CPE 372 Object Oriented Analysis and Design
Exercise 4
Overloading, Overriding and Polymorphism

1. Download the following Java source files from the **Lecture4** subdirectory under **demos:**

   ```
   Square.java
   Triangle.java
   Diamond.java
   Circle.java
   AbstractShape.java
   TextFileReader.java
   ```

2. Also download the two data files, **shapes1.txt** and **shapes2.txt.** Take a look at these files. They are intended to let you create multiple shapes. Each line contains a shape name plus the necessary parameters for that kind of shape, in the same order as in the constructor. For instance:

   SQUARE 120 100 80

   This should create a square with upper left corner (120,100), 80 units on each side.

3. Create a new class called `ShapeReader.java`. This class should extend `TextFileReader.java`, adding a new method called `readShape()`. The `readShape()` method should:

   a) Read a line from a file with a format like **shapes1.txt**;
   b) Parse the line to determine the kind of shape and its parameters;
   c) Call the appropriate constructor (`Square()`, `Triangle()`, etc.);
   d) Return the object as an `AbstractShape` to the calling program.

   For an example that might help, see `PatientFileReader.java` here:

   http://windu.cpe.kmutt.ac.th/cpe111-2018/demos/Lecture13/PatientFileReader.java

   Note that `readShape()` is the only method you need to add to `ShapeReader.` You will **inherit** all the other functionality you need.

4. Create a new class called `ShapeFileTester.java`. You will use this class to test your shape reading and creation, and to explore overriding and polymorphism. This class will have a `main` method:

   ```
   static public void main(String args[])
   ```

   It will expect one command line argument, the name of the shape file to read. (This will be stored in `args[0]`).

5. The logic of this `main` function will be as follows:

   Check that the user has supplied an input filename (look at the **args.length** value which should be 1). If not, print an error message and exit.
   Create an instance of a `ShapeReader`. Try to open the passed file.
   Assuming the file open is successful, enter a loop where you do the following:

   Call `readShape()`. If the function returns null, break out of the loop.
   Print the run time class of the object returned.
   Print the results of calling toString() on the object returned.
   Call calcPerimeter() on the returned shape and print the results.

   When you reach the end of the file and break out of the loop, close the reader and exit.

6. Compile and test `ShapeFileTester`. A sample run should look something like the following:

```
home/goldin: java ShapeFileTester shape1.txt

Trying to open file 'shape.txt'.... success!
   readShape returned an object:  class Circle
      toString: Circle@b1bc7ed
      perimeter: 230
  readShape returned an object:  class Square
      toString: Square@76a43d
      perimeter: 1600
  readShape returned an object:  class Triangle
      toString: Triangle@2318b1
      perimeter: 189

Closing file and exiting
```

7. When you have the program working correctly, modify `Circle.java`, `Square.java`, `Triangle.java` and `Diamond.java`. In each of these classes, add a `toString()` method which will override the one provided by the `Object` class. This method should be declared as follows:

```
public String toString()
```

   Inside the method, create and return a string that gives more useful information about the shape. For example, the `toString()` method for `Square` might return:

   Square at x=100, y=120 with sides 80

8. Recompile and run `ShapeFileTester` again. It should now show you more useful information about each object read from the file and created by `ShapeReader`.

9. Upload all Java files that you have changed. Be sure to follow the coding standards!