OOP's Concept

Subject : _____ Date

Java is programming paradigm where everything is represented as an object.

→ a class is a template, blueprint or contract that defines what how an object's data fields and methods will be.

General OOP's Concepts

* Object
* Class
* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

Encapsulation

→ Binding or wrapping code and data together into a single unit.

→ A java class is a simple example of encapsulation. Java bean is the fully encapsulated class [because all data mems are private and can be accessed only using its getters & setters]

→ It provides security and enables data hiding Usually refered as "black box" ["ready-only or write-only]

Subject : _____ Date

Abstraction

→ captures only those details about an object that are relevant to the current perspective, so that the programmer can focus on a few concepts at a time.

→ Hiding internal details and showing functionality.

→ In java, we use abstract class and interface to achieve abstraction.

→ A powerful way is through the use of hierarchical classifications.

* Difference between Encapsulation & Data Abstraction.

⇒ Abst focuses on the outside view while encapsulation prevents client from viewing inner view.

→ Abstraction solves the prob in the design side while encapsulation is the implementation.

→ Encapsulation is grouping up your abstraction.

* Concrete class.

A class is called concrete if it does not contain any abstract method and implements all abstract method inherited from abstract class or interface it has implemented or extended.

Subject : _____ Date

Inheritance

→ the ability to create classes that share the attributes and methods of existing classes, but with more specific features.

→ Provides code reusability

→ Used to achieve runtime polymorphism

* Java doesn't support multiple inheritance

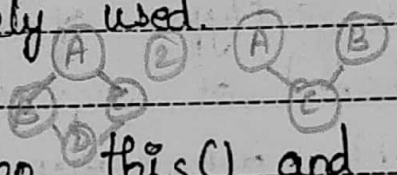
→ Interfaces doesn't facilitate inheritance [not same]

→ why omitted in Java (ambiguity)

• simplicity (dynamic loading of classes)

• rarely used.

(1) diamond problem



common methods

* Difference between this() and super().

→ this is a reference to the current object whereas super is a reference to access members of parent class.

→ super is primarily used to initialize base class members within derived class constructo

* Uses of this()

→ for accessing mem variables if local var have same name

→ constructor chaining

→ passing itself to some method.

Subject : _____ Date

* Are constructors inherited? Can a subclass call the parent's class constructor?

→ No, cannot inherit a constructor.

→ The ability to override a superclasses constructor would erode encapsulation.

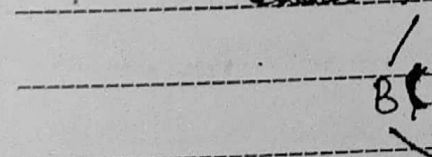
→ also constructor name should be same as class name and so cannot have super class name as const name.

* cannot reduce the visibility of the inherited or overridden method.

* What is covariant return type?

→ states that return type of overriding method can be subtype of the return type declared in method of superclass.

→ applicable only after jdk 1.5 [Java 5]



class SuperClass()
{ A meth_name(); }

class SubClass extends SuperClass
{ B/C meth_name(); }

Polymorphism

→ ability of an object to take different forms

→ 2 types * compile time (or) method overloading

* runtime (or) method overriding

→ note that method signatures should be same for both super & sub classes

Subject: _____ Date

* UpCasting: reference variable of parent class refers to the object of child class A a = new BC();

reference variable [refer to variables] → Object Class [refer to methods]

→ runtime polym cannot be achieved by data members (variables)

* can we overload static method?

→ Yes, can have two or more static methods but should differ in Input parameters.

* cannot overload two methods that only differs by static keyword.

* can we override static method?

→ No, because when we declare static methods with same signature in derived class, there won't be any run-time polymorphism

reason

* If a derived class defines a static method with same signature as static method in base class, the method in the derived class hides the method in the base class.

* realtime example for overriding: Bank which provides method to get rate of interest.

Subject: ---------- Date

* An instance method (non-static) cannot override a static method, and a static method cannot hide an instance method.

```
class Base {                                  class subClass extends Base {
    (static meth in base cls will be hidden)      (cannot override)
    pub static void display();                    public void display();
    (non-static meth will be overriden)           (cannot hide)
    public void print();                          public static void print();
}                                             }
```

* In a subclass, we can overload the methods inherited from superclass. (ie) neither hidden nor overriden. They are new methods unique to subclass.

* cannot override private methods. because private methods cannot be inherited.

* Exception handling in overriden methods

2 senarios: (applicable only for exceptions that are thrown)

→ when super class method does not declare an exception

- subclass cannot declare checked exception but can declare unchecked exception (ArithmeticException or ArrayIndexOutOfBoundsException)

→ when superclass method declares an exception

- subclass can either declare the same exception or can be with no exceptions or can declare subclass exception
- but subclass cannot declare parent exception to specified in superclass. ~~rather when super class~~