جامعة بيرزيت
BIRZEIT UNIVERSITY

COMP231 – Advanced Programming

Assignment # 3

Objectives:

1. Create hierarchy of Classes and Objects using Inheritance relationships.
2. Demonstrate the added value of using the following concepts:
   - Polymorphism
   - Generic biding
   - Multiple Inheritance
   - Abstract classes and Interfaces

Specification

Submission: through Ritaj.
What to submit: A compressed file of your own **well-structured** and **well-commented** JAVA classes and a **word** document.

Deadline: Friday **5/12/2014**.

Tasks

Task 1: Movies[1]

In a movie store 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-dunnit 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.

[1] Note: use a Microsoft word file to answer this question and to draw the **UML**.

Task 2: Company

Write a program to model three different kinds of employees in a company using inheritance and File I/O. Your program will consist of three classes that extend the class **Worker** and a testing class **Company** that reads in payroll information from a file then prints it to the screen.

The Class **Worker**:

The worker has the following properties: **ID number, Name, Hours,** and hour **Rate**. And the following abstract methods:

- **earnedAmount()**: returns the total weekly earned amount
- **toString()**: returns a string representing the object information.
- **equals(Worker)**: returns if the current object's weekly earning equals the entered worker earning.

The Classes **SalariedWorker, HourlyWorker**, and **TemporaryWorker**:

To model employees, extend the provided abstract class **Worker** by writing three new classes: **SalariedWorker**, **HourlyWorker**, and **TemporaryWorker**. Implement the abstract methods as required and make them ***Comparable*** and ***Cloneable***.

The Class **Company**:

Create a class called **Company** with a **main** method to test your classes. This method should create an ArrayList of Workers. Then, populate the array using data from a text file ***payroll.txt*** using appropriate Java I/O techniques. Each line of the file will look similar to this:

Salaried,1,Mamoun,40,20

Based on the first value (e.g., ***Salaried***) your method will know whether to create a new **SalariedWorker**, **HourlyWorker**, or **TemporaryWorker** object. Once created, populate the object with the remaining values (ID: ***1***, name: ***Mamoun***, hours: ***40***, rate: ***20***) then add the object to the array.

Finally, sort the array based on the worker's earning and then iterate through the array of works using the "enhanced for loop" syntax, and print out each worker in the following format, replacing each word in **(bold)** with values from the object:

(**ID**):(**Name**) is a (**Salaried/Hourly/Temporary**) worker. Total weekly earning is (**earned amount**).

Good Luck!