BIRZEIT UNIVERSITY

COMP231 – Advanced Programming

Assignment # 4

Objectives:

1. Create hierarchy of Classes and Objects using Inheritance relationships.
2. Demonstrate the added value of using the following concepts:
   - Polymorphism
   - Generic biding
   - Inheritance
   - Abstract classes and Interfaces
3. To read/write data from/into a file using the **Scanner/PrintWriter** classes.

Specification

Submission: through Ritaj.

What to submit: Your **own** **well-structured** and **well-commented JAVA** files (.java) and UML diagrams (compressed into a **studentId_sec#.rar** file, e.g. 1134567_sec1.rar).

Deadline: **22/12/2015** by midnight.

Tasks

Task 1: Complex numbers

A complex number is a number of the form $a + bi$, where **a** and **b** are real numbers and **i** is $\sqrt{-1}$. The numbers **a** and **b** are known as the real part and imaginary part of the complex number, respectively. You can perform **addition**, **subtraction**, **multiplication**, and **division** for complex numbers using the following formula:

$$a + bi + c + di = (a + c) + (b + d)i$$
$$a + bi - (c + di) = (a - c) + (b - d)i$$
$$(a + bi) * (c + di) = (ac - bd) + (bc + ad)i$$
$$(a + bi) / (c + di) = (ac + bd) / (c^2 + d^2) + (bc - ad)i / (c^2 + d^2)$$

You can also obtain the **absolute** value for a complex number using the following formula:

$$| a + bi | = \sqrt{a^2 + b^2}$$

- Design a class named **Complex** for representing complex numbers and the methods **add**, **subtract**, **multiply**, **divide**, **abs** for performing complex-number operations, and override **toString** method for

returning a string representation for a complex number. The **toString** method returns **a + bi** as a string. If **b** is **0**, it simply returns **a**.

- Provide three constructors **Complex(a, b)**, **Complex(a)**, and **Complex()**. **Complex()** creates a **Complex** object for number **0** and **Complex(a)** creates a **Complex** object with **0** for **b**. Also provide the **getRealPart()** and **getImaginaryPart()** methods for returning the real and imaginary part of the complex number, respectively.
- Your Complex class should also implement the **Cloneable** interface.
- Write a test program that prompts the user to enter two complex numbers and display the result of their addition, subtraction, multiplication, and division. Here is a sample run:

<Output>
Enter the first complex number: **3.5 5.5**
Enter the second complex number: **-3.5 1**
(3.5 + 5.5i) + (-3.5 + 1.0i) = 0.0 + 6.5i
(3.5 + 5.5i) - (-3.5 + 1.0i) = 7.0 + 4.5i
(3.5 + 5.5i) * (-3.5 + 1.0i) = -17.75 + -15.75i
(3.5 + 5.5i) / (-3.5 + 1.0i) = -0.5094 + -1.7i
|3.5 + 5.5i| = 6.519202405202649
<End Output>

Task 2: Movies

In a movie library application, you are asked to consider the following kinds of movies:

- **Movie**, a class describing all kinds of movies.
- **Action**, a movie containing lots of explosions.
- **Romance**, a movie where romantic interest drives the plot.
- **Comedy**, a movie with largely humorous content.
- **Mystery**, a who done it movie.
- **Rescue**, a hybrid ***action-romance*** movie, where the main character attempts to save his or her romantic interest from almost certain doom.
- **Romantic Comedy**, a hybrid ***romance-comedy*** with large amounts of both humorous and romantic content.
- **Hollywood Blockbuster**, an **action-romance-comedy-mystery** movie designed to please crowds.

- What **interfaces** and **classes** would you use to represent the previous list of movies? Write your answer by carefully drawing a **UML class/interface** hierarchy, identifying which nodes are **classes** and which are **interfaces**.
    - Note that there must be a class for each type of the movies, but you may use any interfaces you require to preserve the relationships between types.

- Adequately, implement all the classes/interfaces. Include whatever methods/parameters necessary in each class/interface.
- Your classes should all implement the **Cloneable** and **Comparable** interfaces.
- Force all the sub-classes to override **toString** method.
- Crate a text file "**movies.txt**" that includes information about variety of movies. The following is a sample **movies.text** file:

Type: Title, [Starring], Running time, Country, Language.
Comedy: Minions, [Sandra Bullock, Jon Hamm], 91, USA, English.
Action: The Matrix, [Keanu Reeves, Laurence Fishburne], 120, USA, English.
Hollywood Blockbuster: Gladiator, [Russell Crowe, Joaquin Phoenix], 155, USA, English.
Mystery: Harry Potter, [Daniel Radcliffe, Rupert Grint, Emma Watson], 178, UK, English.
Action: Entrapment, [Sean Connery, Catherine Zeta-Jones], 113, USA, English.
Comedy: Dumb and Dumber, [Jim Carrey, Jeff Daniels], 107, USA, English.
Rescue: Avatar, [Sam Worthington, Zoe Saldana], 161, USA, English.
Romance: Titanic, [Leonardo DiCaprio, Kate Winslet, Billy Zane], 195, USA, English.

- Write a test program that do the following:
  - Open and read the **movies.txt** file that you created in the previous step.
  - Convert each line to the appropriate movie class based on the first token (e.g. **Comedy**).
  - Create an **ArrayList** of movies to store the created movies from the previous step.
  - Write a method to sort the **ArrayList** of movies based on movie type and running time.
  - Write the sorted **ArrayList** of movies into a file called "**sorted_movies.txt**".

Good Luck!