CSE1IOO/CSE4IOO Real-Time (Programming) Test

Information Sheet

- You will code and submit your answers during your allocated test session on *Thursday of Week 13 of the semester*
- You will only have access to a unix account to produce your code.
- You will be given 5 minutes reading time followed by 1.5 hours (1 hour 30 minutes) in which to program and submit your solution.
- You are required to have your student card on display beside the computer during the lab.
- You may bring with you into the laboratory:
  - One unmarked Java textbook
  - A pen

  All other belongings should be secured in your bag.

Real-time Test Details

- The real-time test will require
  - Reading a given text file into an API ArrayList<E> object,
  - Closing that file and then
  - Manipulating the ArrayList.

  Solutions that use any other data structures (i.e. not using ArrayList<E> class) will receive no marks.
- Your code does NOT need to be commented.
- A summary of selected methods for ArrayList, Scanner and BufferedReader will be provided.

Sample Real-time Test

A sample real-time test and a sample text file are provided. You should code a solution to it on your unix account as practice for this test.

Sample Text File

The following text file (to be called Movies.dat) can be used in your practice for Task 2 of real-time test.

```
4
High Noon
Western
500
50000
.
Gladiator
Epics
1000
200000
.
ET
Science Fiction
1000
-1
.
Titanic
Drama
-1
-1
.
```

CSE1IOO Real-Time Test

Student Name:_____

Username:_____Signature:_____

- Reading time: 5 minutes. Writing time: 1hour and 30 minutes
-
- The code does NOT need to be commented.
- Submit all .java files one at a time once you have checked that your solution executes as required. *(Note: if your program does not compile and execute, still submit it for part marks. You can submit the same file more than once.)* Use the command:

  `submit LAB filename`

  where filename means the .java files you have coded (e.g. Film.java and CinemaDriver.java)

Task 1

Code a Java class called `Film` with the following *attributes*:

`title`	a String for the title (name) of the film – may be more than one word. It is immutable (should never change once it is set).
`genre`	a String for the genre of the film – may be more than one word. It is immutable (should never change once it is set).
`screenings`	an int for the number of times the film has been shown. The `screenings` attribute may be changed later.
`attendance`	an int for the total number of customers that have seen the film at all screenings. The `attendance` attribute may be changed later.

Provide the `Film` class with the following *methods*:

a constructor	takes 4 arguments, one for each attribute: the film's title, genre, screenings and attendance.
`toString`	returns a String containing the attributes and their values.
get methods	for all attributes.
set methods	for those attributes that may require them.
`calcAverage`	takes no arguments and returns a double that represents the average number of customers that have seen the film per screening. If either `screenings` or `attendance` is -1 then -1 is returned, otherwise `attendance` divided by `screenings` is returned.

When the `Film` class compiles (using *javac Film.java*), move on to Task 2.

Task 2

Code a Java driver class called `CinemaDriver` to do the following:

a) Read information from a text file *Movies.dat* into a `ArrayList` of `Film` objects. You may hard-code the input file name in `CinemaDriver`.

You may assume the text file exists, is correctly formatted and is not empty. It contains, as the first data item, the number of film records, followed by the film records of the form (each record ends with a dot on a separate line):

title
genre
screenings
attendance
.

A partial listing of an example of *Movies.dat* may be as shown below:

```
6
The Samurai
Actions
222
222222
.
Jaws
Triller
100
-1
.
// other details of other films
```

b) Close the input file after creating the list of `Film` objects (there are no more interactions with this or any other files).

c) Display the current contents of the `ArrayList` to screen. For each film, display the four attributes, separated by the backslash character '\'.

d) Now set the screenings of the film with title "ET" to the value 246000.

e) Again display the current contents of the `ArrayList` to screen.

f) Now add a film with title "Rear Window", genre "Mystery", with 100 screenings seen by 12000 customers to the front of the `ArrayList`.

g) Now remove the fourth film in the list (i.e. the film at index 3). You can assume that there are at least four films.

h) Again display the current contents of the `ArrayList` to screen.

i) Finally for every film currently in the `ArrayList`, display to screen its average attendance per screening (as returned by the `calcAverage` method of the `Film` class) and then display the title of the film with the highest average. You may assume there are no ties.