DAY 48

------

DOES JAVA ALLOW MULTIPLE INHERITANCE?

--> mUltiple inheritance through classes can not be achived in java because of ambiguity probelm(diamond shape problem).

But multiple inheritance can be achived through interface as shown below:

refer dia:1

refer dia:2

EXAMPLE:

--------

interface A

{

int i =10;

}

interface B

{

int i = 20;

}

interface C extends A,B

{

}

class Test

{

public void disp()

{

System.out.println(A.i);

System.out.println(B.i);

}

}

class Demo

{

public static void main(String[] args)

{

Test t = new Test();

t.disp();

}

}

OUTPUT:

------

10

20

RULE 15:

---------

A class implements a interface, the interface extends another interface then the implementing class should provide the body

for all the abstract methods in the hierarchy

refer dia:3

EXAMPLE:

--------

interface calculate1

{

void add();

}

interface calculate2 extends calculate1

{

void sub();

}

class calculator implements calculate2

{

public void add()

{

int a,b,c;

a=10;

b=20;

c=a+b;

System.out.println(c);

}

public void sub()

{

int a,b,c;

a=10;

b=20;

c=b-a;

System.out.println(c);

}

}

class Demo1

{

public static void main(String[] args)

{

calculator c = new calculator();

c.add();

c.sub();

}

}

OUTPUT:

-------

30

10

RULE 16:

--------

An interface can not implement another interface.

refer dia:4

EXAMPLE:

--------

interface A

{

void add();

}

interface B implements A

{

void sub();

}

OUTPUT:

-------

compilation error

RULE 17:

--------

A class can extends another class simultaneoulsy it can implement the interface also in this case the implementing class should extends first and it

should implements

refer dia: 5

EXAMPLE:

--------

class calc

{

public void add()

{

System.out.println("inside add method");

}

}

interface calculate

{

void sub();

}

class calculator extends calc implements calculate

{

public void sub()

{

System.out.println("inside sub method");

}

}

class Demo2

{

public static void main(String[] args)

{

calculator c = new calculator();

c.add();

c.sub();

}

}

OUTPUT:

-------

inside add method

inside sub method

ADAPTER CLASS:

--------------

Adapter class is a normal class in java which as empty body for all the abstract methods of an interface.

refer dia:6

EXAPLE:

---------

interface compute

{

void fun1();

void fun2();

void fun3();

void fun4();

void fun5();

void fun6();

}

class Adapter implements compute

{

public void fun1()

{

}

public void fun2()

{

}

public void fun3()

{

}

public void fun4()

{

}

public void fun5()

{

}

public void fun6()

{

}

}

class Test1 extends Adapter

{

public void fun1()

{

System.out.println("inside fun1 method");

}

}

class Test2 extends Adapter

{

public void fun4()

{

System.out.println("inside fun4 method");

}

}

class Demo3

{

public static void main(String[] args)

{

Test1 t1 = new Test1();

t1.fun1();

Test2 t4 = new Test2();

t4.fun4();

}

}

OUTPUT:

--------

inside fun1 method

inside fun4 method

Instanceof operator:

--------------------

It is a boolean operator,it can be used to check whether the specified reference variable is

representing the specified class object or not that is compatible or not.

Syntax:

-------

ref\_Var instanceof Class\_Name

NOTE: where ref\_Var and Class\_Name must be related otherwise compiler will rise an error like

"incompatible types error".

If ref\_Var class is same as the specified Class\_Name then instanceof operator will return "true".

If ref\_Var class is subclass to the specified Class\_Name then instanceof operator will return "true".

If ref\_Var class is super class to the specified Class\_Name then instanceof operator will return

"false".

EXAPLE:

---------

class A

{

}

class B extends A

{

}

class C

{

}

class Demo4

{

public static void main(String[] args)

{

A a = new A();

B b = new B();

C c = new C();

System.out.println(a instanceof A);// true

System.out.println(a instanceof B);// false

System.out.println(b instanceof A);// true

System.out.println(c instanceof C);// true

}

}

OUTPUT:

--------

true

false

true

true

MARKER INTERFACE:

-----------------

Marker interface is a empty interface which doesnot contain any abstract methods or fields.

If a class implements a marker interface then it aquaries a special behavoir.

example: if a class implements clonable interface then a duplicate object with the same behaviour can be created.

examples for marker inteface are:

1. Cloneable

2. Serialization

3. Runnable

OBJECT CLONING:

---------------

-->The process of creating duplicate object for an existed object is called as Object Cloning.

-->If we want to perform Object Cloning in Java application then we have to use the following steps.

1.Declare an User defined Class.

2.Implement java.lang.Cloneable interface inorder to make eligible any object for cloning.

3.Override Object class clone() method in user defined class.

public Object clone() throws CloneNotSupportedException

4.In Main class,in main() method,access clone() method over the respective object

EXAMPLE:

--------

class Demo5 implements Cloneable

{

int a = 10;

public static void main(String[] args) throws Exception

{

Demo5 d = new Demo5();

System.out.println(d.a);

Demo5 d1=(Demo5)d.clone();

System.out.println(d1.a);

}

}

OUTPUT:

--------

10

10