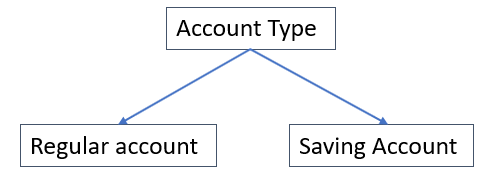
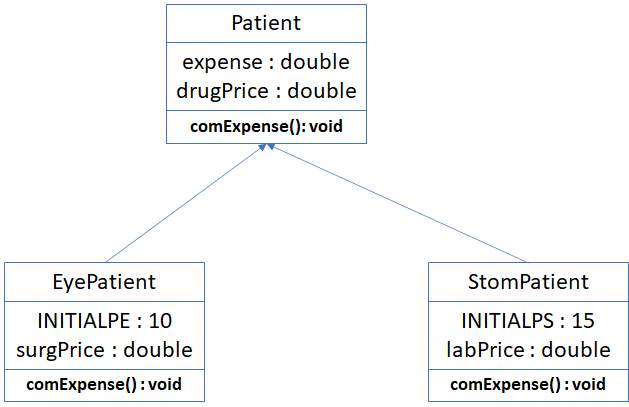
**Object Oriented Programming with Java**

**Lab Practice:6**

1. Consider the following class hierarchy.



* Declare an abstract class called Account which has the attributes Account Id, Name, Address, and balance and an abstract method called calculateInterest.
* Implement the abstract method in the class Savings Account. Use any simple interest formula to calculate the interest. (You can assume that you will get time/number of years as input to the method). You can leave the subclass Checking Account as abstract.
* Finally, compile your program.

1. Consider the following class hierarchy:

Write the code that implements these classes and their methods in Java, each class should contain the following:

(a) Default constructor.

(b) A constructor for an object with its all attributes.

(c) computeExpenses () in Patient class is simply an abstract method that needs to be overridden in the subclasses.

(d) computeExpenses() in the subclasses overrides the inherited one as follows:

- computeExpenses() for the EyePatient stores the total expenses in the expense data member as follows:

-Expense = price of the Surgery+initial Payment +price of drugs

-computeExpenses () for the StomPatient stores the total expenses in the expense data member as follows:

-Expenses = initial Payment +price of drugs+ Lab.

(e) Create a testing class that declares a 1-D array called mClinic then enables 100 patient’s objects to be inserted in mClinic whether they are of Eye-Patient type or of Stomach-Patient type.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*additional Questions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Topic: equals() method and Equality test, Shallow copy vs. Deep copy

1. Consider the following code:

public class Student {  
  
 private String name;  
 private int age;  
  
 public Student(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
 public String getName() {  
 return name;  
 }  
 public int getNum() {  
 return age;  
 }  
}

This is a “Student” class with constructor and getter methods for “name” and “age” attributes defined. First, Student "a" is created through the code below.

Student a = new Student("Lee", "20");

Then I define “b” and “c” Student objects to copy "a" in different ways.

Student b = a;

Student c = new Student("Lee", "20");

Finally, I checked whether the copy was successful through the code below.

System.*out*.println("a == b : " + (a==b));  
System.*out*.println("a.equals(b) : " + (a.equals(b)));  
  
System.*out*.println("a == c : " + (a==c));  
System.*out*.println("a.equals(c) : " + (a.equals(c)));  
  
System.*out*.println("a == d : " + (a==d));  
System.*out*.println("a.equals(d) : " + (a.equals(d)));

After predicting what the result will be, explain the reason for the actual result along with the difference between == and equals() methods.

2. Consider the following code:

String str1 = "Hello";

String str2 = str1;

String str3 = new String("Hello");

I want to copy str1, so I have defined str2, str3 fields in different ways. As in the previous question, I checked the copy result with the code below again this time.

System.*out*.println("str1 == str2 : " + (str1==str2));  
System.*out*.println("str1.equals(str2) : " + (str1.equals(str2)));  
System.*out*.println("str1 == str3 : " + (str1==str3));  
System.*out*.println("str1.equals(str3) : " + (str1.equals(str3)));

Check to see if you get the same conclusion as the previous result, and if you get a different result, explain why.

(Hint : Check the hash code of each object and describe it according to the result.)

I thought the answer below was the most complete answer to this question and created a problem. However, for students new to OOP and unfamiliar with java, I thought the problem might be too difficult. So, like the hint, it would be sufficient to simply explain the relationship between hash code and equals method and explain the difference through the actual output hash code.

*The "equals(Object obj)" method of the Object class belonging to java.lang is designed to simply compare the references of obj and this (self) with == and return the result, and does not compare the contents. However, the String class has an overriding equals() method to compare two strings for equality.*

Topic: toString() method

3. Consider the following code:

public class Student {  
  
 private String name;  
 private String age;  
  
 public Student(String name, String age) {  
 this.name = name;  
 this.age = age;  
 }  
 public String getName() {  
 return name;  
 }  
 public String getNum() {  
 return age;  
 }  
}

This is my result of creating the object "a" of the Student class and calling the toString() method.

Student a = new Student("Lee", "20");

System.out.println(a.toString());

Result: Student@4eec7777

Modify the toString() method by overriding it to show the “name” and “age” attributes of the Student object.