Library Management System

Masterclass in Java

Saba Vashakmadze

Student at Ilia State University

Saba.vashakmadze.1@iliauni.edu .ge

Description

Create Library Management System (LMS) in Java. LMS is widely used software. It can be any complexity.

Our example is basic one, which have the following features:

1. storage for the games

2. ability to add the game in the library

3. ability remove the game from the library

4. ability to print the library game information on the console

LMS structure

We will need the following classes for the software:

1. Game – the game itself.

2. LMS – library management system.

3. LibraryTester – the tester class. This class will be used to test our management system.

Class Game

The class Game should have several fields, including Type and genre. This class can be implemented in the

following way:

```
package library;

public class Game {

private String type, genre;

public String getType() {

return type;
```

}

public void setType(String type) {

this.type = type;

}

public String getGenre() {

return genre;

Class Game

String type

String genre

Class LMS

List <Game> storage

void addGame(Game)

boolean removeGame(Game)

Void printStorage()

}

public void setGenre(String genre) {

this.genre = genre;

}

}

Pay attention to the setters and getters of the fields. In general, all the fields are private (unless the special

requirements are stated) and the access functions are implemented such as setters and getters.

Read about toString() function and implement it for Book class.

Class LMS

The library management system should have an inner structure for storing books. The management system

should have methods for adding the new books and removing the old ones. It should have the ability to print

the entire library content when needed. The class can be implemented in the following way:

```
package library;
import java.util.ArrayList;
import java.util.List;
public class LMS {
// Mapping with Game and the number of this game in the library
private List<Game> storage = new ArrayList<Game>();
// adds the game to the library
public void addGame(Game game) {
storage.add(game);
}
// removes the game from the library
public boolean removeGame(Game game) {
boolean removed = false;
for (int i = 0; i < storage.size(); i++) {
Game b = storage.get(i);
if (b.getType().equals(game.getType()) && b.getGenre().equals(game.getGenre())) {
storage.remove(i);
removed = true;
break;
}
}
return removed;
}
public void printStorage() {
if (storage.isEmpty()) {
System.out.println("The storage is empty");
} else {
for (Game b: storage) {
System.out.println(b.getGenre() + ", " + b.getType());
System.out.println();
}
```

```
}
}
}
```

Pay attention to the usage of the ArrayList class, for loops for the lists, break clause and the String object comparison. It is a good point to understand how Interface works. Usage of the boolean variables can also be observed in this example.

LMS Tester class

Now let's test our management system. First, create some games. Then create LLM and add those games to the library using the LLM. Then try to remove some of the games.

```
package library;
public class LibraryTester {
public static void main(String[] args) {
Student s1 = new Student();
s1.setName("Saba");
s1.setSurname("Vashakmadze");
s1.setPn("12345678912");
Student s2 = new Student();
s2.setName("Giorgi");
s2.setSurname("Giorgadze");
s2.setPn("111111111111");
Game b1 = new Game();
b1.setType("RPG");
b1.setGenre("Adventure");
Game b2 = new Game();
b2.setType("Battleroyale");
b2.setGenre("Shooting");
LMS lms = new LMS();
lms.addGame(b1);
lms.addGame(b1);
lms.addGame(b2);
```

```
lms.removeGame(b1);

lms.printStorage();

}

}
```

We print the state of the library to check if all the methods are working properly.