Saturday, April 30, 2022 9:28 AM

private , package , import , instanceof

instanceof is an operator like -

operators that give result in boolean true or false --

a==b } true or false
a<b } true, false
a>b
a!=b
a>=b
a<=b
a && b
a || b
!a

instanceof operator gives result in boolean } true or false

a instanceof b } if reference "a" is type of "b" then true

____

what is the difference between int and Integer !!!

____

StringBuffer , StringBuilder ----------

____

extends

____

super() --------- this is calling the no param constructor of super class

This can be used in sub class constructor , if used then it is the first line in the sub class constructor

____

abstract class = 0 objects can be created
abstract method = the method has no implementation

Interfaces in Java ------------

They are similar to abstract classes .

What is the need for abstract class ?

Abstract class is the base of the hierarchy !!!

Classes towards the base of hierarchy (super classes)are more **generalized**

Classes towards the leaf(sub classes) are more **specialized**

Vehicle
| | |
TW FW AnimalDriven

```
   Person
   |    |     |
 student Employee  Patient
```

```
   Shape
   |   |     |
   Circle  Triangle  Trapezium
```

```
         LivingBeings
          |             |
      Animals        Plants
    |         |
   reptile   mammal
                 |      |
                 Bat    Humans
```

HW : Type the code and run it

Ex1 ----- Create a hierarchy of Shape

Circle , Rectangle

Radius   length,breadth

areaOfcircle   areaofreact

EstateAgent

Main

showCostOfPlot(Shape shape)

shape.area() * per_square_feet_cost

Advantage of abstract classes and methods is to allow RUN TIME POLYMORPHISM

So that the User of the hierarchy can write methods that are LOW MAINTAINENCE

Even if the hierarchy has more classes the methods of the user will run without change and accommodate new classes of the hierarchy without any change !!!

Interfaces ---------- Used for RUN TIME POLYMORPHISM in hierarchy

So that the user of hierarchy can write methods that will not change even if new classes are added to the hierarchy

Interface	**abstract class**
this is an interface type	is a class type
Can be added to a package	can be added to a package
interface keyword	**abstract class** keyword
Cannot be instantiated	cannot be instantiated

all methods are by default public and abstract (no concrete methods are allowed)	we can have abstract methods plus concrete methods in an abstract class
can have only public static final properties	We can declare any type of properties --- final , non final , static non static , primitive non primitive
Used for runtime polymorphism	Used for run time polymorphism
To create subclass we use **implements** keyword	To create sub class we use **extends** keyword
Sub class of interface has two options 1. Add unimplemented methods --- and implement them OR 2. become abstract	Sub class of abstract class has two options 1. Add unimplemented methods --- and implement them OR 2. become abstract
One subclass can have many super interfaces , MULTIPLE inheritance is allowed	one subclass can extends only one super class , MULTIPLE inheritance not allowed

abstract method	Concrete method

public void show( );	public void show()
	{
	sysout
	}

Write an interface ---------
Ex2 ------------- study.interfaces. Sellable
String getProductInfo()
double getProductSellingPrice()

Create a Hierarchy of Sellable
Class Toy is a Sellable
Name , cost, discount

Class Laptop is a sellable
Cpu , ramsize , harddisksize , cost , gst

HW -- add IceCream is a Sellable , add Clothes isa Sellable
Decide properties on your own
Add constructors , getters and settes , override toString
Plus override Sellable methods

Also implement Packable in IceCream and Clothes
In the Sellable Array of Shop ---- add few elements of icecream and clothes also

Ex3 --- write one more interface study.interfaces. Packable
String getPackingDetails()

The Toy and Laptop should implement Packable also !!!!!

Interface is used for Feature wise grouping
Abstract class hierarchy is used for structural grouping!!!

interface is like an agreement !!!

User Component -----------------> agreement<----------------------- Hierarchy Component

User will call as per the agreement
Hierarchy classes will implement as per the agreement !!

HW --- TRYOUTS for the following

1. Interface T1 methods f1 extends interface T2 methods = f2 , class C1 implements T1 } observe how many methods C1 must implement
2. Interface T3(methods=f3) extends T2 and T4(methods = f4) , class C2 implements T3 } observe how many methods C2 must implement
3. Interface T4 (methods = f4 and default method f5) , class C3 implements T4 } observe how many methods C3 must implement
4. Interface T5 (default method f5) class C4 implements both T4 and T5 } observe how many methods C4 must implement

1. using default keyword in interfaces .

Types of interfaces

1. Tagging Interface = the interface without any methods
2. Functional interface = interface with a single method ONLY
3. Normal interface = interface with a few methods

1. One class can implement TWO interfaces having same method
2. One interface can extends from one or more interfaces , all methods of super interfaces have to be implemented by the implementing class

Default method of interface is not a concrete method , it is a convenience feature….if in problem the method is treated as abstract!!!!
CONVENIENCE --- temporary feature given so that sub class can skip overriding the method of interface

---

Functional Interface = interface having EXACTLY one method