LAB9 Inheritance

Objective:

In the lecture we described inheritance features of Object Oriented Programming. We will define the rules of defining subclasses of programmer-defined classes in this lab. Explanation of how object-oriented programming allows classes to inherit commonly used state and behavior from other classes is also provided.

Understanding Inheritance

In object-oriented programming, inheritance is a way to form new classes using classes that have already been defined. The new classes, known as derived classes or subclass, take over (or inherit) attributes and behavior of the pre-existing classes, which are referred to as base classes (or superclass or ancestor classes). It is intended to help reuse existing code with little or no modification. Inheritance is also sometimes called specialization, because the inheritance relationship represents a hierarchy between classes. For instance, “apple”, "orange", "mango" and many others are specializations of a "fruit". An advantage of inheritance is that modules with sufficiently similar interfaces can share a lot of code, reducing the complexity of the program.

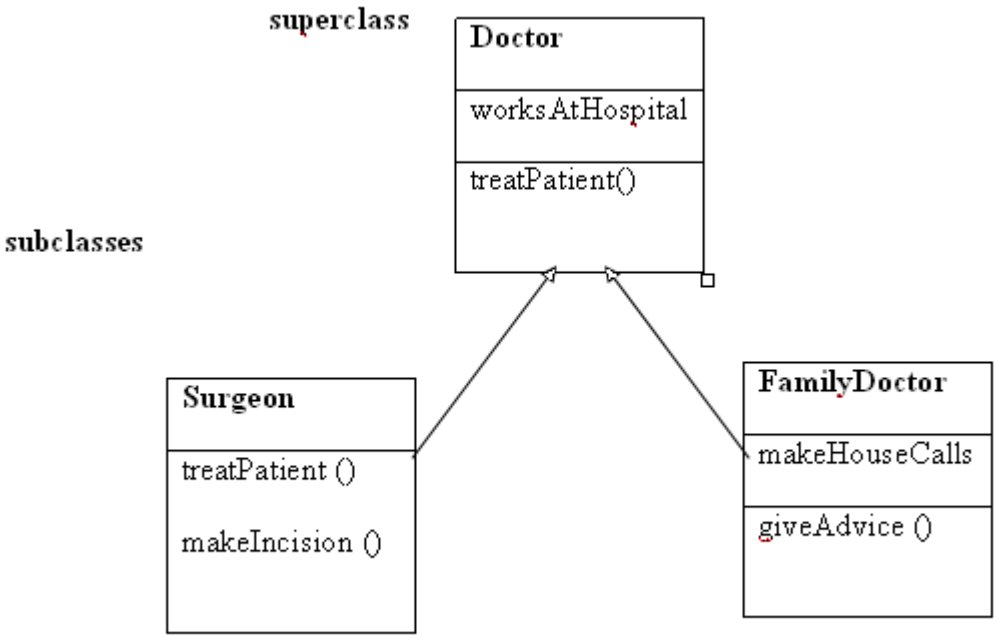
An important thing to keep in mind about inheritance and accessibility is that a derived class does not have access to its base classes' private members. Private members can be accessed only by the immediate type in which they're defined. Protected & public members, however, can be accessed within an inheritance hierarchy. In Java, we say that the subclass extends the superclass.

An Inheritance Example

```
public class Doctor
{

  boolean worksAtHospital;

    Void treatPatient()
      {
        // perform a checkup
      }
}
```

```
public class FamilyDoctor extends Doctor
{
   boolean makesHouseCalls;

    void giveAdvice()
    {
              // give homespun advice
    }
}
```

```
public class Surgeon extends Doctor
{
void treatPatient()
        {
          // performs surgery
        }

void makeIncision()
        {
          // make incision (yikes!)
        }
}
```



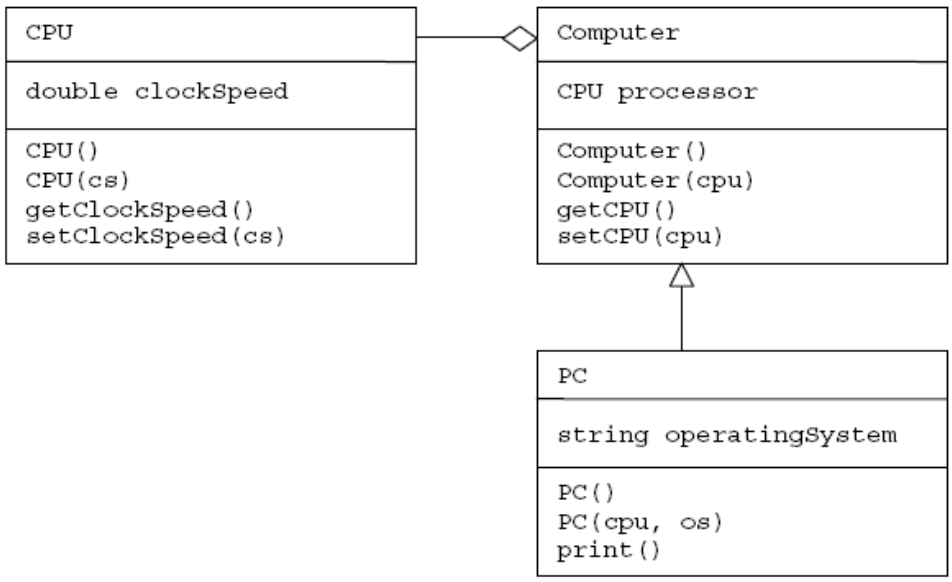
EXERCISE

Question1. Write a java program that includes the class interfaces, class definitions, and code that will test your classes. Write a main() function along with other functions/procedures, as described below:

a) You shall implement classes that support the following relationships:

- A Computer has a central processing unit (CPU).
- A PC is a type of Computer.

b) These class relationships are shown in the following diagram.



c) The CPU class is described as follows:

- One attribute:

  clockSpeed – a double value representing the clock speed of the CPU object. Note: clock speed of a modern computer may be something like 2.5GHz, so a value you could use for testing purposes is 2.5.

- Two constructors:

  A **default constructor**.

  A **constructor** that initializes the attribute, where cs is the clockSpeed.

- Two methods:

  getClockSpeed() – return the clockSpeed value associated with the CPU object.

  setClockSpeed(cs) – change the clockSpeed value associated with the CPU object, where cs is the new value.

d) The Computer class is described as follows:

- One attribute:
  processor – a CPU object.
- Two constructors:
  A **default constructor**.
  A **constructor** that initializes the attribute, where cpu is a CPU object.
- Two methods:
  getCPU() – return the CPU object associated with the Computer object
  setCPU(c) – change the CPU object associated with the Computer object, where c is the new CPU object.

e) The PC class is described as follows:

- One attribute:
  operatingSystem – a string value representing the name of the operating system used on the PC. This value may be something like "Microsoft Windows XP", "Linux", or "Mac OS X".
- Two constructors:
  A **default constructor**.
  A **constructor** that initializes the attributes in the PC and Computer classes, where cpu is the CPU object and os is the operatingSystem name.
- One method:
  print() – display to console output all of the information associated with this PC object. That is, display the operatingSystem value, the Computer object information, and
  the CPU object information. IMPORTANT: The Computer and CPU classes do NOT have print() methods. Do NOT add print() methods to these two classes. Instead, you must use the appropriate accessor (get) method to obtain each value to be displayed.

f) The code associated with your main() function should test your three class definitions. It is important that this code use each constructor defined in each class.

Question2. Develop a program that helps the user to draw basic geometrical shapes using Turtles. The program specification is as follow.

a) Develop a class named JPoint which stores x and y coordinates of stored points with corresponding getters & setters.

b) Develop an abstract Shape class that contains following members.

- **turtle** : Used to store instance of associated circle with shape
- **world** : Used to store world in which shape is drawn
- **perimeter** : Used to store perimeter of shape
- **area** : Used to store area of shape.

It has following abstract methods:

- **abstract public void draw( )**
  To draw shape.
- **abstract public double getPerimeter( )**
  To get perimeter of shape.
- **abstract public double getArea()**
  To get area of shape.
- **abstract public void setDisplay(JPoint p, World w)**
  Here p is used to set initial position of turtle and w used to set world used by shape.

c) Develop a class named Quadrilateral which extends Shape class.

It has following member

- **JPoint Points[]**
  Used to Store four points of Quadrilateral

It should have all the corresponding constructors and methods for valid operation of quadrilateral.

d) Develop a Triangle class which extends Shape class.

It has following members

- **JPoint Points[]**
  Used to Store three points of Triangle
- **int a , b , c**
  Used to store side lengths of Triangle

It should have all the corresponding constructors and methods for valid operation of a Triangle.

e) Develop an Ellipse class which extends Shape class

It has following members

- **JPoint Center**
  Used to Store center point of Ellipse
- **int Rx,Ry**
  Used to store lengths of major and minor axes of Ellipse

It should have all the corresponding constructors and methods for valid operation of an Ellipse.

Test your program by creating one object of each type and calculate the areas and perimeters.

Question 3. Develop a program that contains two classes.The specification of both classes is as follow:

a) Data Class

Data class has following members

- **protected String data**
  This field stores any type of user data

Data class has following member functions

- One argument constructor which takes String object to initialize data.
- **public void setData(String d)**
  To set data.
- **public String getData()**
  To get data.
- **public String toString()**
  To override the toString() method of base class Object. It simply returns data field.

b) EncryptedData Class

This class extends the Data class.

This class has following data members

- **protected String encryptedarray[]**
  Used to encrypt data.
- **protected boolean encrypted**
  To show whether data is encrypted or not.

Data class has following member functions

- One argument constructor which takes String argument to initialize superclass Data
- **public void encrypt( )**
  This function encrypts the data.
- **public void decrypt( )**
  This function decrypts the already encrypted data.
- **public String storeData()**
  This function returns a combination of String which represents EncryptedData object and stores it in a data file.

- **public void loadData(String data)**
  This function loads data from a valid data file.

Encryption Technique :

This is an encryption technique which is a combination of two steps. These steps are as follow:

Step 1: Change the state of each and every bit of lower byte of character, reverse the order of bytes in character and toggle alternative bits of new lower byte.

Step 2: Create an array of Strings. Length of Array is equal to ceil of square root of length of stored data. Any String in array can store maximum characters equal to length of array. Store data column wise is array as first character of data goes to first character of first array , second goes to first character of second array and so on until first columns are finished then start filling second columns of all arrays until data and array both are finished. If there are some character fields in arrays remained but data is exhausted then fill the remaining ones with '*'. You can store data row wise back to data field to store encrypted data.

Suppose data field is "ABCDEFGHIJ"
After first step suppose encrypted data is "KLMNOPQRST"
Then string arrays would be filled like this.

First String	K	O	S	*
Second String	L	P	T	*
Third String	M	Q	*	*
Fourth String	N	R	*	*

Data will be created again as "KOS*LPT*MQ"

Test your classes by creating two EncryptedData classes. First encrypts data and second uses that to decrypt data and display actual data. Use storeData( ) and loadData( ) functions to pass data between two classes. Don't overload a constructor or a function to copy objects.

Hint:

// Use following standard string functions

char charAt (index) : It returns the character at specified index where index of first character is 0

int length( ) : It returns the total number of characters in a String object.

replace(dest,src) : This function creates a new String with same data and replaces all occurance of destination character by src character in the new String and returns the new String without modifying the String object.

Lab9 Inheritance -- Hints

Question # 2

Formulas:

Ellipse

Centre point: a,b
Radius:
Rx: Ellipse width
Ry: Ellipse height
Area: Pi* a* b
Perimeter: Pi√ 2 (a^2+b^2)
To Draw:
Formula
Then draw the ellipse on each point of the circle using the following equation
(a+Rx)*Cos(angle in radian) , (b+Ry)*(angle in radian)

Triangle

Area : √s(s-a)(s-b)(s-c)
Where s = (a+b+c)/2.0 where a, b and c is length of three sides respectively

Perimeter: a+b+c

Quadrilateral

Area: Sum of the two triangles of quadrilateral returns sum of quadrilateral
Perimeter: Sum of all sides i.e. a+b+c+d
To Draw: use turtle function turnToFace(x,y) both to draw quadrilateral and triangle

Note: Solve part (d) first, and then part (c)

Question # 3

Character functions

reverseBytes(string)

For further details view the java documentation which is available at the following link:
http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Character.html