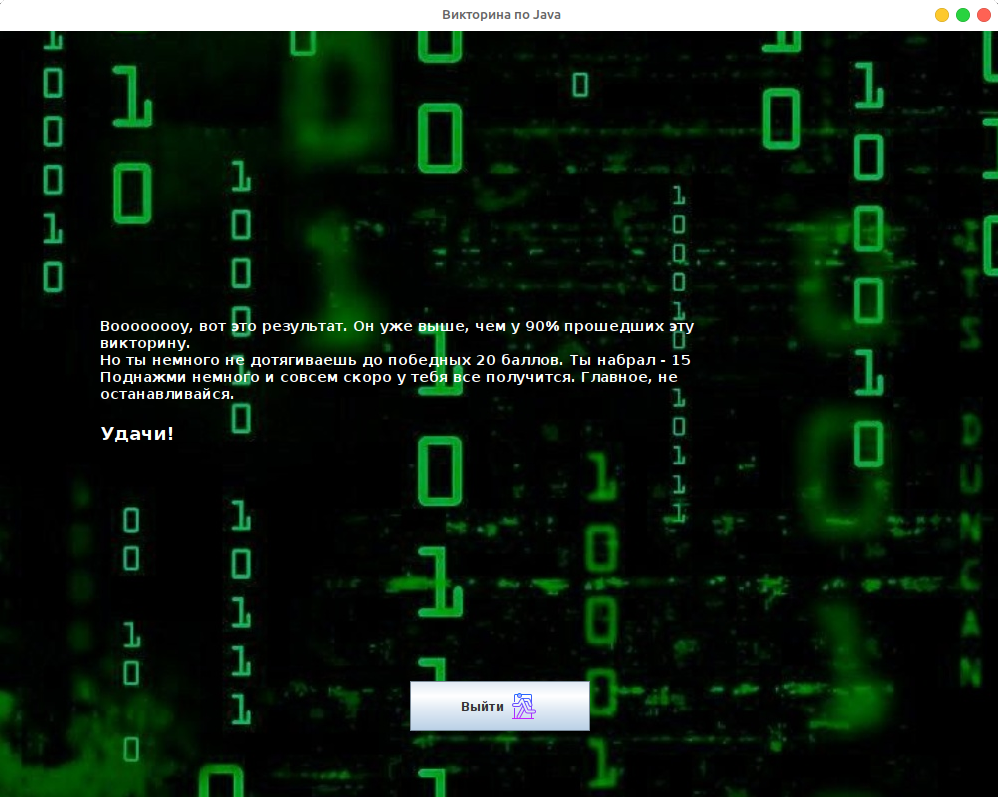


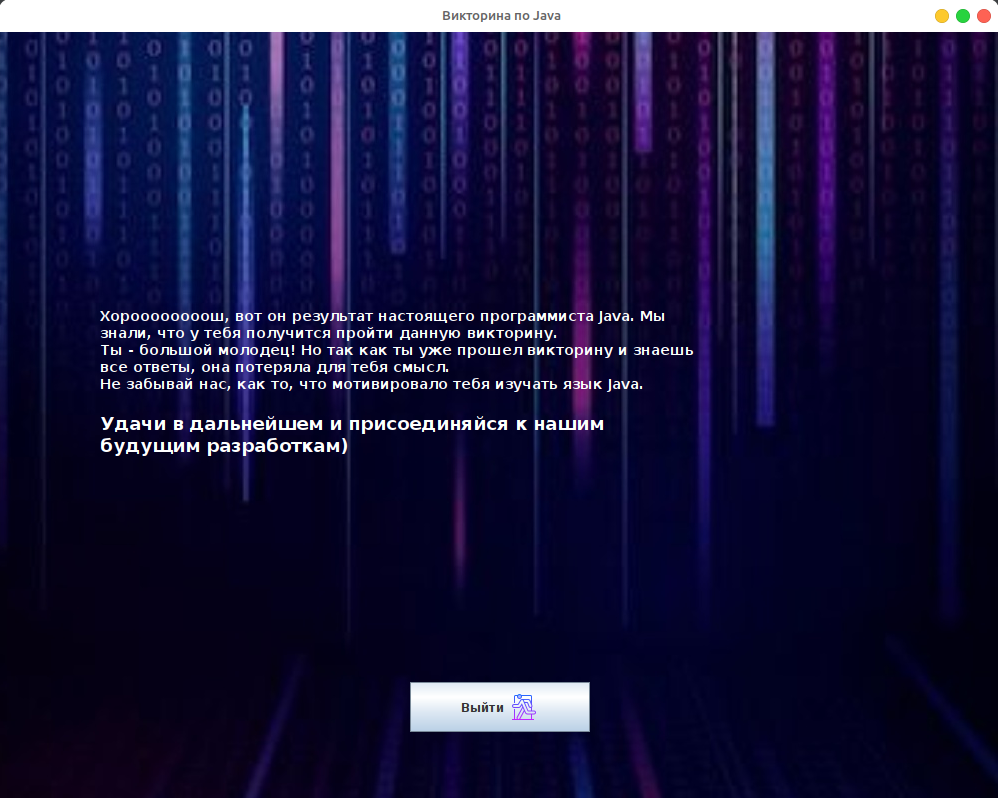












1. **Заключение**

Работа над совместным проектом была начата примерно на 10-ой недели и была закончена прямо к концу дедлайна. Что это означает?

Это означает, что мы столкнулись с массой различных проблем, среди которых можно упомянуть следующие:

- как рисовать фон?

- как считать баллы?

- как сделать так, чтобы блокировались все кнопки?

- как переходить с одной страницы на другую при нажатии кнопки?

- как менять вид кнопок после ответа?

- как показать разный результат?

И еще много-много мелких проблем встречались по ходу проекта. Все они были решены постепенно с осознанием того, как работает библиотека Swing. Да, викторина неидеальна, ведь изначально мы хотели видеть в ней еще немало функций, но к сожалению либо не смогли реализовать, либо просто не успели. Но все же, мы довольны полученным результатом и тем, чему мы смогли научиться. И сейчас, научившись таким вещам, мы уверенно можем сказать, что мы имеем базовые знания по Java и Swing, в частности.

Ссылка на видео-защиту:

Видео-защита <https://youtu.be/atRjYZRRPP4>

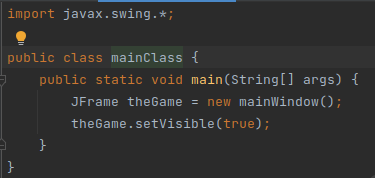
Ссылки на видео-обсуждения:

Видео-обсуждение №1 <https://youtu.be/jTAzmybmfpE>

Видео-обсуждение №2 <https://youtu.be/10ZjhHoYUac>

**Приложение с кодом:**

**MainClass:**



**MainWindow:**

