Contents

[1.0 Introduction 2](#_Toc59184842)

[2.0 Packages and Classes 2](#_Toc59184843)

[2.1 Model Package 2](#_Toc59184844)

[1.1.1 Game Class 2](#_Toc59184845)

[1.1.2 Course Class 3](#_Toc59184846)

[1.1.3 User/Student Class 4](#_Toc59184847)

[2.2 View Package 6](#_Toc59184848)

[2.2.1 Program Class 6](#_Toc59184849)

[3.0 Results 11](#_Toc59184850)

# Introduction

This project focuses on creating an application that allows you to enroll into any course and play simple mini games associated with the course. There are three main parts in this project.

1. Creating mini games.
2. Inputting course details and adding games to be played within the course.
3. Allowing registered students to play games and increase their credits.

This is to ensure that some students can increase their credits while having fun in the course.

# Packages and Classes

This application is created using the J2EE design pattern, MVC. But, the application only contains two specific packages since a database is not integrated to it.

1. Model package
2. View package

## 2.1 Model Package

The model package encapsulates the data of User (Student), Course, and Game. Here, these three classes are implemented to ensure that the abstraction and encapsulation of the application is preserved.

### Game Class

The game class contains the following attributes.

1. gameIndex (String)
2. gameName (String)
3. gameAnswer (String)

And the getters and setters of each variable is created along with the default constructor and parameterized constructor,

The complete implementation of Game class is as follows.

package Model;

public class Game {

private String gameIndex;

private String gameName;

private String gameAnswer;

/\*Default constructor\*/

public Game() {

}

/\*Parameterized Constructor\*/

public Game(String gameIndex, String gameName, String gameAnswer) {

this.gameName = gameName;

this.gameIndex = gameIndex;

this.gameAnswer = gameAnswer;

}

/\*\*

\* @return the gameName

\*/

public String getGameName() {

return gameName;

}

/\*\*

\* @param gameName the gameName to set

\*/

public void setGameName(String gameName) {

this.gameName = gameName;

}

/\*\*

\* @return the gameIndex

\*/

public String getGameIndex() {

return gameIndex;

}

/\*\*

\* @param gameIndex the gameIndex to set

\*/

public void setGameIndex(String gameIndex) {

this.gameIndex = gameIndex;

}

/\*\*

\* @return the gameAnswer

\*/

public String getGameAnswer() {

return gameAnswer;

}

/\*\*

\* @param gameAnswer the gameAnswer to set

\*/

public void setGameAnswer(String gameAnswer) {

this.gameAnswer = gameAnswer;

}

}

### Course Class

The Course class contains following attributes.

1. courseId (String)
2. courseName (String)
3. courseCredit (double)
4. games (String [])

The complete implementation of Course class is as follows.

package Model;

public class Course {

private String courseId;

private String courseName;

private double courseCredit;

private String[] games;

/\*Default Constructor\*/

public Course(){}

/\*Parameterized Constructor\*/

public Course(String courseId,String courseName, double courseCredit, String []games ){

this.courseName = courseName;

this.courseCredit = courseCredit;

this.games = games;

this.courseId = courseId;

}

/\*\*

\* @return the courseName

\*/

public String getCourseName() {

return courseName;

}

/\*\*

\* @param courseName the courseName to set

\*/

public void setCourseName(String courseName) {

this.courseName = courseName;

}

/\*\*

\* @return the courseCredit

\*/

public double getCourseCredit() {

return courseCredit;

}

/\*\*

\* @param courseCredit the courseCredit to set

\*/

public void setCourseCredit(double courseCredit) {

this.courseCredit = courseCredit;

}

/\*\*

\* @return the games

\*/

public String[] getGames() {

return games;

}

/\*\*

\* @param games the games to set

\*/

public void setGames(String[] games) {

this.games = games;

}

/\*\*

\* @return the courseId

\*/

public String getCourseId() {

return courseId;

}

/\*\*

\* @param courseId the courseId to set

\*/

public void setCourseId(String courseId) {

this.courseId = courseId;

}

}

### User/Student Class

The user class has following attributes.

1. studentName (String)
2. userName (String)
3. password (String)
4. courses (HashMap<String, String>)
5. gameCount (HashMap<String, Integer>)

Here, the two hash maps are used to contain the courses that student being registered to and the number of times that particular student had played the games in a particular course.

Following is the implementation of the Student class.

package Model;

import java.util.HashMap;

public class User {

private String studentName;

private String userName;

private String password;

private HashMap<String, String> courses;

private HashMap<String,Integer> gameCount;

/\*Default constructor\*/

public User(){}

/\*Parameterized Constructor\*/

public User(String studentName,String username, String password, HashMap<String, String> courses, HashMap<String,Integer>gameCount){

this.studentName = studentName;

this.userName = username;

this.password = password;

this.courses = courses;

this.gameCount = gameCount;

}

/\*\*

\* @return the userName

\*/

public String getUserName() {

return userName;

}

/\*\*

\* @param userName the userName to set

\*/

public void setUserName(String userName) {

this.userName = userName;

}

/\*\*

\* @return the password

\*/

public String getPassword() {

return password;

}

/\*\*

\* @param password the password to set

\*/

public void setPassword(String password) {

this.password = password;

}

/\*\*

\* @return the courses

\*/

public HashMap<String, String> getCourses() {

return courses;

}

/\*\*

\* @param courses the courses to set

\*/

public void setCourses(HashMap<String, String> courses) {

this.courses = courses;

}

/\*\*

\* @return the studentName

\*/

public String getStudentName() {

return studentName;

}

/\*\*

\* @param studentName the studentName to set

\*/

public void setStudentName(String studentName) {

this.studentName = studentName;

}

/\*\*

\* @return the gameCount

\*/

public HashMap<String,Integer> getGameCount() {

return gameCount;

}

/\*\*

\* @param gameCount the gameCount to set

\*/

public void setGameCount(HashMap<String,Integer> gameCount) {

this.gameCount = gameCount;

}

}

## 2.2 View Package

The View package contains only one class. And it is the Program class.

### 2.2.1 Program Class

First, the Program class is created with the intention of having an instance named program at the main method. That method specifies the courses, games, and students you can have in your program. To hold those values the ArrayList<> data structure has been used.

In instance methods, gameAdding(Game game), userAdding(User user) and courseAdding(Course course) are used so that they can be appended to their respective ArrayLists by calling those functions.

Scanner input = new Scanner(System.in) is the main way used here for inputting data.

First, the user inputs the games, after that user inputs courses and select the games you can assign for that course, so that you do not have to stick to one game. After that student details, courses that they have enrolled, and their grades are input to the system. Then, it gives the opportunity to play games and increase their credits on a certain course. A student can play three times per course and increase the current credits by 1% per a win.

The whole process of this application starts with the creation of the instance Program program = new Program();

First, there is a while loop that allows you to enter game id, game name and the game answer through input instance we created using java.util.Scanner class. You can breakout from the while loop by typing “n” when it asks that you want to enter a game.

Second, there is another while loop for you to enter course id, course name, course credits and the games that you can play in that course. You must enter them as the given instructions and that will create a new Course instance and we can include it in our program instance’s instance variable courseArray. You can breakout from the while loop by typing “n” when it asks whether you want to add another course.

Next, you can enter the student details by typing their name, username, password, the courses they have enrolled and their grades for those courses. Although the constructor of User class demands that it needs a HashMap<String, Integer> object to handle the number of games they will play per a course, we will only be giving that parameter a new HashMap with zero elements.

You can breakout from the while loop which says you to input student details by typing “n” when it asks whether you want to add another student.

After that, you can play games. First you have to input your username and password to play a game. Then the program goes through the userArray and its user elements and find whether there is a user like that. If not, it will tell you that the username or password is incorrect. But, if the username and password is correct, it will show you a table which has all the courses you have enrolled and the games within those courses. Then the program will ask you to select a course and select a game within that course. And you can play the game if you have not already played the games for that course up to three times. If the answer you provide to that particular game is true your credits will increase 1%, and if the answer is wrong the credits will remain the same. To do this the program will go through gameArray, courseArray and userArray several times, so to increase the performance of this, we can integrate this with a database like MySQL.

Then, there will be a package called “controller” for our control classes.

The implementation of the Program class is shown below.

package View;

import Model.Course;

import Model.Game;

import Model.User;

import dnl.utils.text.table.TextTable;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Program {

private ArrayList<Course> coursesArray = new ArrayList<>();

private ArrayList<Game> gameArray = new ArrayList<>();

private ArrayList<User> userArray = new ArrayList<>();

Program() {

}

public void courseAdding(Course course) {

coursesArray.add(course);

}

public void gameAdding(Game game) {

gameArray.add(game);

}

public void userAdding(User user) {

userArray.add(user);

}

public static void main(String[] args) {

/\*Create an instance of the main program\*/

Program program = new Program();

/\*Create an instance of the Scanner\*/

Scanner input = new Scanner(System.in);

/\*

\* First you have to organize the games

\* Then, you can assign certain games to certain courses

\* After that, you should have student who can follow different courses

\* Finally you can let student's play games

\*/

/\*Lets initialize the games\*/

System.out.println("Input games (to exit press 'n'): ");

System.out.println("------------------------------------");

String gameVal = "";

while (!gameVal.equalsIgnoreCase("n")) {

System.out.print("Do you want to add a Game? (y/n): ");

gameVal = input.nextLine();

if (gameVal.equalsIgnoreCase("y")) {

System.out.print("Game Id: ");

String gameId = input.nextLine();

System.out.print("Game Name: ");

String gameName = input.nextLine();

System.out.print("Game answer: ");

String gameAnswer = input.nextLine();

Game game = new Game(gameId, gameName, gameAnswer);

program.gameAdding(game);

} else {

break;

}

}

/\*Initialization of courses\*/

System.out.println("");

System.out.println("Input courses (to exit press 'n'): ");

System.out.println("------------------------------------");

String courseVal = "";

while (!courseVal.equalsIgnoreCase("n")) {

System.out.print("Do you want to add a course? (y/n): ");

courseVal = input.nextLine();

boolean gamesAvailable = program.gameArray.size() != 0;

if (gamesAvailable) {

String gameColumns[] = {"Game Id", "Game Name"};

Object gameData[][] = new String[program.gameArray.size()][2];

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

Game game = program.gameArray.get(i);

String gameDetail[] = {game.getGameIndex(), game.getGameName()};

gameData[i][0] = gameDetail[0];

gameData[i][1] = gameDetail[1];

}

TextTable gameTable = new TextTable(gameColumns, gameData);

if (courseVal.equalsIgnoreCase("y")) {

System.out.print("Course Id: ");

String courseId = input.nextLine();

System.out.print("Course Name: ");

String courseName = input.nextLine();

System.out.print("Course Credits: ");

String courseCredits = input.nextLine();

double courseCredits\_double = Double.parseDouble(courseCredits);

System.out.println("Select course games from the following table. (type games to be included in g1\_id,g2\_id,... format)");

gameTable.printTable();

String courseGameDetails = input.nextLine();

String courseGames[] = courseGameDetails.split(",");

Course course = new Course(courseId, courseName, courseCredits\_double, courseGames);

program.courseAdding(course);

} else {

break;

}

} else {

System.out.println("There are no games available");

}

}

/\*Initialization of student details\*/

System.out.println("");

System.out.println("Input student details (to exit press 'n'): ");

System.out.println("------------------------------------");

String userVal = "";

while (!userVal.equalsIgnoreCase("n")) {

String courseColumns[] = {"Course Id", "Course Name"};

Object courseData[][] = new String[program.coursesArray.size()][2];

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

Course course = program.coursesArray.get(i);

String courseDetail[] = {course.getCourseId(), course.getCourseName()};

courseData[i][0] = courseDetail[0];

courseData[i][1] = courseDetail[1];

}

TextTable courseTable = new TextTable(courseColumns, courseData);

System.out.print("Do you want to add a Student? (y/n): ");

userVal = input.nextLine();

if (userVal.equalsIgnoreCase("y")) {

System.out.print("Student Name: ");

String studentName = input.nextLine();

System.out.print("Student Username: ");

String username = input.nextLine();

System.out.print("Student password: ");

String password = input.nextLine();

System.out.print("Select the courses you enrolled from the following table. (type courses and grades as c1\_id:grade,c2\_id:grade,...)");

courseTable.printTable();

String courseDetails = input.nextLine();

HashMap<String, String> courseMap = new HashMap<String, String>();

String coursesArray[] = courseDetails.split(",");

for (String s : coursesArray) {

String values[] = s.split(":");

courseMap.put(values[0], values[1]);

}

HashMap<String, Integer> gameCount = new HashMap<>();

User user = new User(studentName, username, password, courseMap, gameCount);

program.userAdding(user);

} else {

break;

}

}

/\*Let students play games\*/

System.out.println("");

System.out.println("Let the gaming begins");

System.out.println("--------------------------");

String gamingVal = "";

while (!gamingVal.equalsIgnoreCase("n")) {

System.out.print("Do you want to start a game? (y/n): ");

gamingVal = input.nextLine();

if (gamingVal.equalsIgnoreCase("y")) {

System.out.print("Input username: ");

String username = input.nextLine();

System.out.print("Input password: ");

String password = input.nextLine();

String columnNames[] = {"Course ID", "Course Name", "Course Grade", "Course Games", "Game Ids"};

ArrayList<String> studentCourses = new ArrayList<>();

ArrayList<String> studentCredits = new ArrayList<>();

/\*

ArrayList<String> gameKeys = new ArrayList<>();

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

\*/

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

if (program.userArray.get(i).getUserName().equals(username) && program.userArray.get(i).getPassword().equals(password)) {

User user = program.userArray.get(i);

HashMap<String, String> courseMap = user.getCourses();

//HashMap<String,Integer> gameC = user.getGameCount();

for (String s : courseMap.keySet()) {

studentCourses.add(s);

studentCredits.add(courseMap.get(s));

}

/\*

for (String s : gameC.keySet()) {

gameKeys.add(s);

gameCount.add(gameC.get(s));

}\*/

} else {

System.out.println("Your username or password is incorrect");

}

}

Object data[][] = new Object[studentCourses.size()][5];

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

data[i][0] = studentCourses.get(i);

data[i][2] = studentCredits.get(i);

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

Course course = program.coursesArray.get(i);

if (course.getCourseId().equals(studentCourses.get(i))) {

data[i][1] = course.getCourseName();

String gameIdArray[] = course.getGames();

String gIDs = "";

ArrayList<String> gameNames = new ArrayList<>();

for (int k = 0; k < gameIdArray.length; k++) {

gIDs += gameIdArray[k] + ", ";

for (int m = 0; m < program.gameArray.size(); m++) {

Game game = program.gameArray.get(m);

if (game.getGameIndex().equals(gameIdArray[k])) {

gameNames.add(game.getGameName());

}

}

}

String gNames = "";

for (int k = 0; k < gameNames.size(); k++) {

gNames += gameNames.get(k) + ", ";

}

data[i][3] = gNames;

data[i][4] = gIDs;

}

}

}

TextTable detailsTable = new TextTable(columnNames, data);

detailsTable.printTable();

System.out.print("Input course: ");

String course = input.nextLine();

System.out.print("Input game you are going to play(Input game Id): ");

String game = input.nextLine();

System.out.print("Input game answer: ");

String gameAnswer = input.nextLine();

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

Game g = program.gameArray.get(i);

if (g.getGameIndex().equalsIgnoreCase(game)) {

boolean compareAnswer = g.getGameAnswer().equalsIgnoreCase(gameAnswer);

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

if (program.userArray.get(j).getUserName().equals(username) && program.userArray.get(i).getPassword().equals(password)) {

User u = program.userArray.get(j);

for (int k = 0; k < program.coursesArray.size(); k++) {

if (program.coursesArray.get(k).getCourseId().equalsIgnoreCase(course)) {

Course c = program.coursesArray.get(k);

HashMap<String, Integer> gameDetails = u.getGameCount();

String key = c.getCourseId(); //game + " " +c.getCourseId()

boolean isHashMap = gameDetails.containsKey(key);

if (isHashMap) {

int temp = gameDetails.get(key);

if (temp != 3) {

gameDetails.put(key, temp + 1);

if (compareAnswer) {

System.out.println("Your asnwer is correct");

double grade = Double.parseDouble(u.getCourses().get(key)) \* (1.01);

HashMap<String, String> updatedCourse = u.getCourses();

updatedCourse.put(key, Double.toString(grade));

u.setCourses(updatedCourse);

} else {

System.out.println("Your answer is incorrect");

}

} else {

System.out.println("You have already played this game three times");

}

} else {

gameDetails.put(key, 1);

if (compareAnswer) {

System.out.println("Your asnwer is correct");

double grade = Double.parseDouble(u.getCourses().get(key)) \* (1.01);

HashMap<String, String> updatedCourse = u.getCourses();

updatedCourse.put(key, Double.toString(grade));

u.setCourses(updatedCourse);

} else {

System.out.println("Your answer is incorrect");

}

}

u.setGameCount(gameDetails);

}

}

}

}

}

}

}

}

}

}

# 3.0 Results

The execution of the program is shown below. To view the data in tabular manner, the TextTable object was used. To use that object j-text-utils-0.3.4.jar file has been used in the program.

