CPRG 251
Assignment 1 (Modules 1 – 2)
Spring/Summer 2020

Name: ______________________

Marks: ________ / 29

D2L Submission Instructions

1) One ZIP file needs to submit to D2L with the following naming convention **CPRG251_A1_Firstname_Lastname.zip** using your first and last name.
    1. If working in a group of two (2), only one team member needs to submit to D2L (both can if you so wish). Both members will receive the same feedback. The file should have the following naming convention: **CPRG251_A1_Lastname of member 1_Lastname of member 2.zip**
2) The ZIP file must contain the following:
    1. The following directory structure:
        - bin/ – Compiled Java files.
        - src/ – Java source code files:
            - sait/mms/application/
            - sait/mms/managers/
            - sait/mms/problemdomain/
        - doc/ – Generated Javadoc files.
            - Ensure the *private* option is checked and everything is included in the generated documentation.
        - lib/ – Any third-party libraries. This folder can be empty.
        - res/ – Any resource or data files.
        - test/ – Unit test cases. This folder can be empty.
    2. A text file named **Readme.txt** in the root folder of the ZIP archive and contain:
        - A project title.
        - What the program does.
        - The date.
        - The author
        - How to run the program.
    3. A runnable JAR file in the root folder of the ZIP archive.
        a. Use the naming convention: **FirstInitial.Lastname1.jar** (i.e.: J.Blow1.jar).
        b. It is to be built using only Eclipse IDE and JDK 1.8x.

Assignment Instructions

1. You will use only Eclipse IDE.
2. The due date for this assignment is posted in D2L in the assignment submission area and in the provided calendar located in the *Course Information* area**.** Any assignment submitted after the due date will receive a mark of zero, but feedback maybe given.
3. Submissions must be student's original work. Refer to the Academic Misconduct (AC.3.4) policies and procedures.

Problem

Write a movie management system using object-oriented design principles. The program will read from the supplied data file into a single array list. The data file (movies.txt) contains information about the movies. Each movie record has the following attributes:

- Duration (in minutes)
- Title
- Year of release

Each record in the movies.txt file is formatted as follows:

- `Duration,Title,Year`
- e.g.: `91,Gravity,2013`

Specifically, you have to create an interactive menu driven application that gives the user the following options:

1. **Add a new movie and save.**
   - The user will be prompted to enter the duration in minutes, title of the new movie, and the year the movie was released. Before the movie is added, the inputs provided by the user should be validated:
     - The duration and year of the movie should not be zero and the title of the movie should not be empty.
   - When you add a new movie, the program should update the data file by saving the new movie at the end of the movie list.
2. **Generate list of movies released in a year.**
   - The user will input a year and the program will display a list of all the movies released in that year along with the duration (in minutes) of all the movies.
   - The list of movies does not have to be sorted.
3. **Generate list of random movies.**
   - The user will input the number of movies and the program will display a list containing the amount of random movies along with the duration (in minutes) of all the movies.
   - There is no minimum or maximum duration for the movies in the list.
   - You can use *Collections.shuffle* in the *java.utils* package to randomize the movie list.

4. **Exit the program.**

- Save the list of movies back into the data file “movies.txt” using the above format (`Duration,Title,Year).`

Notes:

To follow the object-oriented principles, your project should contain ONLY the following classes and methods in their respective package.

Package	**Class**	**Methods**
`sait.mms.application`	`AppDriver`	`main`
`sait.mms.managers`	`MovieManagementSystem`	`displayMenu, addMovie, generateMovieListInYear, generateRandomMovieList, loadMovieList`
`sait.mms.problemdomain`	`Movie`	`Non-default constructor, Accessor methods, toString`

You cannot use parallel and/or nested arrays in this assignment (you can use ArrayList).

Sample runs:

An example of an added and saved a new movie (where input is shown in **bold underline**):

```
Movie Management system
1     Add New Movie and Save
2     Generate List of Movies Released in a Year
3     Generate List of Random Movies
4     Exit

Enter an option: 1

Enter duration: 100
Enter movie title: We are Gold
Enter year: 2019
Saving movies...
Added movie to the data file.
```

An example of a generated list of movies released in a year (where input is shown in **bold underline**):

```
Movie Management system
1     Add New Movie and Save
2     Generate List of Movies Released in a Year
3     Generate List of Random Movies
4     Exit

Enter an option: 2

Enter in year: 1996

Movie List
Duration    Year  Title
103         1996  DragonHeart
93          1996  Trainspotting
145         1996  Independence Day
Total duration: 341 minutes
```

An example of invalid option (where input is shown in **bold underline**):

```
Movie Management system
1     Add New Movie and Save
2     Generate List of Movies Released in a Year
3     Generate List of Random Movies
4     Exit

Enter an option: 5

Invalid option!
```

An example of a generated list of random movies (where input is shown in **bold underline**):

```
Movie Management system
1     Add New Movie and Save
2     Generate List of Movies Released in a Year
3     Generate List of Random Movies
4     Exit

Enter an option: 3

Enter number of movies: 5

Movie List
Duration    Year  Title
129         2016  Now You See Me 2
139         1999  Fight Club
136         2014  Captain America: The Winter Soldier
81          1995  Toy Story
103         2017  Life

Total duration: 588 minutes
```

Marking Guide

Follows submission guidelines			
1.	Correct files		
2.	File naming conventions followed		
3.	NO extra files		
Subtotal			**/3**
Movie Management System			
4.	Compile and run		
5.	Runnable JAR file		
6.	Generate proper documentation		
7.	Clear instructions		
8.	Follow object-oriented structure		
9.	Follow naming conventions		
10.	Follow packages and classes		
11.	Menu design		
12.	Add and save new movie		
13.	Validate inputted movie		
14.	Generate list of movies released in a year		
15.	Generate list of random movies		
Subtotal			**/26**
Total			**/29**