Homework Assignment #7

CS5004 – Object-Oriented Design
Northeastern University – Silicon Valley
Summer 2019

Due Sunday 07/14 at 11:00pm PDT

Grading: Each programming problem is graded as follows

- A submission which does not compile gets 0.
- A submission which compiles but does something completely irrelevant gets 0.
- A submission which works (partially) correctly, gets (up to) %80 of the total credit.
- %20 is reserved for the coding style. Follow the coding style described in the book.

Formatting: Each class must reside in its own file. If you need to instantiate objects of class `A` in your class `B`, your `B.java` file must import `A.java` first. The names you choose for your classes should match the ones specified in each problem. Also, for each problem i, you must have a `Problem_i.java` file containing (only) the class which has the `main` method. Finally, you must put all files related to problem i into a folder named `problem_i`. The files *must be placed in the right sub-folder*. Make sure that your code complies. See the note in Problem 1 for an example.
Submission: For each problem i submit one `problem_i.zip` file to Blackboard. The zip file should maintain the structure of the sub-folders. See the note in Problem 1 for an example.

Problem 1 [30pts]. Listed next is the skeleton for a class named `Item`. Each item has a name and unique ID number:

```
class Item {
    private String name;
    private int id;
}
```

Flesh out the class with appropriate accessors, constructors, and mutators. The `id`'s are unique and are assigned by your store and can be set from outside the `Item` class.
Next, modify the class so that it implements the `Comparable` interface. The class also

overrides the `compareTo` method. This method imposes an order between instances of the `Item` class depending upon their ID. Test your class by creating an array of sample items and sort them in an ascending order using a sorting method that takes as input an array of type `Comparable`. You do not need to enforce uniqueness of IDs; that is, you can assume that IDs are unique.

Problem 2 [40pts]. Define an interface named `Shape` with a single method named area that calculates the area of the geometric shape:

```
public double area();
```

Next, define a class named `Circle` that implements `Shape`. The `Circle` class should have an instance variable for the radius, a constructor that sets the radius, accessor/ mutator methods for the radius, and an implementation of the area method. Also define a class named `Rectangle` that implements `Shape`. The `Rectangle` class should have instance variables for the height and width, a constructor that sets the height and width, accessor and mutator methods for the height and width, and an implementation of the area method.
The following code should work with your code

```
public static void main(String[] args)
{
    Circle c = new Circle(4); // Radius of 4
    Rectangle r = new Rectangle(4,3); // Height = 4, Width = 3
    ShowArea(c);
    ShowArea(r);
}

public static void ShowArea(Shape s)
{
    double area = s.area();
    System.out.println("The area of the shape is " + area);
}
```