TUTORIAL 03 – MORE ON CLASSES AND OBJECTS

Guided Tasks

Task 01 – Rectangle – 20 minutes

```
public class Rectangle {
      public double length; // from 0.5 to 5.0 cm
      public double width; // from 10.0 to 100.0 cm
};
```

1) What is the issue with the above class as per the OOP principles we discussed in Lecture 01?

2) Provide a redesigned Rectangle class that uses more appropriate access modifiers according to the encapsulation principle. Write the code

3) How does the default constructor look like when you do not provide one? Write the code

4) Provide a constructor for the Rectangle class that will initialize the instance variables to suitable (valid) start values.

5) Provide signatures for set and get methods for the above instance variables

6) Write bodies for the methods for which you provided signatures in question (5) above. Note that for the set method, the implementation must prevent the two instance variables from being set to invalid values.

7) Write a main method that instantiates the Rectangle class and use the set and get methods you designed.

Task 02 – Cylinder – 40 minutes

Write a class called Cylinder that satisfies the following specification

1. Identify and add the necessary instance variables of a Cylinder

2. Add a static variable to keep track of how many Cylinder objects are created

3. Add a default constructor

4. Provide a constructor for the Cylinder class that will initialize the instance variables to suitable (valid) start values.

5. Provide getter and setter methods

6. Provide a way of increasing the static variable to keep track of the no of cylinder objects created

7. Provide a way of accessing the static variable to view the no of cylinder objects created

If the following task cannot be completed during this week that is fine as and when we progress complete this at home

8. ADDITIONAL CHALLENGE 01 – override the toString() method
9. ADDITIONAL CHALLENGE 02 – override the equals() method
10. ADDITIONAL CHALLENGE 03 – When you override the equals() method what is the other method you will override and also provide code to override the method
11. ADDITIONAL CHALLENGE 04 – what is the interface we must implement and override the method if we wish to sort object and also provide the code

Task 03 – Date Class – 20 minutes

Implement the Date Class with the following specifications that contains of:

- Three *private instance variables*. They represent day (int), month (int), and year (int).

- One *constructor* to initialise the date with values:
  ```
  public Date (int day, int month, int year) { //write code
  here }
  ```

- Public *getters* and *setters*:
  ```
  public void setDay(int day){ // write

  code here} public void setMonth(int

  month){ // write code here} public void

  setYear(int year){ // write code here}

  public int getDay(){ //write code here}

  public int getMonth(){ //write code here}

  public int getYear(){ //write code here}
  ```

- A *toString()* method that returns the string "Date[day = ?, month = ?, year =?]" Example : `"Date [ day = 23, month = 11, year = 2020]`

  If you don't remember the functionality of the "toString()" method go to https://www.javatpoint.com/understanding-toString()-method

Task 04 – Director Class – 20 minutes

1) Implement a class Director that models a movie's director and the class contains:

   - Four *private instance variables*. They represent first name (string), surname (string), the number of movies directed (int) and the date of birth (Date) of the director.

   - One *constructor* to initialises the name and the surname of the Director with values: `public Director (String name, String Surname) {…}`

   - Public *getters* and *setters*:

     ```
     public  string getName(){ … } public  string

     getSurname(){ … } public  Date getDoB(){ … }

     public int getNumberOfMovie(){ … }

     public void setDoB(Date date) { … }

     public void setNumberOfMovie (int num) { … }
     ```

     There are no setter methods for name and surname because we don't want to change these attributes.

   - A *toString()* method that returns the string "Director[name = ?, surname = ?, dob = ?, movies directed = ?
     ]"
     Example : "`Director [ name = James, surname = Cameron, dob = 16/8/1954, movies directed = 23]`

2) Note that the class Director uses a Date object to represent the date of birth of the director. You can use the implementation of the class Date that you wrote in the previous task.

3) Write a test class to test the public methods you implemented. Note that you should create an instance of *Date* before you can construct an instance of *Director*:

```
/*
 * A test class for the Director class.
 */
public class Test {
   public static void main(String[] args) {

      // Test constructor
      Director director = new Director("James", "Cameron");

      // Test Setters and Getters

      // Crete an object Date to represent the dob
      Date dob = new Date (16, 8, 1954);
      director.setDoB(dob);
```

```
        director.setNumberOfMovie(23);

        System.out.println(director);  // toString()
        System.out.println("name is: " + director.getName());
        System.out.println("surname is: " + director.getSurname());
        System.out.println("dob is: " + director.getDoB().getDate());
        System.out.println("number of directed movies is: " +
                                   director.getNumberOfMovie());


    }
}
```

Task 05 – Movie Class – 20 minutes

Movie Class

1) Implement a class called *Movie* that models a movie directed by a *director.* This class contains:

- Four *private instance variables*. They represent the title of the movie (string), the category (string), the number of awards (int) and the director (Director).

- One *constructor* to initialize the title and the category, the name of the Director, with values:

```
public Movie (String title, String category, Director director)
{ … }
```

- Public *getters* and *setters*:

```
public  string getTitle(){ … }

public  string getCategory() { … }

public Director getDirector() { … }

public void setNumAwards(int numAwards) { … }

public int getNumAwards() { … }
```

- A *toString()* method that returns the string "Movie[title = ?, category = ?, director name = ?, director surname = ?, number of awards = ?]"

  Example : "`Movie [ title = Avatar, category = Fantasy, director name = James, director surname = Cameron, number of awards = 3]`

2) Write a test class to test all the public method in the class *Movie*. Note that you have to create an instance of *Director* before you can construct an instance of *Movie*.

```
/*
 * A test program for the Movie class.
 */
public class Test {
   public static void main(String[] args) {

      // We need a Director instance to create a Movie instance
      Director director = new Director("James", "Cameron");
      Date dob = new Date(16, 8, 1954);
      director.setDoB(dob);
      director.setNumberOfMovie(23);

      System.out.println(director);  // Director's toString()
```

```
        // Test Movie's constructor and toString()
        Movie movie = new Movie("Avatar", "Fantasy",director);
        System.out.println(movie);  // Movie's toString()

        // Test Setters and Getters
        movie.setNumAwards(23);

        System.out.println(movie);  // Book's toString()


       System.out.println("title is: " + movie.getTitle());
       System.out.println("category is: " + movie.getCategory ());
       System.out.println("name of director is: " +
       movie.getDirector().getName());
       System.out.println("surname of director is: " +
       movie.getDirector().getSurname());
      System.out.println("number of awards is " + movie.getNumAwards());
   }
}
```

Unguided Tasks

Week 04 – 1 hour

Improve the class Date:

Add the input validation when the user inserts the date according to the values in the orange box shown in the figure above.

Modify the method `getDate()` in order to return the date in the form gg/mm/yyyy. Add a leading zero when is needed *or* class and the usage of the class within the *Movie* class.