Friday, 9 August 2019

Session-4

OOP-2

Q Write a program to maintain data about different accounts in a

bank. Create a class Account which has following members:

private int accNo private String accName private String accBal

Parameterized constructor to initialize all instance variables. Getter

and setter methods for all members.

Create a class Saving derived from Account having following

members private double rateOfInt - Rate of Interest variable

Parameterized constructor to initialize accNo, accName, accBal and

rateOfInt Setter and getter method for rateOfInt instance variable

Create a class Current derived from Account having following

members private double rateOfInt - Rate of Interest variable

Parameterized constructor to initialize accNo, accName, accBal and

rateOfInt Setter and getter method for rateOfInt instance variable

Create a class TestAccount having main method. Scan data from

user for Saving object. Create an object of Saving class and print

initialized data. Scan data from user for Current object. Create an

object of Current class and print initialized data.

**I believe the above was already completed as 3-5, and we just need to answer questions.**

- 1. Can we achieve run time polymorphism by data members?

**I’m not sure how to answer. The data members themselves could be upcasted (declared as base class but instantiated as child class). Then method overrides will demonstrate runtime polymorphism.**

- 2. Can I declare a class as private?

**Not an outer class. That would render it pointless if you could (you could leave default package-private access though).**

- 3. What is the difference b/w Abstraction and Encapsulation.

**Abstraction is about hiding implementation from a user to make something easier to use. It can also help with modularity.   
Encapsulation is about making classes distinct from each other, related to having a single class with single responsibility. Specifically, it allows making fields or methods private, restricting access. This may have more to do with security than ease of use, distinguishing it from abstraction.**

- 4. Method Overloading rules? Can we overload the super class method in

sub class. Discuss with an example.

**You can override the super method, as long as it was not final.**

- 5. Method overriding rules.

- 1. You have the following code in a file called Test.java

-

class Base{

public static void main(String[] args){

System.out.println("Hello");

} }

public class Test extends Base{}

-

- What will happen if you try to compile and run this?

- a. It will fail to compile.

- b. Runtime error

- c. Compiles and runs with no output.

- d. Compiles and runs printing "Hello”

**d) It compiles and runs as expected**

-

- 2. Name the access modifier which when used with a method, makes it available to all

- the classes in the same package and to all the subclasses of the class.

**package-private access**

- 3. What is the result of attempting to compile and run this?

-

class Base{

String s = "Base";

String show() {

return s;

}

}

class Derived extends Base{

String s = "Derived";

}

public class Test {

void print(Base b) {

System.out.println(b.show());

}

void print(Derived d){

System.out.println (d.show());

}

public static void main(String[] args){

Test t = new Test();

Base b = new Derived();

t.print(b);

}

}

- a. Code will not compile

- b. Run time error

- c. Will compile and run printing "Derived“

- d. Will compile and run printing "Base"

**Assuming these are in separate files, (c).**

- 4. Which of the following are correct. Select the one correct answer.

- a. An import statement, if defined, must always be the first non-comment statement of

- the file.

- b. private members are accessible to all classes in the same package.

- c. An abstract class can be declared as final.

- d. Local variables cannot be declared as static

**(a) is true**

- 5. If a base class has a method defined as void method() { }

- Which of the following are legal prototypes in a derived class of this class. Select the

two

- correct answers.

- a. void method() { }

- b. int method() { return 0;}

- c. void method (int i) { }

- d. private void method() { }

**b, c**

-

-

- Inheritance : transfer from a parent to child, super class and refer as sub class.

- private members of class can be access only in the super class and not in sub class

- protected : private and the inherited class

- Types Of Inheritances

- 1. Single – Any one class has only one super class

- 2. Multiple - More than one super goes to a sub class - Not supported......

- 3. Multilevel - One super, one sub and again the child of the sub class

- 4. Hierarchal - One super class has multiple children

- in my super class if I have a constructor and I want to call it, in my child class.

- // super() keyword - This is only for constructor