CSE 230
PROGRAMMING PROJECT

1. A Shopping Cart

In this exercise you will complete a class that implements a shopping cart as an array of items. The file *Item.java* contains the definition of a class named *Item* that models an item one would purchase. An item has a name, price, and quantity (the quantity purchased). The file *ShoppingCart.java* implements the shopping cart as an array of Item objects.

1. Complete the *ShoppingCart* class by doing the following:
   a. Declare an instance variable *cart* to be an array of items and instantiate *cart* in the constructor to be an array holding the *capacity* number of items.
   b. Fill in the code for the *increaseSize* method. Your code should be similar to that in Listing 7.8 of the text but instead of doubling the size just increase it by 3 elements.
   c. Fill in the code for the *addToCart* method. This method should add the item to the cart and update the *totalPrice* instance variable (note this variable takes into account the quantity).
   d. Compile your class.

2. Write a program that simulates shopping. The program should have a loop that continues as long as the user wants to shop up to two times of size increase of *capacity*. Each time through the loop read in the name, price, and quantity of the item the user wants to add to the cart. After adding an item to the cart, the cart contents should be printed. After the loop print a "Please pay ..." message with the total price of the items in the cart.

Deliverables

- A printout of the complete ShoppingCart.*java* and the simulation program and the final execution.

```
// ************************************************************
//   Item.java
//
//   Represents an item in a shopping cart.
// ************************************************************
import java.text.NumberFormat;
public class Item
{
    private String name;
    private double price;
    private int quantity;

    // -------------------------------------------------------
    //  Create a new item with the given attributes.
    // -------------------------------------------------------
    public Item (String itemName, double itemPrice, int numPurchased)
    {
      name = itemName;
      price = itemPrice;
      quantity = numPurchased;
    }

    // -------------------------------------------------------
    //   Return a string with the information about the item
    // -------------------------------------------------------
    public String toString ()
    {
      NumberFormat fmt = NumberFormat.getCurrencyInstance();

      return (name + "\t" + fmt.format(price) + "\t" + quantity + "\t"
            + fmt.format(price*quantity));
    }

    // -------------------------------------------------
    //   Returns the unit price of the item
    // -------------------------------------------------
    public double getPrice()
    {
      return price;
    }

    // -------------------------------------------------
    //   Returns the name of the item
    // -------------------------------------------------
    public String getName()
    {
      return name;
    }

    // -------------------------------------------------
    //   Returns the quantity of the item
    // -------------------------------------------------
    public int getQuantity()
    {
      return quantity;
    }
}
```

```
// ************************************************************************
//   ShoppingCart.java
//
//   Represents a shopping cart as an array of items
// ************************************************************************
import java.text.NumberFormat;

public class ShoppingCart
{
    private int itemCount;      // total number of items in the cart
    private double totalPrice;  // total price of items in the cart
    private int capacity;       // current cart capacity

    // -----------------------------------------------------------
    //  Creates an empty shopping cart with a capacity of 5 items.
    // -----------------------------------------------------------
    public ShoppingCart()
    { capacity = 5;
      itemCount = 0;
      totalPrice = 0.0;
    }

    // -----------------------------------------------------------
    //  Adds an item to the shopping cart.
    // -----------------------------------------------------------
    public void addToCart(String itemName, double price, int quantity)
    {
    }

    // -----------------------------------------------------------
    //  Returns the contents of the cart together with
    //  summary information.
    // -----------------------------------------------------------
    public String toString()
    {
      NumberFormat fmt = NumberFormat.getCurrencyInstance();

      String contents = "\nShopping Cart\n";
      contents += "\nItem\t\tUnit Price\tQuantity\tTotal\n";

      for (int i = 0; i < itemCount; i++)
          contents += cart[i].toString() + "\n";

      contents += "\nTotal Price: " + fmt.format(totalPrice);
      contents += "\n";

      return contents;
    }

    // -----------------------------------------------------------
    //  Increases the capacity of the shopping cart by 3
    // -----------------------------------------------------------
    private void increaseSize()
    {
    }
}
```

2. Exploring Inheritance

File *Dog.java* contains a declaration for a Dog class. Save this file to your directory and study it—notice what instance variables and methods are provided. Files *Labrador.java* and *Yorkshire.java* contain declarations for classes that extend Dog. Save and study these files as well.

File *DogTest.java* contains a simple driver program that creates a dog and makes it speak. Study DogTest.java, save it to your directory, and compile and run it to see what it does. Now modify these files as follows:

1. Add statements in DogTest.java after you create and print the dog to create and print a Yorkshire and a Labrador. Note that the Labrador constructor takes two parameters: the name and color of the labrador, both strings. Don't change any files besides DogTest.java. Now recompile DogTest.java; you should get an error saying something like

   ```
   ./Labrador.java:7: cannot find symbol
   symbol  : constructor Dog()
   location: class Dog
       {
       ^
   1 error
   ```

   In the Labrador.java, Dog() isn't called anywhere.
   a. What's going on? (Hint: What call must be made in the constructor of a subclass?)
      =>

   b. Fix the problem (which really is in Labrador) so that DogTest.java creates and makes the Dog, Labrador, and Yorkshire all speak.

2. Add code to DogTest.java to print the average breed weight for both your Labrador and your Yorkshire. Use the avgBreedWeight() method for both. What error do you get? Why?

   =>

   Fix the problem by adding the needed code to the Yorkshire class.

3. Add an abstract *int avgBreedWeight()* method to the Dog class. Remember that this means that the word *abstract* appears in the method header after *public*, and that the method does not have a body (just a semicolon after the parameter list). It makes sense for this to be abstract, since Dog has no idea what breed it is. Now any subclass of Dog must have an avgBreedWeight method; since both Yorkshire and Laborador do, you should be all set.

Save these changes and recompile DogTest.java. You should get an error in Dog.java (unless you made more changes than described above). Figure out what's wrong and fix this error, then recompile DogTest.java. You should get another error, this time in DogTest.java. Read the error message carefully; it tells you exactly what the problem is. Fix this by changing DogTest (which will mean taking some things out).

Deliverables

- A UML class diagram for the classes Dog, Labrador, Yorkshire.
- A printout of the complete Dog.java, Labrador.java, Yorkshire.java, DogTest.java without any errors and the final execution.

```
// ****************************************************************
// Dog.java
//
// A class that holds a dog's name and can make it speak.
//
// ****************************************************************
public class Dog
{
    protected String name;

    // ----------------------------------------------------------
    // Constructor -- store name
    // ----------------------------------------------------------
    public Dog(String name)
    {
      this.name = name;
    }

    // ----------------------------------------------------------
    // Returns the dog's name
    // ----------------------------------------------------------
    public String getName()
    {
      return name;
    }

    // ----------------------------------------------------------
    // Returns a string with the dog's comments
    // ----------------------------------------------------------
    public String speak()
    {
      return "Woof";
    }
}
```

```
// ****************************************************************
// Labrador.java
//
// A class derived from Dog that holds information about
// a labrador retriever.  Overrides Dog speak method and includes
// information about avg weight for this breed.
//
// ****************************************************************

public class Labrador extends Dog
{
    private String color; //black, yellow, or chocolate?
    private static int breedWeight = 75;

    public Labrador(String name,  String color)
    {
      this.color = color;
    }

    // -----------------------------------------------------------
    // Big bark -- overrides speak method in Dog
    // -----------------------------------------------------------
    public String speak()
    {
      return "WOOF";
    }

    // -----------------------------------------------------------
    // Returns weight
    // -----------------------------------------------------------
    public static int avgBreedWeight()
    {
      return breedWeight;
    }
}
```

```
// ********************************************************************
// Yorkshire.java
//
// A class derived from Dog that holds information about
// a Yorkshire terrier. Overrides Dog speak method.
//
// ********************************************************************

public class Yorkshire extends Dog
{

    public Yorkshire(String name)
    {
      super(name);
    }

    // -----------------------------------------------------------
    // Small bark -- overrides speak method in Dog
    // -----------------------------------------------------------
    public String speak()
    {
      return "woof";
    }

}

// ********************************************************************
// DogTest.java
//
// A simple test class that creates a Dog and makes it speak.
//
// ********************************************************************

public class DogTest
{
    public static void main(String[] args)
    {
      Dog dog = new Dog("Spike");
      System.out.println(dog.getName() + " says " + dog.speak());

    }
}
```

3. Painting Shapes

In this lab exercise you will develop a class hierarchy of shapes and write a program that computes the amount of paint needed to paint different objects. The hierarchy will consist of a parent class Shape with three derived classes - Sphere, Rectangle, and Cylinder. For the purposes of this exercise, the only attribute a shape will have is a name and the method of interest will be one that computes the area of the shape (surface area in the case of three-dimensional shapes). Do the following.

1. Write an abstract class Shape with the following properties:
   An instance variable shapeName of type String
   An abstract method area()
   A toString method that returns the name of the shape

2. The file *Sphere.java* contains a class for a sphere which is a descendant of Shape. A sphere has a radius and its area (surface area) is given by the formula 4*PI*radius^2. Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the Cylinder class are descendants of the Shape class. A rectangle is defined by its length and width and its area is length times width. A cylinder is defined by a radius and height and its area (surface area) is PI*radius^2*height. Define the toString method in a way similar to that for the Sphere class.

3. The file *Paint.java* contains a class for a type of paint (which has a "coverage" and a method to compute the amount of paint needed to paint a shape). Correct the return statement in the amount method so the correct amount will be returned. Use the fact that the amount of paint needed is the area of the shape divided by the coverage for the paint. (NOTE: Leave the print statement - it is there for illustration purposes, so you can see the method operating on different types of Shape objects.)

4. The file *PaintThings.java* contains a program that computes the amount of paint needed to paint various shapes. A paint object has been instantiated. Add the following to complete the program:
   Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30.
   Make the appropriate method calls to assign the correct values to the three amount variables.
   Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes.

Deliverables

- A UML class diagram for classes Shape, Sphere, Rectangle, Cylinder, Paint and PaintThings;.
- A printout of the complete classes Shape, Sphere, Rectangle, Cylinder, Paint and PaintThings without any errors and the final execution.

```
//********************************************
//  Sphere.java
//
//  Represents a sphere.
//********************************************
public class Sphere extends Shape
{
    private double radius;  //radius in feet

   //---------------------------------
   //  Constructor: Sets up the sphere.
   //---------------------------------
   public Sphere(double r)
   {
      super("Sphere");
```

```
      radius = r;
   }

   //-----------------------------------------
   //  Returns the surface area of the sphere.
   //-----------------------------------------
   public double area()
   {
       return 4*Math.PI*radius*radius;
   }

   //-----------------------------------
   //  Returns the sphere as a String.
   //-----------------------------------
   public String toString()
   {
       return super.toString() + " of radius " + radius;
   }
}

//*******************************************************
//   Paint.java
//
//   Represents a type of paint that has a fixed area
//   covered by a gallon. All measurements are in feet.
//*******************************************************

public class Paint
{
    private double coverage;  //number of square feet per gallon

    //-----------------------------------------
    //  Constructor:  Sets up the paint object.
    //-----------------------------------------
    public Paint(double c)
    {
        coverage = c;
    }

    //-------------------------------------------------
    //  Returns the amount of paint (number of gallons)
    //  needed to paint the shape given as the parameter.
    //-------------------------------------------------
    public double amount(Shape s)
    {
        System.out.println ("Computing amount for " + s);
        return 0;
    }

}
```

```
//********************************************************
//  PaintThings.java
//
//  Computes the amount of paint needed to paint various
//  things. Uses the amount method of the paint class which
//  takes any Shape as a parameter.
//********************************************************

import java.text.DecimalFormat;

public class PaintThings
{
    //-----------------------------------------
    // Creates some shapes and a Paint object
    // and prints the amount of paint needed
    // to paint each shape.
    //-----------------------------------------
   public static void main (String[] args)
   {
      final double COVERAGE = 350;
      Paint paint = new Paint(COVERAGE);

      Rectangle deck;
      Sphere bigBall;
      Cylinder tank;

      double deckAmt, ballAmt, tankAmt;

      // Instantiate the three shapes to paint

      // Compute the amount of paint needed for each shape


      // Print the amount of paint for each.
      DecimalFormat fmt = new DecimalFormat("0.#");
      System.out.println ("\nNumber of gallons of paint needed...");
      System.out.println ("Deck " + fmt.format(deckAmt));
      System.out.println ("Big Ball " + fmt.format(ballAmt));
      System.out.println ("Tank " + fmt.format(tankAmt));
   }
}
```

PROJECT 1 DELIVERABLES

Submit hardcopy and softcopy of the following items to the lab assistant.

- A cover page with the project number, due date, and the names of your Project Team Members.
- Deliverables from the exercise 1, 2 and 3.
- This page, with the appropriate signature and date, indicating that the project has been completely and correctly demonstrated in lab.

LABORATORY SIGNATURE

PROJECT TEAM MEMBERS:

STUDENT NAME _____

STUDENT NAME _____

STUDENT NAME _____

_____ _____

LAB INSTRUCTOR SIGNATURE **DATE**