МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Кафедра програмного забезпечення автоматизованих систем

Курсовий проект

Автоматизована система перевірки зору “Eyesight-Checker”

з дисципліни

“Архітектура та проектування програмного забезпечення”

Пояснювальна записка

|  |  |
| --- | --- |
| ПРИЙНЯЛИ:  Викл. Метелап В. В.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка)  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ р.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Підпис) | ВИКОНАВ:  Ст. групи ПЗ-144  Качанов Б.В.  “\_\_\_”\_\_\_\_\_\_\_ \_\_\_\_\_\_ р.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Підпис) |

м. Черкаси

2017

ЗМІСТ

Зм.

Лист

№ документа

Підпис

Дата

Розроб.

Керівник

Керівник

.

Тулуб В.О.

Метелап В. В.

ЧДТУ.161453.007 ПЗ

ВЕБ-ДОДАТОК

«TIME MANAGEMENT»

Пояснювальна записка

Літ.

Лист

Листів

НД

**ФІТІС,**

кафедра ПЗАС, ПЗ-144

2

[ВСТУП 3](#_Toc485742848)

[1.ПОСТАНОВКА ЗАДАЧІ 4](#_Toc485742849)

[1.1 Функціональні вимоги 4](#_Toc485742850)

[1.2 Вимоги до технічних і програмних засобів 5](#_Toc485742851)

[2.ПРОЕКТУВАННЯ 6](#_Toc485742852)

[2.1 Розробка use-case сценаріїв 6](#_Toc485742853)

[2.2 Проектування бази даних 7](#_Toc485742854)

[3. РОЗРОБКА 9](#_Toc485742855)

[3.1 Логічна структура програми 9](#_Toc485742856)

[3.2 Опис підпрограм і класів 11](#_Toc485742857)

[3.3 Розробка UML діаграм логічного рівня 12](#_Toc485742858)

[3.4 Список аномальних ситуацій і реакцій програми 13](#_Toc485742859)

[ВИСНОВКИ 14](#_Toc485742860)

[ДОДАТКИ 15](#_Toc485742861)

[Додаток А. Скрипти об’єктів бази даних 15](#_Toc485742862)

[Додаток Б. Лістинг back end частини програми 21](#_Toc485742863)

[Додаток В. Інструкція користувачеві 30](#_Toc485742864)

[Додаток Г. Структура програми 48](#_Toc485742865)

[СПИСОК Використаних джерел 49](#_Toc485742866)

# ВСТУП

У наш час все-більше стають актуальними інструменти, що роблять життя простіше, за допомогою яких можна робити певні дії зручнішим способом. У цій пояснювальній записці буде описуватись інструмент, який являє собою додаток під назвою «Eyesight-Checker». Цей додаток дозволяє здійснювати перевірку зору за допомогою компьютера та отримати результати та рекомендації щодо стану зору користувача. Використовуючи цей додаток користувач матиме змогу перевірити стан свого зору не виходячи з дому.

Розробники цього проекту вважають, що в час, коли кожна людина проводить досить багато часу перед екранами компьютерів, необхідно систематично перевіряти свій зір, щоб одразу відвідати лікаря, якщо стан зору погіршиться.

Метою курсового проекту є полегшення процесу перевірки стану зору не виходичи з дому.

# 1.ПОСТАНОВКА ЗАДАЧІ

## 1.1 Функціональні вимоги

Курсовий проект являє собою прикладний додаток, що складається з інтерфейсу користувача та алгоритмів перевірки зору. В клієнтському інтерфейсі мають бути передбачені наступні функції:

* Вибір між режимом перевірки зору та ігровими режимами;
* Отримання даних про користувача (Ім’я, Прізвище, Вік);
* Інструкція;
* Перевірка зору за допомогою таблиці Сівцева;
* Ігрові режими;
* Отримання результатів тестування;

Додаток має бути кросплатформним та коректно працювати з усіми операційними системами (Windows, Unix).

## 1.2 Вимоги до технічних і програмних засобів

Для належного функціонування програми необхідно:

1. IBM-сумісний комп’ютер з 64-бітною архітектурою.
2. 1Gb оперативної пам’яті.
3. 500Mb вільного місця на жорсткому диску.
4. Клавіатура та миша.

1.3 Вибір програмних та технічних засобів

Для створення цього проекту використовувалась мова програмування Java та платформа JavaFX. Обрана середа розробки програм — Intellij IDEA. Також використовалась система контролю версій GIT. Для роботи з графікою використовувалась програма Adobe Photoshop CS5.

# 2.ПРОЕКТУВАННЯ ПРОГРАМИ

## 2.1. Оцінка задач

## 1. Оцінка алгоритмів тестування зору за допомогою таблиці Сівцева.

## 2. Масштабування таблиці Сівцева відповідно до дістанціі між екраном компьютера та очима користувача.

## 3. Вибір мови програмування та програмних засобів.

## 4. Зображення макету інтерфейсу користувача.

## 5. Розробка інтерфесу користувача.

## 6. Розробка режиму тестування зору.

## 7. Створення макету ігрових режимів.

## 8. Розробка ігрових режимів.

## 9. Тестування.

## 2.2. Сканы

**Тут будут сканы листов с макетами окон!**

# 3. РОЗРОБКА

## 3.1 Логічна структура програми

Розроблений проект на мові програмування Java складається з:

* Папки src, що буде містити всі файли вихідного коду;
* Файлу Eyesight\_checker.iml;

Папка src містить:

* Папки controllers, еуеsightChecker, в яких зберігаються файли з java-класами, що містять код програми;
* Папку img, яка містить зображення, що використовуються під час роботи програми;

На рис. 2 зображено загальну структуру проекту:

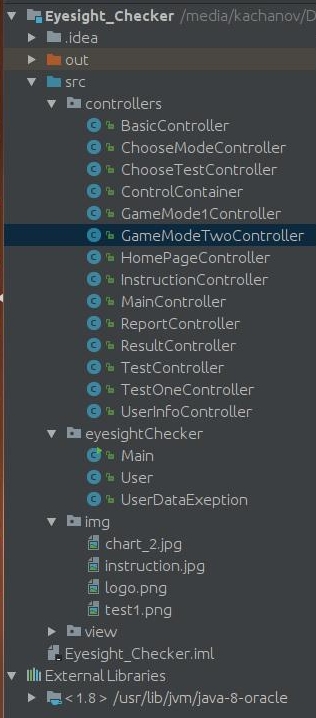


Рисунок 2. Загальна структура проекту.

## 3.2 Опис підпрограм і класів

Всередині пакета config знаходяться класи, що відповідають за зчитування файлів конфігурацій і налаштування програми; пакет контроллер містить контроллери для передачі даних на jsp сторінки перед віправкою їх користувачеві; пакет model містить entity-класи для роботи з базою даних; пакет dao містить інтерфейси та класи потрібні для реалізації шаблону програмування DAO (Data Access Object); пакет security містить класи, що відповідають за безпеку та захист даних (в моєму проекті це шифрування паролів користувачів), пакет service містить класи та інтерфейси, необхідні для організації зв’язку між dao та контролерами. Клас LibraryApplication містить метод main та метод для виклику конфігурації, цей клас запускає програму.

**ЭТУ СТРАНИЦУ НУЖНО ПЕРЕДЕЛАТЬ ПОД НАС!**

## 3.3 Список аномальних ситуацій і реакцій програми

1. В програмі передбачено недоступність деяких дій незареєстрованим користувачам.
2. Передбачено виведення повідомлень про помилки при невдалій реєстрації та авторизації.
3. Передбачено неможливість використання однакової електронної пошти (виводиться помилка про невдачу реєстрації).
4. Неможливо зареєструвати користувача не заповнивши всі поля (виводиться помилка про невдачу реєстрації).
5. При будь яких помилках при додаванні або оновленні записів бази даних автоматично виконуються відкат за допомогою механізму транзакцій, який використовується у фреймворку Hibernate.

**ХЗ нужно ли делать этот раздел.**

# ВИСНОВКИ

В ході виконання курсового проекту я закріпив теоретичні знання і розширив практичні навички з дисципліни «Архітектура та проектування програмного забезпечення». В цьому проекті я займався оцінкою поставленої задачі, проектування програми, моделювання інтерфейсу користувача та робота з графікою.

# ДОДАТКИ

## Додаток А. Лістинг програми

## 1. Клас Main

**package** eyesightChecker;

**import** javafx.application.Application;

**import** javafx.stage.Stage;

**import** controllers.MainController;

**public class** Main **extends** Application {

**private static** MainController *mainController*;

**private static** User *currentUser* = **new** User(); *//* ***TODO: 13.06.17 redo***

@Override

**public void** start(Stage primaryStage) **throws** Exception {

*mainController* = **new** MainController(primaryStage);

}

**public static void** main(String[] args) {

*launch*(args);

}

**public static** MainController getMainController() {

**return** *mainController*;

}

**public static** User getCurrentUser() {

**return** *currentUser*;

}

**public static void** setCurrentUser(User currentUser) {

Main.*currentUser* = currentUser;

}

}

## 2. Клас User

**package** eyesightChecker;

**public class** User {

**private** String **name**;

**private** String **surname**;

**private int age**;

**private int** answerCount;

**private int rightAnswerCount**;

**private int problemTableRow**;

**public** User(String name, String surname, **int** age, **int** answerCount,

**int** rightAnswerCount, **int** problemTableRow) {

**this**.**name** = name;

**this**.**surname** = surname;

**this**.**age** = age;

**this**.answerCount = answerCount;

**this**.**rightAnswerCount** = rightAnswerCount;

**this**.**problemTableRow** = problemTableRow;

}

**public** User(String name, String surname, **int** age) {

**this**.**name** = name;

**this**.**surname** = surname;

**this**.**age** = age;

}

**public** User() {

}

**public** String getName() {

**return name**;

}

**public void** setName(String name) {

**this**.**name** = name;

}

**public** String getSurname() {

**return surname**;

}

**public void** setSurname(String surname) {

**this**.**surname** = surname;

}

**public int** getAge() {

**return age**;

}

**public void** setAge(**int** age) {

**this**.**age** = age;

}

**public int** getAnswerCount() {

**return** answerCount;

}

**public void** setAnswerCount(**int** answerCount) {

**this**.answerCount = answerCount;

}

**public int** getRightAnswerCount() {

**return rightAnswerCount**;

}

**public void** setRightAnswerCount(**int** rightAnswerCount) {

**this**.**rightAnswerCount** = rightAnswerCount;

}

**public int** getProblemTableRow() {

**return problemTableRow**;

}

**public void** setProblemTableRow(**int** problemTableRow) {

**this**.**problemTableRow** = problemTableRow;

}

}

## 3. Клас ChooseTestController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXML;

**public class** ChooseModeController **extends** BasicController {

@FXML

**void** goToGameMode1() {

Main.*getMainController*().switchToStage(MainController.***GAME\_MODE\_ONE\_CONTAINER***);

}

@FXML

**void** goToGameModeTwo() {

Main.*getMainController*().switchToStage(MainController.***GAME\_MODE\_TWO\_CONTAINER***);

}

}

## 4. Клас ControlContainer

**package** controllers;

**import** javafx.scene.Parent;

**import** javafx.stage.Stage;

**public class** ControlContainer {

**private** Parent **parent**;

**private** Stage **stage**;

**private** BasicController **controller**;

**public** ControlContainer() {

}

**public** ControlContainer(Parent parent, Stage stage, BasicController controller) {

**this**.**parent** = parent;

**this**.**stage** = stage;

**this**.**controller** = controller;

}

**public** Parent getParent() {

**return parent**;

}

**public void** setParent(Parent parent) {

**this**.**parent** = parent;

}

**public** Stage getStage() {

**return stage**;

}

**public void** setStage(Stage stage) {

**this**.**stage** = stage;

}

**public** BasicController getController() {

**return controller**;

}

**public void** setController(BasicController controller) {

**this**.**controller** = controller;

}

}

## 5. Клас GameMode1Controller

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXML;

**import** javafx.geometry.HPos;

**import** javafx.scene.control.Alert;

**import** javafx.scene.control.Label;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.paint.Color;

**import** javafx.scene.paint.Paint;

**import** javafx.scene.shape.Rectangle;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Random;

**public class** GameMode1Controller **extends** BasicController{

@FXML

**private** GridPane **gpGameMode1**;

@FXML

**private** Label **lblTask**;

**private** List<Rectangle> **rectangles**;

**private** List<Color> **colors**;

**private** Integer **totalTargetRects**;

**private** Integer **totalCorrectAnswers**;

**private** Integer **selectedRectsCount**;

**private** Integer **colorId**;

**private** Color **targetColor**;

String[] **colorName** = {**"blue"**, **"red"**, **"green"**};

@FXML

**void** refresh() **throws** IOException {

Main.*getMainController*().externalInitGameModeOneStage();

Main.*getMainController*().switchToStage(MainController.***GAME\_MODE\_ONE\_CONTAINER***);

}

@FXML

**void** goToResult() **throws** IOException {

**if** (**totalTargetRects**.equals(**totalCorrectAnswers**) && **totalTargetRects**.equals(**selectedRectsCount**)) {

Main.*getMainController*().switchToStage(MainController.***RESULT\_CONTAINER***);

Main.*getMainController*().externalInitGameModeOneStage();

((ResultController)Main.*getMainController*().getControlContainers().

get(MainController.***RESULT\_CONTAINER***).getController()).getLblResult().

setText(**"Level complete!"**);

} **else** {

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle(**""**);

alert.setHeaderText(**null**);

alert.setContentText(**"Look for the other "** + **colorName**[**colorId**] + **" squares, please!"**);

alert.showAndWait();

}

}

@FXML

**private void** initialize() {

**totalCorrectAnswers** = 0;

**totalTargetRects** = 0;

**selectedRectsCount** = 0;

**rectangles** = **new** ArrayList<>();

**colors** = **new** ArrayList<>();

addColors();

**targetColor** = (Color) getRandomColor(**colors**);

**colorId** = (**targetColor** == Color.***BLUE***) ? 0 : (**targetColor** == Color.***RED***) ? 1 : 2;

**lblTask**.setText(**"Choose "** + **colorName**[**colorId**] + **" squares"**);

**for** (**int** i = 0; i < 4; i++) {

**for** (**int** j = 0; j < 3; j++) {

Rectangle rect = **new** Rectangle(0, 0, 72, 72);

rect.setFill(getRandomColor(**colors**));

**if** ((Color) rect.getFill() == **targetColor**) {

**totalTargetRects**++;

}

rect.setStroke(Color.***BLACK***);

**rectangles**.add(rect);

addRect(rect, i, j);

rect.setOnMousePressed(event -> {

System.***out***.println(**"mouse click detected! "** + event.getSource());

Rectangle rectangle = (Rectangle) event.getSource();

Color color = (Color) rectangle.getFill();

**if**(rect.getStroke() != Color.***AZURE***) {

**if** (color == **targetColor**)

**totalCorrectAnswers**++;

rectangle.setStroke(Color.***AZURE***);

**selectedRectsCount**++;

} **else** {

**if** (color == **targetColor**)

**totalCorrectAnswers**--;

rectangle.setStroke(Color.***BLACK***);

**selectedRectsCount**--;

}

});

}

}

}

**private** Paint getRandomColor(List<Color> colors) {

**return** colors.get(**new** Random().nextInt(3));

}

**private void** addColors() {

**colors**.add(Color.***BLUE***);

**colors**.add(Color.***RED***);

**colors**.add(Color.***GREEN***);

}

**private void** addRect(Rectangle rect, **int** colIndex, **int** rowIndex) {

**gpGameMode1**.add(rect, colIndex, rowIndex);

**gpGameMode1**.*setHalignment*(rect, HPos.***CENTER***);

}

**public** GridPane getGpGameMode1() {

**return gpGameMode1**;

}

**public** List<Rectangle> getRectangles() {

**return rectangles**;

}

**public** Label getLblTask() {

**return lblTask**;

}

}

## 6. Клас GameModeTwoController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.animation.TranslateTransition;

**import** javafx.fxml.FXML;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.shape.Circle;

**import** javafx.util.Duration;

**import** java.io.IOException;

**import** java.util.Random;

**public class** GameModeTwoController **extends** BasicController {

**private static** Duration *TRANSLATE\_DURATION* = Duration.*seconds*(0.3);

@FXML

**private** GridPane **gpGameModeTwo**;

@FXML

**private** Circle **circle**;

@FXML

**void** goToResult() **throws** IOException {

Main.*getMainController*().switchToStage(MainController.***RESULT\_CONTAINER***);

}

@FXML

**void** initialize() {

moveCircleOnMousePress(**circle**, createTranslateTransition(**circle**));

}

**private** TranslateTransition createTranslateTransition(Circle circle) {

TranslateTransition transition = **new** TranslateTransition(*TRANSLATE\_DURATION*, circle);

transition.setOnFinished(t -> {

circle.setCenterX(circle.getTranslateX() + circle.getCenterX());

circle.setCenterY(circle.getTranslateY() + circle.getCenterY());

});

**return** transition;

}

**private void** moveCircleOnMousePress(Circle circle, TranslateTransition transition) {

circle.setOnMouseMoved(event -> {

circle.setRadius(circle.getRadius() - 0.05);

transition.setToX(**new** Random().nextInt(150) - circle.getCenterX());

transition.setToY(**new** Random().nextInt(150) - circle.getCenterY());

transition.playFromStart();

**if** (circle.getRadius() < 5) {

**try** {

goToResult();

} **catch** (IOException e) {

e.printStackTrace();

}

}

});

}

**public** Circle getCircle() {

**return circle**;

}

**public** GridPane getGpGameModeTwo() {

**return gpGameModeTwo**;

}

}

7. HomePageController.

**package** controllers;

**import** javafx.fxml.FXML;

**import** eyesightChecker.Main;

**import** java.io.IOException;

**public class** HomePageController **extends** BasicController{

@FXML

**void** goToUserInfoStage() {

Main.*getMainController*().switchToStage(MainController.***INFO\_CONTAINER***);

}

@FXML

**void** goToTestOne() **throws** IOException {

Main.*getMainController*().switchToStage(MainController.***TEST\_ONE\_CONTAINER***);

}

@FXML

**void** goToChooseModeStage() {

Main.*getMainController*().switchToStage(MainController.***CHOOSE\_MODE\_CONTAINER***);

}

}

## 8. Клас InstructionController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXML;

**import** java.io.IOException;

**public class** InstructionController **extends** BasicController{

@FXML

**void** goToChooseTest() **throws** IOException {

Main.*getMainController*().switchToStage(MainController.***CHOOSE\_TEST\_CONTAINER***);

}

}

## 9.Клас MainController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXMLLoader;

**import** javafx.scene.Parent;

**import** javafx.scene.Scene;

**import** javafx.stage.Stage;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**public class** MainController {

**private** List<ControlContainer> **controlContainers**;

**private** Stage **currentStage**;

**public static final int *REPORT\_CONTAINER*** = 0;

**public static final int *INFO\_CONTAINER*** = 1;

**public static final int *INSTRUCTION\_CONTAINER*** = 2;

**public static final int *CHOOSE\_MODE\_CONTAINER*** = 3;

**public static final int *GAME\_MODE\_ONE\_CONTAINER*** = 4;

**public static final int *RESULT\_CONTAINER*** = 5;

**public static final int *TEST\_CONTAINER*** = 6;

**public static final int *HOME\_PAGE\_CONTAINER*** = 7;

**public static final int *CHOOSE\_TEST\_CONTAINER*** = 8;

**public static final int *TEST\_ONE\_CONTAINER*** = 9;

**public static final int *GAME\_MODE\_TWO\_CONTAINER*** = 10;

**public** MainController(Stage primaryStage) **throws** Exception {

**controlContainers** = **new** ArrayList<>(12);

**controlContainers**.add(***REPORT\_CONTAINER***, initInfoStage());

**controlContainers**.add(***INFO\_CONTAINER***, initInfoStage());

**controlContainers**.add(***INSTRUCTION\_CONTAINER***, initInstructionStage());

**controlContainers**.add(***CHOOSE\_MODE\_CONTAINER***, initChooseModeStage());

**controlContainers**.add(***GAME\_MODE\_ONE\_CONTAINER***, initGameMode1());

**controlContainers**.add(***RESULT\_CONTAINER***, initResultStage());

**controlContainers**.add(***TEST\_CONTAINER***, initTestStage());

**controlContainers**.add(***HOME\_PAGE\_CONTAINER***, initHomeStage(primaryStage));

**controlContainers**.add(***CHOOSE\_TEST\_CONTAINER***, initChooseTestStage());

**controlContainers**.add(***TEST\_ONE\_CONTAINER***, initTestOneStage());

**controlContainers**.add(***GAME\_MODE\_TWO\_CONTAINER***, initGameModeTwo());

}

**private** ControlContainer initHomeStage(Stage primaryStage) **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/homePage.fxml"**));

Parent parent = fxmlLoader.load();

primaryStage.setTitle(**"Eyesight checker"**);

primaryStage.setScene(**new** Scene(parent));

HomePageController controller = fxmlLoader.getController();

**currentStage** = primaryStage;

primaryStage.show();

**return new** ControlContainer(parent, primaryStage, controller);

}

**private** ControlContainer initReportStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/report.fxml"**));

Parent parent = fxmlLoader.load();

ReportController controller = fxmlLoader.getController();

controller.getLblReportText().setText(**"Name: "** + Main.*getCurrentUser*().getName() + **"\n"** +

**"Surname: "** + Main.*getCurrentUser*().getSurname() + **"\n"** +

**"Age: "** + Main.*getCurrentUser*().getAge() + **"\n"** +

**"\n--------------------------------------------------------------------------------"** +

**"\n\n\t\t\t** **"** +

**"Result: "** + Main.*getCurrentUser*().getRightAnswerCount() + **"/"** +

Main.*getCurrentUser*().getAnswerCount() + **" right answer.\n\n"** +

((Main.*getCurrentUser*().getProblemTableRow() < 7 ) ?

(**"You have problem in "** + (Main.*getCurrentUser*().getProblemTableRow()) +

**" row, please visit your doctor."**) : (**"You have good eyesight."** +

((Main.*getCurrentUser*().getProblemTableRow() < 10 ) ?

**"\nBut do not forget to do regular exercises"** +

**" for your eyes\nand from time to time check your eyesight."** : **""**))

));

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Report"**);

**return new** ControlContainer(parent, stage, controller);

}

**public void** externalInitTestStage() **throws** IOException {

**controlContainers**.remove(***TEST\_CONTAINER***);

**controlContainers**.add(***TEST\_CONTAINER***, initTestStage());

}

**public void** externalInitReportStage() **throws** IOException {

**controlContainers**.remove(***REPORT\_CONTAINER***);

**controlContainers**.add(***REPORT\_CONTAINER***, initReportStage());

}

**public void** externalInitGameModeOneStage() **throws** IOException {

**controlContainers**.remove(***GAME\_MODE\_ONE\_CONTAINER***);

**controlContainers**.add(***GAME\_MODE\_ONE\_CONTAINER***, initGameMode1());

}

**private** ControlContainer initResultStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/result.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Result"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initGameMode1() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/gameMode1.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Game mode 1"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initGameModeTwo() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/gameModeTwo.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Game mode 2"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initChooseTestStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/chooseTest.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Choose test"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initTestOneStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/test1.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Test mode 1"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initChooseModeStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/chooseMode.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Choose mode"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initInstructionStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/instruction.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Instruction"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initTestStage() **throws** java.io.IOException {

FXMLLoader testStageLoader = **new** FXMLLoader(getClass().getResource(**"/view/test.fxml"**));

Parent parent = testStageLoader.load();

TestController controller = testStageLoader.getController();

controller.getPositionLabel().setText(**"Write letter from "** + (controller.getVPosition() + 1) +

**" row and "** + (controller.getHPosition() + 1) + **" column"**);

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"Test"**);

**return new** ControlContainer(parent, stage, controller);

}

**private** ControlContainer initInfoStage() **throws** java.io.IOException {

FXMLLoader fxmlLoader = **new** FXMLLoader(getClass().getResource(**"/view/userInfo.fxml"**));

Parent parent = fxmlLoader.load();

BasicController controller = fxmlLoader.getController();

Stage stage = **new** Stage();

stage.setScene(**new** Scene(parent));

stage.setTitle(**"User Info"**);

**return new** ControlContainer(parent, stage, controller);

}

**public void** switchToStage(**int** newStage) {

**currentStage**.hide();

**currentStage** = **controlContainers**.get(newStage).getStage();

**currentStage**.show();

}

**public** List<ControlContainer> getControlContainers() {

**return controlContainers**;

}

**public** Stage getCurrentStage() {

**return currentStage**;

}

}

## 10. Клас ReportController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.Label;

**public class** ReportController **extends** BasicController{

@FXML

**void** goToHomePage() {

Main.*getMainController*().switchToStage(MainController.***HOME\_PAGE\_CONTAINER***);

}

@FXML

**private** Label **lblReportText**;

**public** Label getLblReportText() {

**return lblReportText**;

}

**public void** setLblReportText(Label lblReportText) {

**this**.**lblReportText** = lblReportText;

}

}

## 11. Клас ResultController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.Label;

**public class** ResultController **extends** BasicController {

@FXML

**private** Label **lblResult**;

@FXML

**void** goToHomePage() {

Main.*getMainController*().switchToStage(MainController.***HOME\_PAGE\_CONTAINER***);

}

@FXML

**void** goToGameMode1() {

Main.*getMainController*().switchToStage(MainController.***GAME\_MODE\_ONE\_CONTAINER***);

}

@FXML

**void** goToChooseModeStage() {

Main.*getMainController*().switchToStage(MainController.***CHOOSE\_MODE\_CONTAINER***);

}

**public** Label getLblResult() {

**return lblResult**;

}

}

## 12. Клас TestController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.animation.TranslateTransition;

**import** javafx.event.EventHandler;

**import** javafx.fxml.FXML;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Label;

**import** javafx.scene.control.TextField;

**import** javafx.scene.input.KeyCode;

**import** javafx.scene.input.KeyEvent;

**import** javafx.scene.input.MouseEvent;

**import** javafx.scene.paint.Color;

**import** javafx.scene.shape.Circle;

**import** java.io.IOException;

**import** java.util.Random;

**public class** TestController **extends** BasicController {

**private** String[] **letters** = {

**"fp"**,

**"toz"**,

**"lped"**,

**"pecfd"**,

**"edfczp"**,

**"felopzd"**,

**"defpotec"**,

**"lefodpct"**,

**"fdpltceo"**,

**"pezolcftd"**

};

**private** String[] **lettersRu** = {

**"шб"**,

**"мнк"**,

**"ымбш"**,

**"бынкм"**,

**"иншмк"**,

**"ншыикб"**,

**"шинбкы"**,

**"кншмыби"**,

**"бкшмиын"**,

**"нкибмшыб"**

};

**private int vPosition** = 0;

**private int hPosition** = **new** Random().nextInt(2);

@FXML

**void** goToReportStage() {

**try** {

Main.*getMainController*().externalInitReportStage();

} **catch** (IOException e) {

e.printStackTrace();

}

Main.*getMainController*().switchToStage(MainController.***REPORT\_CONTAINER***);

}

@FXML

**void** checkCharOnEnter() {

Scene scene = Main.*getMainController*().getControlContainers().

get(MainController.***TEST\_CONTAINER***).getStage().getScene();

scene.setOnKeyPressed(keyEvent -> {

**if** (keyEvent.getCode() == KeyCode.***ENTER***) {

checkLetter();

}

});

}

@FXML

**private** TextField **tfCompare**;

@FXML

**private** Label **positionLabel**;

@FXML

**private** Label **lblDebug**;

@FXML

**public void** checkLetter() {

**boolean** answer = **letters**[**vPosition**].substring(**hPosition**, **hPosition** + 1).equalsIgnoreCase(getTfCompare().getText());

**lblDebug**.setText(**""** + answer);

**lblDebug**.textFillProperty().set((answer) ? Color.***LIMEGREEN*** : Color.***RED***);

Main.*getCurrentUser*().setAnswerCount(Main.*getCurrentUser*().getAnswerCount() + 1);

Main.*getCurrentUser*().setRightAnswerCount(Main.*getCurrentUser*().getRightAnswerCount() + (answer ? 1 : 0));

**if** ((Main.*getCurrentUser*().getAnswerCount() -

Main.*getCurrentUser*().getRightAnswerCount()) >= 5) {

Main.*getCurrentUser*().setProblemTableRow(**vPosition** + 1);

goToReportStage();

}

**if** ((Main.*getCurrentUser*().getRightAnswerCount() == 2 || **vPosition** >= 1) &&

((Main.*getCurrentUser*().getRightAnswerCount() - 2) % 3 == 0) && answer) {

**vPosition**++;

}

**if** (!(**vPosition** >= 10)) {

**hPosition** = **new** Random().nextInt(**letters**[**vPosition**].length());

**positionLabel**.setText(**"Write letter from "** + (**vPosition** + 1) +

**" row and "** + (**hPosition** + 1) + **" column "**);

} **else** {

goToReportStage();

}

**tfCompare**.setText(**""**);

}

**public** Label getPositionLabel() {

**return positionLabel**;

}

**public** Label getLblDebug() {

**return lblDebug**;

}

**public** TextField getTfCompare() {

**return tfCompare**;

}

**public int** getVPosition() {

**return vPosition**;

}

**public void** setVPosition(**int** vPosition) {

**this**.**vPosition** = vPosition;

}

**public int** getHPosition() {

**return hPosition**;

}

**public void** setHPosition(**int** hPosition) {

**this**.**hPosition** = hPosition;

}

}

## 13. Клас TestOneController

**package** controllers;

**import** eyesightChecker.Main;

**import** javafx.application.Platform;

**import** javafx.fxml.FXML;

**import** javafx.geometry.HPos;

**import** javafx.geometry.VPos;

**import** javafx.scene.Node;

**import** javafx.scene.control.Control;

**import** javafx.scene.control.Label;

**import** javafx.scene.layout.ColumnConstraints;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.layout.Pane;

**import** javafx.scene.layout.RowConstraints;

**import** javafx.scene.paint.Color;

**import** javafx.scene.text.Font;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.concurrent.Executors;

**import** java.util.concurrent.ScheduledExecutorService;

**import** java.util.concurrent.ThreadLocalRandom;

**import** java.util.concurrent.TimeUnit;

**public class** TestOneController **extends** BasicController {

**private** List<Label> **labels**;

**private int rowQuantity** = 3;

**private int colQuantity** = 4;

**private int index**;

@FXML

**private** GridPane **gpTest**;

@FXML

**void** goToReportStage() **throws** IOException {

Main.*getMainController*().externalInitReportStage();

Main.*getMainController*().switchToStage(MainController.***REPORT\_CONTAINER***);

}

@FXML

**void** initialize() {

**labels** = **new** ArrayList<>();

**int** randomNum;

**for** (**int** i = 0; i < **rowQuantity**; i++) {

**gpTest**.addRow(i);

**for** (**int** j = 0; j < **colQuantity**; j++) {

randomNum = ThreadLocalRandom.*current*().nextInt(65, 90 + 1);

**gpTest**.addColumn(i);

**index** = ((j + 1) + i \* (**colQuantity**) - 1);

**labels**.add(**new** Label(**""** + (**char**) (randomNum)));*//(index + 1)));*

**labels**.get(**index**).setTextFill(Color.***BLACK***);

**labels**.get(**index**).setFont(Font.*font*(**"Arial"**, 15));

**gpTest**.add(**labels**.get(**index**), j, i);

}

}

**for** (Node n : **gpTest**.getChildren()) {

**if** (n **instanceof** Control) {

Control control = (Control) n;

control.setMaxSize(Double.***MAX\_VALUE***, Double.***MAX\_VALUE***);

control.setStyle(**"-fx-background-color: white; -fx-alignment: center;"**);

}

**if** (n **instanceof** Pane) {

Pane pane = (Pane) n;

pane.setMaxSize(Double.***MAX\_VALUE***, Double.***MAX\_VALUE***);

pane.setStyle(**"-fx-background-color: white; -fx-alignment: center;"**);

}

}

**gpTest**.setStyle(**"-fx-background-color: black; -fx-padding: 2; -fx-hgap: 1; -fx-vgap: 1;"**);

**gpTest**.setSnapToPixel(**false**);

*//* ***TODO: 20.06.17 Add leveling***

ColumnConstraints oneThird = **new** ColumnConstraints();

oneThird.setPercentWidth(100 / (0.0 + **colQuantity**));

oneThird.setHalignment(HPos.***CENTER***);

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

**gpTest**.getColumnConstraints().addAll(oneThird);

RowConstraints oneHalf = **new** RowConstraints();

oneHalf.setPercentHeight(100 / (0.0 + **rowQuantity**));

oneHalf.setValignment(VPos.***CENTER***);

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

**gpTest**.getRowConstraints().addAll(oneHalf);

}

@FXML

**void** update() {

*//* ***TODO: 20.06.17 fix not linked labels***

**int** randomNum;

**for** (**int** i = 0; i < **rowQuantity**; i++) {

**for** (**int** j = 0; j < **colQuantity**; j++) {

randomNum = ThreadLocalRandom.*current*().nextInt(65, 90 + 1);

**index** = ((j + 1) + i \* (**colQuantity**) - 1);

**labels**.get(**index**).setText(**""** + (**char**) (ThreadLocalRandom.*current*().nextInt(65, 90 + 1)));

}

}

**for** (Node n : **gpTest**.getChildren()) {

**if** (n **instanceof** Label) {

Label label = (Label) n;

((Label) n).setText(**""** + (**char**)ThreadLocalRandom.*current*().nextInt(65, 90 + 1));

}

}

}

}

14. Клас UserInfoController.

**package** controllers;

**import** eyesightChecker.Main;

**import** eyesightChecker.UserDataExeption;

**import** javafx.fxml.FXML;

**import** javafx.scene.control.Alert;

**import** javafx.scene.control.TextField;

**import** java.util.regex.Pattern;

**public class** UserInfoController **extends** BasicController {

*// private Pattern pattern = Pattern.compile("^([A-Za-zЇ-ї])([A-Za-zЇ-ї'ʼ`])+([A-Za-zЇ-ї])$");*

**private** Pattern **pattern** = Pattern.*compile*(**"^([A-ZЇ-Я])([a-zа-ї'ʼ`])+([a-zа-ї])$"**);

@FXML

**private** TextField **tfName**;

@FXML

**private** TextField **tfSurname**;

@FXML

**private** TextField **tfAge**;

@FXML

**void** goToInstruction() **throws** UserDataExeption {

String messageError = **"information"**;

**try** {

**if** (**pattern**.matcher(**tfName**.getText()).matches() &&

isSymbolsOccurrenceOnce(**tfName**.getText())) {

Main.*getCurrentUser*().setName(toCorrectString(**tfName**.getText()));

} **else** {

messageError = **"name"**;

**throw new** UserDataExeption();

}

**if** (**pattern**.matcher(**tfSurname**.getText()).matches() &&

isSymbolsOccurrenceOnce(**tfSurname**.getText())) {

Main.*getCurrentUser*().setSurname(toCorrectString(**tfSurname**.getText()));

} **else** {

messageError = **"surname"**;

**throw new** UserDataExeption();

}

**if** (Integer.*parseInt*(**tfAge**.getText()) >= 5 && Integer.*parseInt*(**tfAge**.getText()) <= 100) {

Main.*getCurrentUser*().setAge(Integer.*parseInt*(**tfAge**.getText()));

} **else** {

messageError = **"age"**;

**throw new** UserDataExeption();

}

Main.*getCurrentUser*().setAnswerCount(0);

Main.*getCurrentUser*().setRightAnswerCount(0);

Main.*getCurrentUser*().setProblemTableRow(0);

**tfName**.setText(**""**);

**tfSurname**.setText(**""**);

**tfAge**.setText(**""**);

Main.*getMainController*().switchToStage(MainController.***INSTRUCTION\_CONTAINER***);

} **catch** (Exception e) {

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle(**"Information Dialog"**);

alert.setHeaderText(**null**);

alert.setContentText(**"Please, write correct "** + messageError + **"!"** +

**"\n"**);

alert.showAndWait();

}

}

**private** String toCorrectString(String name) {

**return** name.substring(0, 1).toUpperCase() + name.substring(1).toLowerCase();

}

**private boolean** isSymbolsOccurrenceOnce(String text) {

String string = **""**;

String symbols = **"'ʼ`- "**;

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

string = text.replace(symbols.substring(i, i + 1), **""**);

}

**return** (text.length() - string.length()) <= 1;

}

}

## Додаток Б. Інструкція користувачеві

АВТОМАТИЗОВАНА СИСТЕМА ПЕРЕВІРКИ ЗОРУ “EYESIGHT-CHECKER”

ІНСТРУКЦІЯ КОРИСТУВАЧЕВІ

ЧДТУ. 161453-ПЗ

Аркушів 10

Розробник Качанов Б.В.

м. Черкаси

2017

**Інструкція для користувачів**

Для початку роботи з програмою необхідно запустити її. Після запуску користувач побачить головне вікно програми.

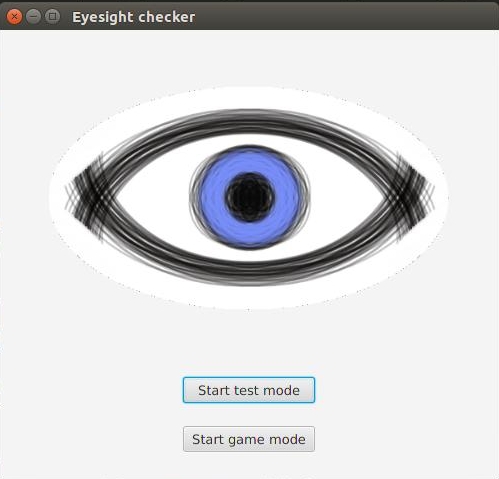


Рисунок 1. Головне вікно програми

Якщо користувач натисне кнопку «Start test mode», то він перейде у режим тестування та побачить вікно для вводу свої персональних даних.

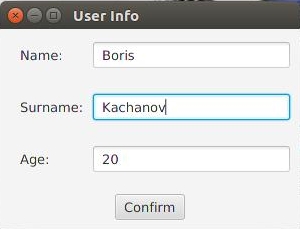


Рисунок 2. Вікно інформації користувача

Якщо користувач введе неправильну інформацію, то побачить на екрані повідомлення про помилку.

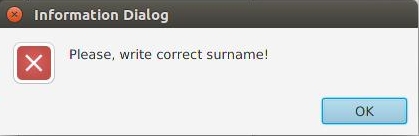


Рисунок 3. Повідомлення про помилку у введених даних.

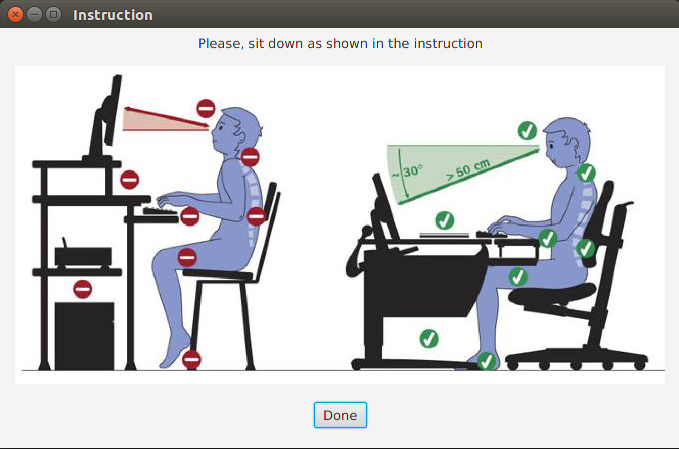
Після вводу даних та натискання кнопки «Confirm» користувач побачить інстукцію та буде повинен натиснути кнопку «Done».

Рисунок 3. Вікно з інстукцією.

Після згоди з інстукцією користувач мусить обрати один з двох режимів тестування.



Рисунок 4. Вікно для вибору режиму тестування.

Якщо користувач обрав перший режим, то перед ним з’явиться вікно з таблицею Сівцева і почнеться тестування.

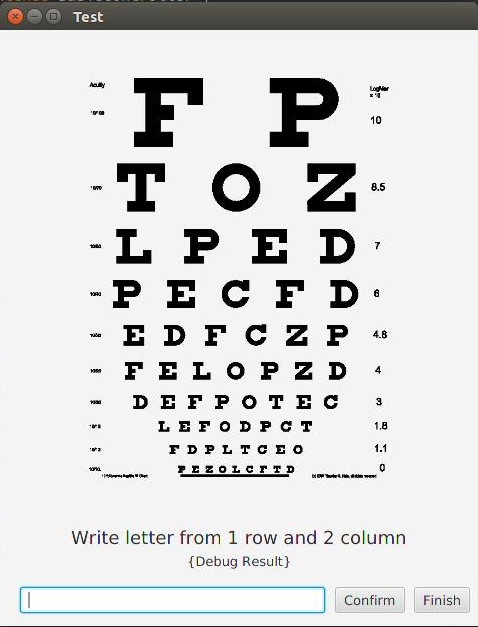


Рисунок 5. Перший режим тестування.

Під час цього тестування користувач мусить вписувати у поле букви, які вимагає програма та натискати кнопку «Confirm».

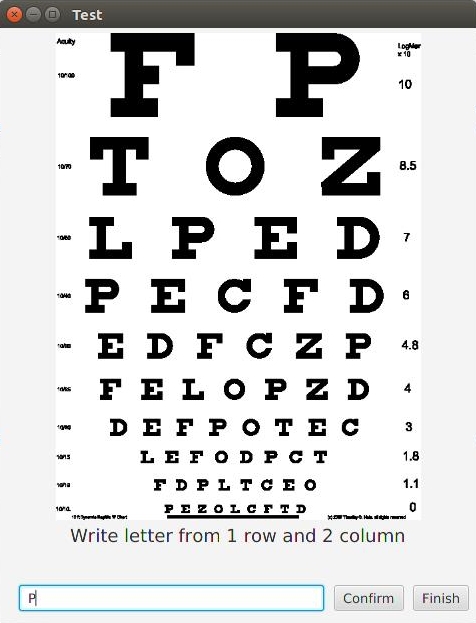


Рисунок 6. Відповідь на перше запитання.

Ці дії користувач мусить повторювати до тих пір, поки не натисне клавишу «Finish». Перед користувачем з’явиться вікно з результатами та рекомендаціями, коли тестування закінчиться.

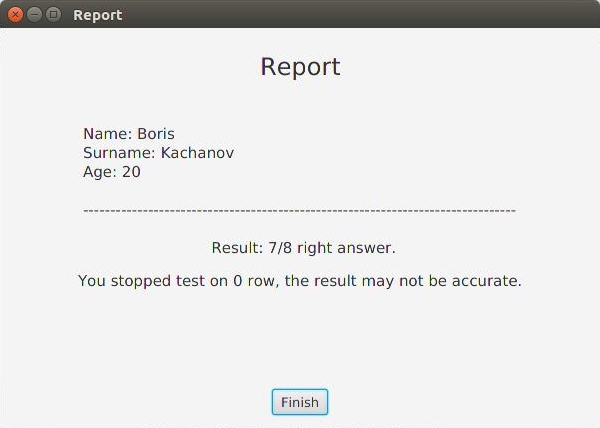


Рисунок 7. Вікно результатів.

Після повторного натискання кнопки «Finish» знову з’явиться головне вікно програми. Якщо користувач обере другий режим тестування, то на екрані з’явиться вікно для другого режима тестування зору.

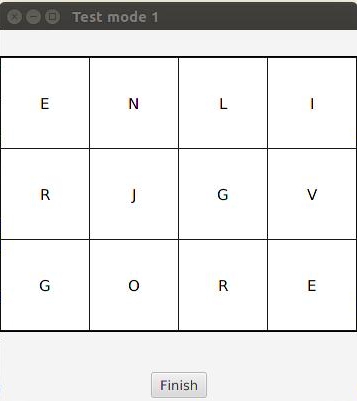


Рисунок 8. Другий режим тестування.

Після проходження другого режима тестування знову з’явиться вікно з результатами тесту, а повторне натискання кнопки «Finish» повернуть користувача до головного вікна.

Якщо на головному вікні користувач натисне кнопку «Start game mode», то з’явиться вікно вибору ігрового режиму.



Рисунок 8. Вікно вибору ігрового режиму.

Натиснувши кнопку «Mode 1» користувач запустить перший ігровий режим.

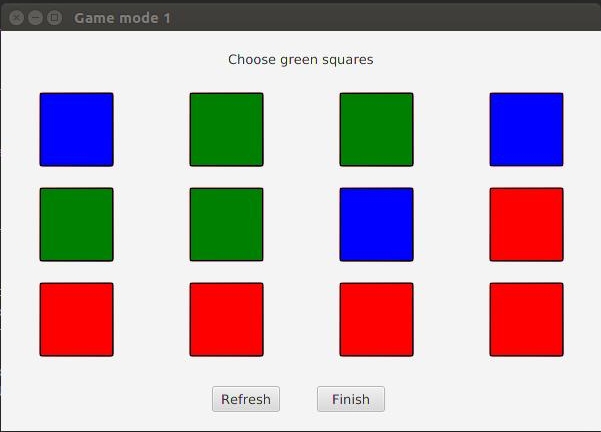


Рисунок 9. Перший ігровий режим.

В першому ігровому режимі користувач має за допомогою миші обрати комірки відповідно до завдання та натиснути кнопку «Confirm»

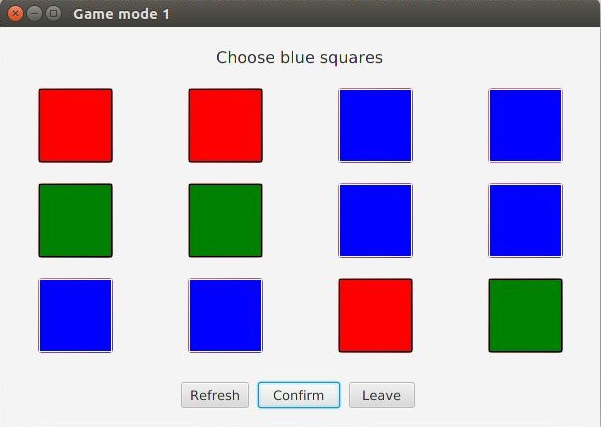


Рисунок 10. Демонстрація роботи першого ігрового режиму.

При натисканні кнопки «Refresh» кольорові комірки буде змінено на нові.

При натисканні кнопки «Leave» гра буде припинена. При натисканні кнопки «Confirm» з’явиться вікно, що повідомить користувача про кінець гри.

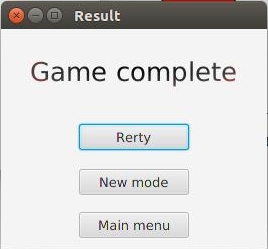


Рисунок 11. Повідомлення про кінець гри.

Кнопка «Retry» запустить ігровий режим знову. Кнопка «New mode» поверне користувача на вікно вибору ігрового режиму. Кнопка «Main menu» поверне користувача на головне меню.

Якщо користувач обере другий ігровий режим, то перед нам з’явиться відповідне вікно.

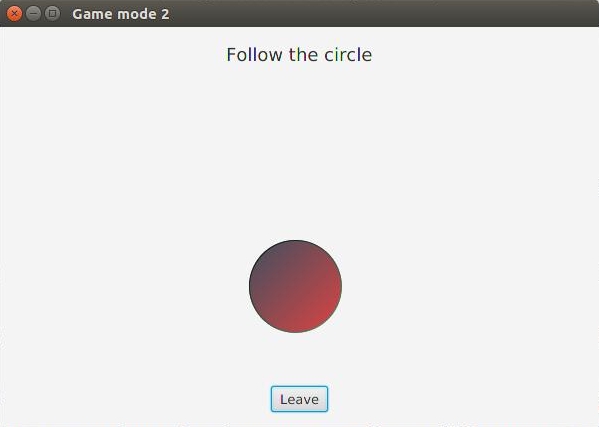


Рисунок 12. Вікно другого ігрового режиму.

Суть другого ігрового режиму полягає у тому, що користувачу необхідно за допомогою миші навести курсор до кола. Коло, в свою чергу, буде намагатися втікти від курсора, та зменьшуватися у розмірах. Гра закінчиться, коли коло буде зовсім маленького розміру, або коли користувач натисне кнопку «Leave».

# СПИСОК Використаних джерел

1. Stack Overflow [Електронний ресурс]: [Веб-сайт]. – Режим доступу: <http://stackoverflow.com/>.  – Назва з екрана.

2. Офіційна документація мови програмування Java [Електронний ресурс]: [Веб-сайт]. – Режим доступу: <https://docs.oracle.com/javase/8/>. – Назва з екрана.

3. «Философия Java» - Брюс Эккель.