Northeastern Illinois University

CS200 Programming II, Fall 2018

Homework 5

Due: Friday, October 26, 2018 at 10:00 pm

Problem 1 (30 points):

Create a class named `ParseString`, the class should have the following:

- A String instance variable named `str`
- A constructor that accepts one String parameter and sets the instance variable
- A method named `sumOfAll()` which takes no parameters and returns a double
  - This method should find the *sum of all the numeric values* in the instance variable `str`

You may assume the values are separated only by commas (,) or white spaces. The numeric values in `str` may be integers or doubles.

Instructions:

- Download the needed files and look for `TestParseString.java`
- Uncomment the code, but otherwise do not modify the `main()` method in `TestParseString.java`
- Run your class with `TestParseString.java`, and test your implementation

Sample output provided below:

```
Sum of all numbers in “12.3,4.9 6,9 0,8” is:
40.2

Sum of all numbers in “120,8 7.3,6.4 0.1,8.0” is:
149.8
```

Problem 2 (40 points):

Create a properly encapsulated class named `Shape.java`, which has the following:

- A boolean instance variable named `isFilled`
- A String instance variable named `color`
- A default, no-arg constructor that sets `isFilled` to `true` and `color` to "`Green`"
- An overloaded constructor which accepts two parameters, a boolean and a String and sets the instance variables accordingly
- A getter and setter method for each instance variable
- An overridden `toString()` method, which *returns* a String. The String should contain the values of the instance variables in the following format:

```
Filled: true
Color: Green
```

Create a properly encapsulated class named `Circle.java`, which **inherits** from `Shape` and has the following:

- A double instance variable named `radius`
- A default no-arg constructor which sets `radius` to `1`
- An overloaded constructor which takes one double parameter and sets the instance variable radius to the value passed in
- An overloaded constructor which takes three parameters: a double for `radius`, a boolean for `isFilled`, and a String for `color` and sets the instance variables accordingly. Hint: invoke the matching constructor from the superclass
- A getter and setter method for the `radius`
- A method named `getArea()` which calculates and returns the area of the circle
- An overridden `toString()` method that returns a String. The returned String should contain: the value of the radius, the area of the circle, then the result of calling the `toString()` method in the superclass. The return String should be formatted as follows:

```
Radius: 2.67
Area: 22.396099868176275
Filled: true
Color: Green
```

Create a properly encapsulated class named `Rectangle.java`, which also **inherits** from `Shape` and has the following:

- Two double instance variables named `width` and `length`
- A default, no-arg constructor which sets `length` to `2` and `width` to `1`
- An overloaded constructor which takes two double parameters and sets the instance variables `width` and `length` to the values passed in
- Another overloaded constructor which takes four parameters: a double for `width`, a double for `length`, a boolean for `isFilled`, and a String for `color` and sets the instance variables accordingly. Hint: invoke the matching constructor from the superclass
- A getter method for each instance variable
- A void `setLW()` method which takes two parameters `x` and `y`, and sets the `length` variable to the largest value passed in, and the `width` variable to the smallest value passed in. You may assume that `x` and `y` are always positive and have distinct values. Also call this method in your constructors so that your instance variables will always have legal values. Note that a rectangle's length is always greater than it's width
- A method named `getArea()` which calculates and returns the `area` of the rectangle
- An overridden `toString()` method. The returned String should contain: the value of the `width`, the value of the `length`, the `area` of the rectangle, then the result of calling the `toString()` method from the superclass. The return String should be formatted as follows:

```
Width: 3.2
Length: 4.0
Area: 2.8
Filled: false
Color: Red
```

Instructions:

- Download the needed files and look for `TestShape.java`
- Uncomment the code, but otherwise do not modify the `main()` method in `TestShape.java`
- Run your classes with `TestShape.java`, and test your implementation

If you implemented your classes correctly, your output should match the following output from `TestShape`:

```
c1:
Radius: 2.67
Area: 22.396099868176275
Filled: true
Color: Green

c2:
Radius: 3.0
Area: 28.274333882308138
Filled: false
Color: Red

r1:
Width: 2.0
Length: 3.0
Area: 6.0
Filled: true
Color: Blue

r2:
Width: 3.2
Length: 4.0
Area: 12.8
Filled: False
Color: Red
```

Problem 3 (30 points):

What is the exact output of the following program? You should *trace this program by hand*, and save an **output page** and a **tracing page** as pdf files.

```
public class TestFruits
{
   public static void main(String[] args)
   {
      Apple a = new Apple(2);
      System.out.println(Apple.count);
      System.out.println();

      Fruit[] fr = new Fruit[3];
      fr[0] = a;
      fr[1] = new RedDelicious();
      fr[2] = new Fruit(4);
      System.out.println();

      System.out.println("Using array fr:  ");
      for(int  i  =  0;  i  <  fr.length;  i++)
      {
         fr[i].display();
         System.out.println(Apple.count);
         System.out.println();
      }
   }
}
```

```
public class Fruit
{
   public int f;

   public Fruit(int f)
   {
      System.out.println("Fruit");
      this.f  =  f;
   }

    public int getF()
    {
       return this.f;
    }

    public void f1()
    {
       System.out.println("FRUIT f1");
    }

    public void f2()
    {
       f1();
       System.out.println("FRUIT f2");
    }

    public void display()
    {
       System.out.println("display method from Fruit invoked");
       f2();
    }
}
```

```
public class Apple extends Fruit
{
   public static int count = 0;

   public Apple(int a)
   {
      super(a);
      System.out.println("Apple");
      count = count + getF();
   }

   public void f1()
   {
      System.out.println("Apple f1");
   }

   public void f2()
   {
      f1();
      System.out.println("Apple f2");
      super.f1();
   }

   public void display()
   {
      System.out.println("display method from Apple invoked");
      f2();
   }
}
```

```
public class RedDelicious extends Apple
{
   public static int count;

   public RedDelicious()
   {
      super(3);
      System.out.println("RedDelicious");
      count = count + getF();
   }

   public void f1()
   {
      System.out.println("RedDelicious f1");
   }

   public void f2()
   {
      f1();
      System.out.println("RedDelicious f2");
      super.f1();
   }

   public void display()
   {
      System.out.println("display method from RedDelicious invoked");
      f1();
   }
}
```

General Instructions:

No hard copies will be collected. Do not send your files through e-mail! You should submit your work on D2L by the due date. See syllabus for late homework policy.

What to turn in:

There should be four `.java` files, an output page and a tracing page. Place these six files into a zip file and name it `YourNameHW5.zip`, and submit the zip file to the **Homework 5** folder in D2L. Do **not** turn in `.class` files.

Helpful Links:

How to zip files in Mac OS

How to zip files in Windows