Chapter 7: Inheritance
Lab Exercises

Exploring Inheritance

Create a folder `dogProject`. Copy the code below into a new file `Dog.java` which contains a declaration for a `Dog` class. Study it—notice what instance variables and methods are provided. Copy the code for files `Labrador.java` and `Yorkshire.java` which contain declarations for classes that extend `Dog`. Save and study these files as well.

Copy the code for `DogTest.java` which contains a simple driver program that creates a dog and makes it speak. Study `DogTest.java` 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:16: cannot find symbol
   symbol  : constructor Dog()
   location: class Dog
       {
       ^
   ```

   If you look at line 16 of `Labrador.java` it's just a {, and the compile error "`constructor Dog()`" isn't called anywhere in this file.

   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. Remove the comments for the code in the method `avgBreedWeight()` in the `Labrador` class. 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).

```
// ****************************************************************
// Dog.java
//
// A class that holds a dog's name and can make it speak.
//
// ****************************************************************
public class Dog
{
    private 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 int breedWeight = 75;

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

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

    // -----------------------------------------------------------
    // Returns weight
    // -----------------------------------------------------------
    /*
    public 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 "yap yap";
    }

}



// ****************************************************************
// 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());

    }
}
```

Employee

Create a folder `EmployeeProject`. Copy the files `Firm.java, Staff.java, StaffMember.java, Volunteer.java`, `Employee.java, Executive.java`, and `Hourly.java` from below. The program illustrates inheritance and polymorphism. In this exercise you will add one more employee type to the class hierarchy. The employee will be one that is an hourly employee but also earns a commission on sales. Hence the class, which we'll `name Commission`, will be derived from the `Hourly` class.

1. Study the code for each class. On your assignments sheet, draw a hierarchy diagram for the classes: `StaffMember, Volunteer, Employee, Executive, Hourly and Commission`

2. Write a class named `Commission` with the following features:

- It extends the `Hourly` class.
- It has two instance variables (in addition to those inherited): one is the total sales the employee has made (type `double`) and the second is the commission rate for the employee (the commission rate will be type `double` and will represent the percent (in decimal form) commission the employee earns on sales (so .2 would mean the employee earns 20% commission on sales)).
- The constructor takes 6 parameters: the first 5 are the same as for `Hourly` (name, address, phone number, social security number, hourly pay rate) and the 6th is the commission rate for the employee. The constructor should call the constructor of the parent class with the first 5 parameters then use the 6th to set the commission rate.
- One additional method is needed: public void addSales (double tSales) that adds the parameter to the instance variable representing total sales.
- Override the `pay` method from the `Hourly` class. The method should return the total pay which will be computed by calling the `pay` method of the parent class to compute the pay for hours worked then adding to that the pay from commission on sales. (See the pay method in the Executive class.) The total sales should be set back to 0 (note: you don't need to set the `hoursWorked` back to 0—why not?).

3. To test your class, update `Staff.java` as follows:

- Increase the size of the array to 8.
- Add two commissioned employees to the `staffList`—make up your own names, addresses, phone numbers and social security numbers. Have one of the employees earn $6.25 per hour and 20% commission and the other one earn $9.75 per hour and 15% commission.
- For the first additional employee you added, put the hours worked at 35 and the total sales $400; for the second, put the hours at 40 and the sales at $950.

4. Compile and run the program. Make sure it is working properly by calculating the pay for each added employee by hand and comparing to your output.

```
//********************************************************************
//  Firm.java       Author: Lewis/Loftus
//
//  Demonstrates polymorphism via inheritance.
//********************************************************************

public class Firm
{
   //-----------------------------------------------------------------
   //  Creates a staff of employees for a firm and pays them.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Staff personnel = new Staff();

      personnel.payday();
   }
}
```

```
//********************************************************************
//  Staff.java       Author: Lewis/Loftus
//
//  Represents the personnel staff of a particular business.
//********************************************************************

public class Staff
{
   StaffMember[] staffList;

   //-----------------------------------------------------------------
   //  Sets up the list of staff members.
   //-----------------------------------------------------------------
   public Staff ()
   {
      staffList = new StaffMember[6];

      staffList[0] = new Executive ("Sam", "123 Main Line",
         "555-0469", "123-45-6789", 2423.07);

      staffList[1] = new Employee ("Carla", "456 Off Line",
         "555-0101", "987-65-4321", 1246.15);
      staffList[2] = new Employee ("Woody", "789 Off Rocker",
         "555-0000", "010-20-3040", 1169.23);

      staffList[3] = new Hourly ("Diane", "678 Fifth Ave.",
         "555-0690", "958-47-3625", 10.55);

      staffList[4] = new Volunteer ("Norm", "987 Suds Blvd.",
         "555-8374");
      staffList[5] = new Volunteer ("Cliff", "321 Duds Lane",
         "555-7282");

      ((Executive)staffList[0]).awardBonus (500.00);

      ((Hourly)staffList[3]).addHours (40);
   }

   //-----------------------------------------------------------------
   //  Pays all staff members.
   //-----------------------------------------------------------------
   public void payday ()
   {
      double amount;

      for (int count=0; count < staffList.length; count++)
      {
         System.out.println (staffList[count]);

         amount = staffList[count].pay();  // polymorphic

         if (amount == 0.0)
            System.out.println ("Thanks!");
         else
            System.out.println ("Paid: " + amount);

         System.out.println ("-----------------------------------");
      }
   }
}
```

```
//********************************************************************
//  StaffMember.java       Author: Lewis/Loftus
//
//  Represents a generic staff member.
//********************************************************************

public abstract class StaffMember
{
   protected String name;
   protected String address;
   protected String phone;

   //-----------------------------------------------------------------
   //  Sets up a staff member using the specified information.
   //-----------------------------------------------------------------
   public StaffMember (String eName, String eAddress, String ePhone)
   {
      name = eName;
      address = eAddress;
      phone = ePhone;
   }

   //-----------------------------------------------------------------
   //  Returns a string including the basic employee information.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = "Name: " + name + "\n";

      result += "Address: " + address + "\n";
      result += "Phone: " + phone;

      return result;
   }

   //-----------------------------------------------------------------
   //  Derived classes must define the pay method for each type of
   //  employee.
   //-----------------------------------------------------------------
   public abstract double pay();
}
```

```
//********************************************************************
//  Volunteer.java       Author: Lewis/Loftus
//
//  Represents a staff member that works as a volunteer.
//********************************************************************

public class Volunteer extends StaffMember
{
   //-----------------------------------------------------------------
   //  Sets up a volunteer using the specified information.
   //-----------------------------------------------------------------
   public Volunteer (String eName, String eAddress, String ePhone)
   {
      super (eName, eAddress, ePhone);
   }

   //-----------------------------------------------------------------
   //  Returns a zero pay value for this volunteer.
   //-----------------------------------------------------------------
   public double pay()
   {
      return 0.0;
   }
}
```

```
//********************************************************************
//  Employee.java       Author: Lewis/Loftus
//
//  Represents a general paid employee.
//********************************************************************

public class Employee extends StaffMember
{
   protected String socialSecurityNumber;
   protected double payRate;

   //-----------------------------------------------------------------
   //  Sets up an employee with the specified information.
   //-----------------------------------------------------------------
   public Employee (String eName, String eAddress, String ePhone,
                    String socSecNumber, double rate)
   {
      super (eName, eAddress, ePhone);

      socialSecurityNumber = socSecNumber;
      payRate = rate;
   }

   //-----------------------------------------------------------------
   //  Returns information about an employee as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = super.toString();

      result += "\nSocial Security Number: " + socialSecurityNumber;

      return result;
   }

   //-----------------------------------------------------------------
   //  Returns the pay rate for this employee.
   //-----------------------------------------------------------------
   public double pay()
   {
      return payRate;
   }
}
```

```
//********************************************************************
//  Executive.java       Author: Lewis/Loftus
//
//  Represents an executive staff member, who can earn a bonus.
//********************************************************************

public class Executive extends Employee
{
   private double bonus;

   //-----------------------------------------------------------------
   //  Sets up an executive with the specified information.
   //-----------------------------------------------------------------
   public Executive (String eName, String eAddress, String ePhone,
                     String socSecNumber, double rate)
   {
      super (eName, eAddress, ePhone, socSecNumber, rate);

      bonus = 0;  // bonus has yet to be awarded
   }

   //-----------------------------------------------------------------
   //  Awards the specified bonus to this executive.
   //-----------------------------------------------------------------
   public void awardBonus (double execBonus)
   {
      bonus = execBonus;
   }

   //-----------------------------------------------------------------
   //  Computes and returns the pay for an executive, which is the
   //  regular employee payment plus a one-time bonus.
   //-----------------------------------------------------------------
   public double pay()
   {
      double payment = super.pay() + bonus;

      bonus = 0;

      return payment;
   }
}
```

```
//********************************************************************
//  Hourly.java       Author: Lewis/Loftus
//
//  Represents an employee that gets paid by the hour.
//********************************************************************

public class Hourly extends Employee
{
   private int hoursWorked;

   //-----------------------------------------------------------------
   //  Sets up this hourly employee using the specified information.
   //-----------------------------------------------------------------
   public Hourly (String eName, String eAddress, String ePhone,
                  String socSecNumber, double rate)
   {
      super (eName, eAddress, ePhone, socSecNumber, rate);

      hoursWorked = 0;
   }

   //-----------------------------------------------------------------
   //  Adds the specified number of hours to this employee's
   //  accumulated hours.
   //-----------------------------------------------------------------
   public void addHours (int moreHours)
   {
      hoursWorked += moreHours;
   }

   //-----------------------------------------------------------------
   //  Computes and returns the pay for this hourly employee.
   //-----------------------------------------------------------------
   public double pay()
   {
      double payment = payRate * hoursWorked;

      hoursWorked = 0;

      return payment;
   }

   //-----------------------------------------------------------------
   //  Returns information about this hourly employee as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result = super.toString();

      result += "\nCurrent hours: " + hoursWorked;

      return result;
   }
}
```