CSC 1302 Principles of Computer Science II

Assignment 6: Java Inheritance

(Due on 11:59 pm, 7/20/2021)

Purpose:

A class in Java can extend another class to become a subclass. When class B extends class A, B becomes subclass (child) and A becomes superclass (parent). The subclass can reuse all the features of the parent class. An interface defines a set of specifications that other classes must implement. Implementing an interface allows a class to become more formal about the behavior it promises to provide.

In this assignment, we will practice how to extend classes and create corresponding objects. We will also practice using the interface. The meaning of implementing an interface is not very different than extending a class but it comes with an additional caveat. When a class implements an interface, it has to provide an implementation of all methods declared inside interface.

Program #1:

Finish Exercises 5, 6, 7, 8 in the end of Chapter 9 (Page657 and 658). A scanned copy is as the following.

5. For the next four problems, consider the task of representing types of tickets to campus events. Each ticket has a unique number and a price. There are three types of tickets: walk-up tickets, advance tickets, and student advance tickets. Figure 9.10 illustrates the types:

- Walk-up tickets are purchased the day of the event and cost $50.
- Advance tickets purchased 10 or more days before the event cost $30, and advance tickets purchased fewer than 10 days before the event cost $40.
- Student advance tickets are sold at half the price of normal advance tickets: When they are purchased 10 or more days early they cost $15, and when they are purchased fewer than 10 days early they cost $20.

Implement a class called `Ticket` that will serve as the superclass for all three types of tickets. Define all common operations in this class, and specify all differing operations in such a way that every subclass must implement them. No actual objects of type `Ticket` will be created: Each actual ticket will be an object of a subclass type. Define the following operations:

- The ability to construct a ticket by number.
- The ability to ask for a ticket's price.
- The ability to `println` a ticket object as a `String`. An example `String` would be `"Number: 17, Price: 50.0"`.

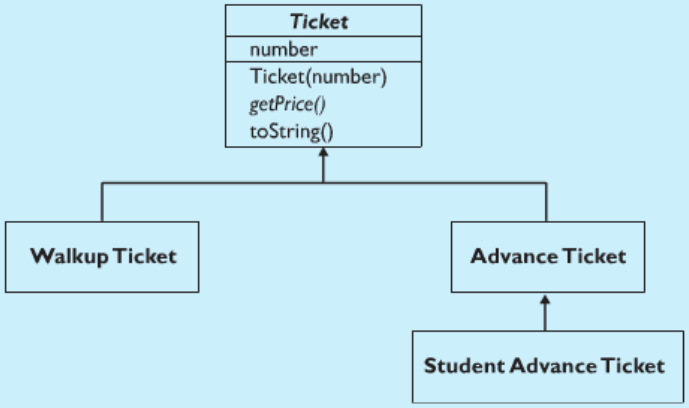

Figure 9.10 Classes of tickets that are available to campus events

6. Implement a class called `WalkupTicket` to represent a walk-up event ticket. Walk-up tickets are also constructed by number, and they have a price of $50.

7. Implement a class called `AdvanceTicket` to represent tickets purchased in advance. An advance ticket is constructed with a ticket number and with the number of days in advance that the ticket was purchased. Advance tickets purchased 10 or more days before the event cost $30, and advance tickets purchased fewer than 10 days before the event cost $40.

8. Implement a class called `StudentAdvanceTicket` to represent tickets purchased in advance by students. A student advance ticket is constructed with a ticket number and with the number of days in advance that the ticket was purchased. Student advance tickets purchased 10 or more days before the event cost $15, and student advance tickets purchased fewer than 10 days before the event cost $20 (half of a normal advance ticket). When a student advance ticket is printed, the `String` should mention that the student must show his or her student ID (for example, `"Number: 17, Price: 15.0 (ID required)"`).

Program #2:

Given partial codes of the Colored interface and Point class as the following:

```
//Colored interface
public interface Colored{
     public String getColor();
}

//Point class
public class Point {
    private int x;
    private int y;

    public Point() {
        this(0, 0);
    }
    public Point(int x, int y) {
        setLocation(x, y);
    }
    public boolean equals(Object o) {
        if (o instanceof Point) {
            Point other = (Point) o;
            return x == other.x && y == other.y;
        } else {
            return false;
        }
    }
    public void setLocation(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}
```

Write ColoredPoint class so that implements the Colored interface and extends Point so that Points have colors. Override toString method to print out the coordinates and color of the point, override equals method so that it compares color as well. And write the necessary constructors, accessors and mutators.

Write a client class and create objects of the ColoredPoint class. Print out the colored point and compare if they are equal.

Criteria:

1. Upload all of the .java and the .class files to the CSc1302 dropbox on http://icollege.gsu.edu.
2. Your assignment will be graded based on the following criteria: (a) Are your programs runnable without errors? (b) Do your programs complete the tasks with specified outputs? (c) Do you follow the specified rules to define your methods and programs? (d) Do you provide necessary comments include the programmer information, date, title of the program and brief description of the program.
3. Make sure that both the .java and .class files are named and uploaded to icollege correctly. If any special package is used in the program, be sure to upload the package too. Should you use any other subdirectory (whatsoever) your program would not be graded, and you will receive a **0 (zero)**.
4. No copying allowed. If it is found that students copy from each other, all of these programs will get **0**.