OBJECT ORIENTED PROGRAMMING (JAVA) (CST8284)

Assignment 2

Implementing Inheritance and Polymorphism

(Marks: 9%)

You are required to:

- Carefully review **all files** provided to you to understand the logic and hierarchy involved in this application, and what you need to do.
- Do **NOT** proceed with the tasks in this slide deck until you have reviewed each code file.
- Examine the tasks stated in this slide deck to ensure that you are doing the right thing.
- You will use the files you were given in **this assignment folder**

Overview

- In this assignment you will leverage on your knowledge of **inheritance**, and **polymorphism** and to implement polymorphic behavior on using inheritance.

- You are required to modify the **Clock.java** class that has been provided to you in this lab, such that it will include the full functions as described in this document.

- The **ClockDemo.java** is provided and it tests the **Clock.java** class.
- The **WorldClockDemo.java** is provided and it tests the **WorldClock.java** class.

YOUR TASKS

Tasks: (0)

- Load the following given classes into Eclipse IDE:
  - Clock.java
  - ClockDemo.java
  - WorldClockDemo.java

Tasks: (1)

- Modify the class Clock whose getHours() and getMinutes() methods return the current time at your location.
  - Hint: Call java.time.LocalTime.now().toString() and extract the time from that string.
- Provide a getTime() method that returns a string with the hours and minutes by calling the getHours() and getMinutes() methods.
  - Hint: use String methods to extract the time and the hour.
- Test your code by executing **ClockDemo** main() method.

Tasks: (2)

- Provide a subclass **WorldClock** inheriting from **Clock,** whose constructor accepts a time offset. For example, if you live in Ottawa, a new WorldClock(-4) should show the time in Ottawa, four time zones after Universal Time Clock UTC. WorldClock(1) shows time in a time zone 1 hour ahead of UTC.

- In **WorldClock.java** class, override getTime() as needed to return the correct time in the given time zone.
  - Be careful as the hour can't exceed 24 hours in a day, or be negative.

- Test your code by executing WorldClockDemo.java

Tasks: (3)

- Create **AlarmClock** class that inherits from **Clock**.
    - When setAlarm(hours, minutes) is called, the AlarmClock stores the alarm.
    - When getTime() is called, and the alarm time has been reached or *exceeded*, return the time followed by the string "Alarm“, and clear the alarm.

- Write a test class **ClocksDemo.java,** that would:
    1. Create a Clock, a WorldClock and an AlarmClock with some parameters.
    2. Stores all clocks in an array of Clock type.
    3. Loop through the array and use polymorphism to ask each clock for its time, and print this time along with the clock class name.

DEMO YOUR WORK…

Demo your lab to the Professor

- Show your professor the updates you made to the classes **Clock, WorldClock, and AlarmClock.**
- Show your professor how you used polymorphism to make your code more maintainable.
- Run your files to show that it works <u>correctly</u>
- Prepare to answer some questions.

Grading Rubric

- Show your professor the updates you made to the classes **Clock, WorldClock, and AlarmClock. (50%)**

- Show your professor how you used polymorphism to make your code more maintainable**. (20%)**

- Your program runs correctly. **(15%)**

- Prepare to answer some questions on concepts of inheritance and polymorphism. **(15%)**

References

- Java How to Program, Early Objects Plus MyProgrammingLab with Pearson eText -- Access Card Package, 11/E. Author: Deitel ISBN: 9780134800271