Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Použité OOP Princípy

Základy objektovo-orientovaného programovania

Adam Jurčišin

Dátum: 14.12.2022

Dedenie

Trieda Player dedí z triedy User.

```
package iamquiz.model;

public class Player extends User{
    private int life = 5;
    private int correctQuestions;

    public int getLife() {
        return life;
    }

    public void setLife(int life) {
        this.life = life;
    }

    public int getCorrectQuestions() {
        return correctQuestions;
    }

    public void setCorrectQuestions() {
        correctQuestions++;
    }
}
```

Viacnásobné dedenie

Trieda SuperUser dedí z triedy Player, ktorá dedí z triedy User.

```
public class Player extends User{
```

```
public class SuperUser extends Player{
```

Agregácia

V triede FileHandling v metóde loadFile vytvárame inštancie triedy Category a Question.

```
public Category loadFile() {
    Category category = new Category(fileQuestionsName);
    try (Scanner loader = new Scanner(new FileReader(fileQuestionsName))) {
        while (loader.hasNext()) {
            Question question = new Question();
            String loadedQuestion = loader.nextLine();
            question.setQuestion(loadedQuestion);
            for (int numberOfAnswer = 0; numberOfAnswer < 4; numberOfAnswer++) {
                String loadedAnswer = loader.nextLine();
                question.setAnswers(loadedAnswer);
            }
            category.setQuestions(question);
        }
    }
    catch (IOException e) {
        System.out.println("Whoops");
    }
    return category;
}
```

Kompozícia

V triede Category vytvárame inštanciu ArrayListu typu Question.

```
public class Category {
    private final String categoryName;
    private ArrayList<Question> questions = new ArrayList<>();
```

Preťažovanie

V triede GameMode preťažujeme konštruktor podľa počtu zadaných argumentov.

```
public GameMode() {
    gameModeName = "undefined";
}
public GameMode(String gameModeName) {
    this.gameModeName = gameModeName;
}
```

Prekonávanie

V abstraktnej triede User prekonávame metódu toString() triedy Object.

```
@Override
public String toString() {
    return name.toUpperCase();
}
```

Zapuzdrenie

V triede Question používame zapuzdrenie pre získanie prístupu k privátnym atribútom triedy.

```
public class Question {
    private String question;
    private int indexOfCorrectAnswer = 0;
    private final ArrayList<String> answers = new ArrayList<>();

    public ArrayList<String> getAnswers() {
        return answers;
    }

    public void setAnswers(String answer) {
        answers.add(answer);
    }

    public String getQuestion() {
        return question;
    }

    public void setQuestion(String question) {
        this.question = question;
    }
```

Statická metóda

V triede GameScreenControl používame statickú metódu setScannedCharacter().

```
public static void setScannedCharacter() {
    try{
        scannedCharacter = sc.nextLine().charAt(0);
    }
    catch(Exception e){
        System.out.println("No character was entered.");
    }
}
```

Statický atribút

V triede GameScreenControl používame statické atribúty actualScreen a scannedCharacter.

```
public class GameScreenControl {
    private static String actualScreen = "mainMenu";
    private static char scannedCharacter = ' ';
```

Finálny atribút

V triede Game používame finálny atribút GAMENAME.

```
public class Game {
    private static final String GAMENAME = "I AM QUIZ";
```

Finálna metóda

V triede FileHandling používame finálnu metódu loadFile(), ktorá načíta súbor s otázkami.

```
public final Category loadFile() {
    Category category = new Category(fileQuestionsName);
    try (Scanner loader = new Scanner(new FileReader(fileQuestionsName))) {
        while (loader.hasNext()) {
            Question question = new Question();
            String loadedQuestion = loader.nextLine();
            question.setQuestion(loadedQuestion);
            for (int numberOfAnswer = 0; numberOfAnswer < 4; numberOfAnswer++) {
                String loadedAnswer = loader.nextLine();
                question.setAnswers(loadedAnswer);
            }
            category.setQuestions(question);
        }
    }
    catch (IOException e) {
        System.out.println("Whoops");
    }
    return category;
}
```

Rozhranie

```
public interface Screens {
    1 implementation
    void displayMainMenu();
```

Default metóda

Interface Screens obsahuje defaultnú metódu displayEndScreen().

```
default void displayEndScreen(){
    System.out.println("|\n|\n|\n|\t\t\t\t\t\tThank you for playing!\n|\n|\n|");
}
```

Abstraktná trieda

Trieda User je abstraktnou triedou.

```
public abstract class User {
    public String name;
```

Abstraktná metóda

Abstraktná trieda User obsahuje abstraktnú metódu displayMode3Stats().

```
1 implementation
public abstract void displayMode3Stats();
```

Upcasting

```
public void setGameMode3Settings(char scannedCharacter, GameScreen gameScreen){
    User user1 = new Player();
    User user2 = new Player();
```

Downcasting

```
        startGame3(gameScreen, (Player)user1, (Player)user2, gameMode);
    }
}

public void startGame3(GameScreen gameScreen, Player player1, Player player2, GameMode gameMode){
```

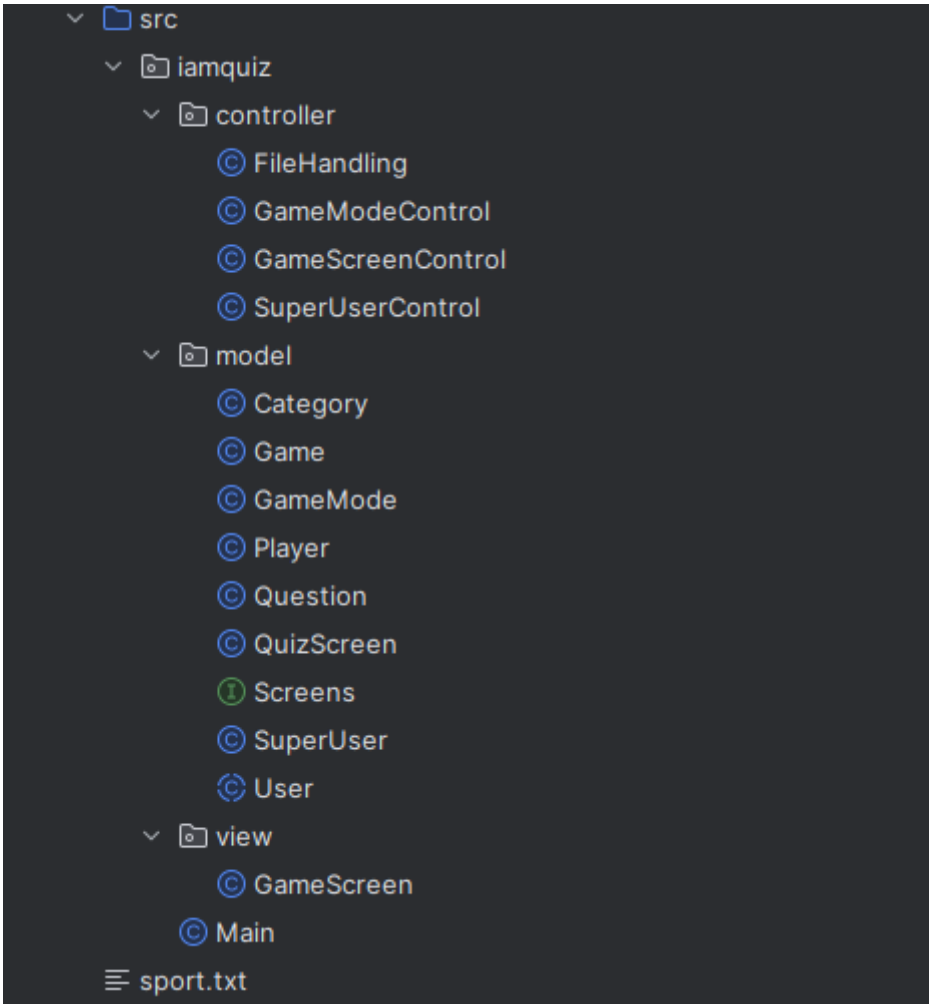
Polymorfizmus

V triede GameScreen v metóde displayTypesOfUsers využívame polymorfizmus pri volaní metódy whoAmI(), ktorá sa nachádza v triedach User, Player a SuperUser, ktorá vypisuje rozdielne typy používateľov volaním rovnakej metódy.

```
public void displayTypesOfUsers(ArrayList<User> typesOfUsers) {
    displayBorder();
    System.out.println("|\n|\t\t\t\t\t\t\t SUPERUSER MENU\n|\tTypes of users in quiz:");
    for (User typesOfUser : typesOfUsers) {
        System.out.println("|\t\t" + typesOfUser.whoAmI());
    }
    System.out.println("|\n|Press 'x' to go back to menu");
    displayBorderBottom();
}
```

Roztriedenie do balíkov

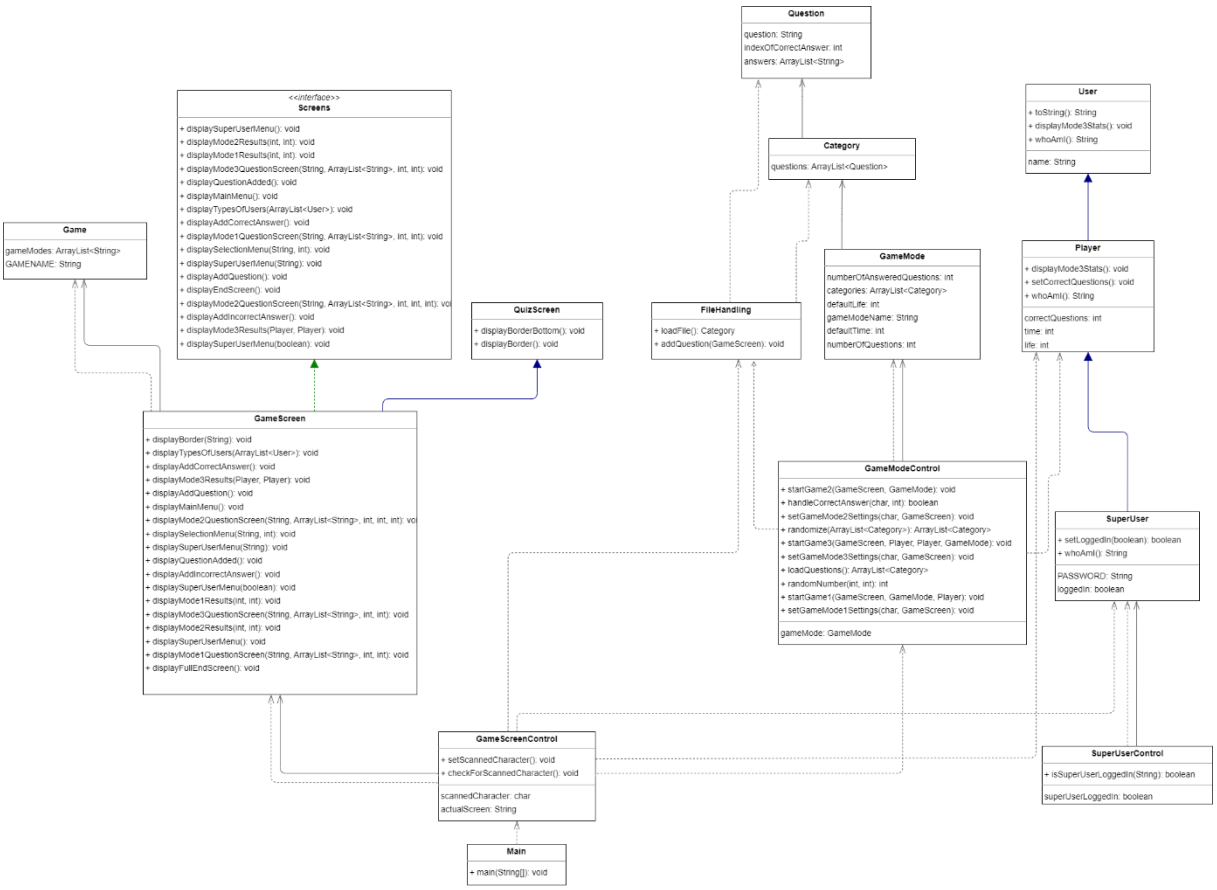
Vhodné roztriedenie tried do balíkov podľa návrhového vzoru MVC.



Ostatné body

V programe využívame camelCase, pri triedach využívame PascalCase, pri metódach využívame komentáre.

UML Diagram


Question
question: String
indexOfCorrectAnswer: int
answers: ArrayList<String>
<<interface>>
Screens
+ displaySuperUserMenu(): void
+ displayMode2Results(int, int): void
+ displayMode1Results(int, int): void
+ displayMode3QuestionScreen(String, ArrayList<String>, int, int): void
+ displayQuestionAdded(): void
+ displayMainMenu(): void
+ displayTypesOfUsers(ArrayList<User>): void
+ displayAddCorrectAnswer(): void
+ displayMode1QuestionScreen(String, ArrayList<String>, int, int): void
+ displaySelectionMenu(String, int): void
+ displaySuperUserMenu(String): void
+ displayAddQuestion(): void
+ displayEndScreen(): void
+ displayMode2QuestionScreen(String, ArrayList<String>, int, int, int): voi
+ displayAddIncorrectAnswer(): void
+ displayMode3Results(Player, Player): void
+ displaySuperUserMenu(boolean): void
Category
questions: ArrayList<Question>
User
+ toString(): String
+ displayMode3Stats(): void
+ whoAmI(): String
name: String
Game
gameModes: ArrayList<String>
GAMENAME: String
GameMode
numberOfAnsweredQuestions: int
categories: ArrayList<Category>
defaultLife: int
gameModeName: String
defaultTime: int
numberOfQuestions: int
Player
+ displayMode3Stats(): void
+ setCorrectQuestions(): void
+ whoAmI(): String
correctQuestions: int
time: int
life: int
QuizScreen
+ displayBorderBottom(): void
+ displayBorder(): void
FileHandling
+ loadFile(): Category
+ addQuestion(GameScreen): void
GameScreen
+ displayBorder(String): void
+ displayTypesOfUsers(ArrayList<User>): void
+ displayAddCorrectAnswer(): void
+ displayMode3Results(Player, Player): void
+ displayAddQuestion(): void
+ displayMainMenu(): void
+ displayMode2QuestionScreen(String, ArrayList<String>, int, int, int): voi
+ displaySelectionMenu(String, int): void
+ displaySuperUserMenu(String): void
+ displayQuestionAdded(): void
+ displayAddIncorrectAnswer(): void
+ displaySuperUserMenu(boolean): void
+ displayMode1Results(int, int): void
+ displayMode3QuestionScreen(String, ArrayList<String>, int, int): void
+ displayMode2Results(int, int): void
+ displaySuperUserMenu(): void
+ displayMode1QuestionScreen(String, ArrayList<String>, int, int): void
+ displayFullEndScreen(): void
GameModeControl
+ startGame2(GameScreen, GameMode): void
+ handleCorrectAnswer(char, int): boolean
+ setGameMode2Settings(char, GameScreen): void
+ randomize(ArrayList<Category>): ArrayList<Category>
+ startGame3(GameScreen, Player, Player, GameMode): void
+ setGameMode3Settings(char, GameScreen): void
+ loadQuestions(): ArrayList<Category>
+ randomNumber(int, int): int
+ startGame1(GameScreen, GameMode, Player): void
+ setGameMode1Settings(char, GameScreen): void
gameMode: GameMode
SuperUser
+ setLoggedIn(boolean): boolean
+ whoAmI(): String
PASSWORD: String
loggedin: boolean
GameScreenControl
+ setScannedCharacter(): void
+ checkForScannedCharacter(): void
scannedCharacter: char
actualScreen: String
SuperUserControl
+ isSuperUserLoggedIn(String): boolean
superUserLoggedIn: boolean
Main
+ main(String[]): void