Update readme.md
Jonas Solhaug Kaad authored 3 months ago

Forked from an inaccessible project.

readme.md 3.75 KiB

Practice Assignment 2

You are the person responsible for recruiting new actors in Hollywood. As you also happen to be a gifted java developer, you conclude that you can make recruitment better and easier by creating a program that can save the candidates to a text file, and later retrieve their information.

You are provided two classes `Actor.java` and `Helpers.java` in the package `actor_details`.

Task 2a

Complete the implementation of `Actor.java`. Remember to use the correct access modifiers for variables and methods.

- Declare 3 variables:
  - `name(String)`
  - `age(int)`
  - `hasBeenInterviewed(boolean)`
- Create a constructor to initialize the 3 variables.
- Implement 3 `getter` methods for all the variables.
- Implement and override the `toString` method, and use the `getter` methods in the `toString()` to print the actors information.
- Implement the `main()` method:
  - Create 3 `Actor` objects with the following data:
  - ```
    Ryan Reynolds           46      false
    Leonardo DiCaprio       48      false
    Tom Holland             26      true
    ```
  - Create an `ArrayList` of the type `Actor`.
  - Add the `Actor` objects to the `ArrayList`.
  - Print the `ArrayList` to check your implementation.

Example of correct output:

```
{ name='Ryan Reynolds', age='46', hasBeenInterviewed='false'}
{ name='Leonardo DiCaprio', age='48', hasBeenInterviewed='false'}
{ name='Tom Holland', age='26', hasBeenInterviewed='true'}
```

Task 2b

Complete the implementation of `Helpers.java` file.

Implement `writeToFile(ArrayList<Actor> actorList)` method:

- Use a `PrintWriter/FileWriter` or similar to write to the file `Actors.txt`.
- Iterate over all objects in the `actorList` and write each object to the file, using the `toString` method.
- Use the `try-with-resources` or ensure that you enclose the output stream in a `try – catch` clause and close your output stream after use.
```

- Make sure to catch the relevant exceptions that can be thrown.

Implement `readFromFile(String inputFile)` method:

- Read from an input file given as `inputFile` using a `Scanner` or `FileReader`
- Iterate over the contents of the file and print it out to the screen.
- Use the `try-with-resources` or ensure that you enclose the output stream in a `try – catch` clause and close your output stream after use.
- Make sure to catch the relevant exceptions that can be thrown.

Implement `checkHasBeenInterviewed(String inputFile)` method

- Read from an input file given as `inputFile` using a `Scanner` or `FileReader`
- Iterate over the contents of the file and only print those actors who have been interviewed (`hasBeenInterviewed=true`).
  - ***Hint:*** *You can use `.split(",")` to split the line on every comma "," and then save it to a `String[]` and check that array for the interview condition.*
- Use the `try-with-resources` or ensure that you enclose the output stream in a `try – catch` clause and close your output stream after use.
- Make sure to catch the relevant exceptions that can be thrown.

Task 2c

Finally call/invoke your helper methods in the `main()` method of `Actor.java` . Assuming, the `ArrayList` you made earlier comprises 3 actor objects. First:

- Call `writeToFile()` method and pass the `ArrayList` you made as an argument.
- Then call the `readFromFile()` method and pass `Actors.txt` as an argument
- Finally call the `checkHasBeenInterviewed()` method and pass `Actors.txt` as an argument again.

Hint: *As the methods throw an exception remember to wrap in try catch block.*

Example of correct output (all methods are implemented):

```
{ name='Ryan Reynolds', age='46', hasBeenInterviewed='false'}
{ name='Leonardo DiCaprio', age='48', hasBeenInterviewed='false'}
{ name='Tom Holland', age='26', hasBeenInterviewed='true'}

{ name='Tom Holland', age='26', hasBeenInterviewed='true'}
```

File - C:\Users\vivek\Downloads\Practice_Assignment2_VOP\src\main\java\actor_details\Actor.java

```java
package actor_details;

import java.io.IOException;
import java.util.ArrayList;

public class Actor {
    //Det første del af opgaven er simpel fra OOP.
    //Husk, at bruge Generator til at danne Getters/Setter.

    String name;
    int age;
    boolean hasBeenInterviewed;

    public Actor(String name, int age, boolean hasBeenInterviewed){
        this.name = name;
        this.age = age;
        this.hasBeenInterviewed = hasBeenInterviewed;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isHasBeenInterviewed() {
        return hasBeenInterviewed;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

```

```
40      public boolean hasBeenInterviewed() {
41          return hasBeenInterviewed;
42      }
43
44
45      public String toString(){
46          return "Name: " + getName() + ", Age: " +
    getAge() + ", Interviewed: " + hasBeenInterviewed();
47      }
48
49      public static void main(String[] args) throws
    IOException {
50
51          Actor skuespiller1 = new Actor("Kim Larsen",
    59,true);
52          Actor skuespiller2 = new Actor("Chris Evans",
    39,true);
53          Actor skuespiller3 = new Actor("Jackie Chan",
    40,false);
54
55          ArrayList<Actor> skuespillerne = new
    ArrayList<>();
56          skuespillerne.add(skuespiller1);
57          skuespillerne.add(skuespiller2);
58          skuespillerne.add(skuespiller3);
59
60          for(Actor skuespiler : skuespillerne){
61              System.out.println(skuespiler.toString
    ());
62          }
63          Helpers.writeToFile(skuespillerne);
64          Helpers.readFromFile("Actors.txt");
65          Helpers.checkHasBeenInterviewed("Actors.txt"
    );
66      }
67  }
68
```

```java
package actor_details;

import java.io.*;
import java.util.ArrayList;
import java.util.Scanner;

public class Helpers {

    //Følg trinene forsigtigt.
    //Husk, at anvende Try/Catch når du arbejde med Filer.
    //Husk, at lukke filerne så de ikke spiser din kode som mad.
    //Brug toString for at kunne konvertere dit data til string.

    /**
     *
     * This method takes the name of the file as an argument, then prints the contents of the file to the
     * users screen.
     *
     * @throws IOException
     */
    public static void readFromFile(String inputFile) throws IOException {
        try {
            Scanner input = new Scanner(new FileReader("Actor.txt"));
            //Vores word er kendetegnet for ord.
            String word;
            try (Scanner scanner = new Scanner(new FileReader("inputFile"))) {
                //Vi læser filen og siger, at inputter skal skanne/begynde fra næste linje.
                while (scanner.hasNextLine()) {
                    //Linjen skal tages som input af skanner-klassen.
                    //Derefter skal linjen printes.
                    String line = scanner.nextLine();
                    System.out.println(line);
```

```
33                  }
34                  //Husk, at bruge Try/Catch og benyt
   de rigtige exceptions.
35              } catch (FileNotFoundException e) {
36                  System.out.println("File was not
   found: " + e.getMessage());
37              } catch (IOException e) {
38                  System.out.println("Reading the File
   was not Possible" + e.getMessage());
39              }
40          }
41
42          /**
43           *
44           * This method takes the name of the file as
   its argument and only prints
45           * out the actors who have been interviewed
46           *
47           * @throws IOException
48           */
49          public static void checkHasBeenInterviewed (
   String inputFile){
50              //Samme koncept her med File-Læsning.
51              try {
52                  Scanner insert = new Scanner(new
   FileReader("inputFile"));
53                  while (insert.hasNextLine()) {
54                      //Når du læser filen, skal du
   lave det som vi lige har gjort før.
55                      //Definer din linje som input fra
    skanner linje der skal læses.
56                      String line = insert.nextLine();
57                      //Vi vil gerne have at vores
   actordata skal splittes i komma.
58                      String[] actordata = line.split(
   ",");
59                      if (actordata.length >= 3) {
60                          String name = actordata[0].
   trim();
61                          int age = Integer.parseInt(
   actordata[1].trim());
```

```
62                          boolean hasBeenInterviewed
    = Boolean.parseBoolean(actordata[2].trim());
63                          Actor actor = new Actor(name
   , age, hasBeenInterviewed);
64
65                          if (actor.hasBeenInterviewed
   ) {
66                              System.out.println(actor
   .toString());
67                              insert.close();
68                          }
69                      }
70                  }
71              } catch (FileNotFoundException e) {
72                  e.printStackTrace();
73              }
74          }
75      }
76
77      /**
78       *
79       * Implement the method below so the contents of
    the object
80       * are written to the file Actors.txt, in the
   root of the project.
81       * if the file already exists overwrite it.
82       *
83       * @throws IOException
84       */
85      public static void writeToFile(ArrayList<Actor>
   actorList) throws IOException {
86          try {
87              //Den er meget simpel og her skal vi
   bare læse filen og køre hvert ord igennem ved at
   sætte det som string.
88              //Derefter lukke vi filen.
89              FileWriter writer = new FileWriter("
   Actor.txt");
90              for (Actor lists : actorList) {
91                  writer.write(actorList.toString());
92                  writer.close();
```

```
 93                 }
 94             } catch (FileNotFoundException e) {
 95                 e.printStackTrace();
 96             }
 97
 98         }
 99 }
100
101
102
103
```