DAY-44

------

ABSTRACTION

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

Abstraction is one of the importent obeject oriented features in java.

It is process of hiding the internal implementation and sharing only the related functionality to the user.

ABSTRACTION: It is all about hiding the implementaion(body) of a overridden method.

ENCAPSULATION : It is all about hiding the data members(variable and methods).

In java abstraction can be achived in two ways:

1. Abstarct class [0-100%]

2. Interface

EXAMPLE: PROGRAM WITHOUT ABSTRACTION

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

class Vehicle

{

void Move()

{

System.out.println("vehicle is moving");

}

void Start()

{

System.out.println("vehicle is started");

}

}

class Car extends Vehicle

{

void Start()

{

System.out.println("self start or auto start");

}

}

class Bike extends Vehicle

{

void Start()

{

System.out.println("kick start");

}

}

class Demo

{

public static void main(String[] args)

{

Vehicle V = new Vehicle();

Car C = new Car();

Bike B = new Bike();

V.Move();

V.Start();

C.Move();

C.Start();

B.Move();

B.Start();

}

}

EXAMPLE: PROGRAM WITH ABSTARCTION

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

abstract class Vehicle

{

void Move()

{

System.out.println("vehicle is moving"); // concrete method

}

abstract void Start(); // abstract method

}

class Car extends Vehicle

{

void Start()

{

System.out.println("self start or auto start");

}

}

class Bike extends Vehicle

{

void Start()

{

System.out.println("kick start");

}

}

class Demo

{

public static void main(String[] args)

{

//Vehicle V = new Vehicle();

Car C = new Car();

Bike B = new Bike();

C.Move();

C.Start();

B.Move();

B.Start();

}

}

NOTE:1

-------

1. ABSTRACT METHOD : Abstarct methods are overridden methods where it will be not have implemenation.

2. abstract is a non access modifier keyword which used as prefix for abstarct methods and classes.

3. If any method is abstract in a class then that class is also considered as abstratc class and prefixed with abstract keyword.

4. For abstract classes we can not create the object(not required to create).

5. all the abstract methods should end with the semicolon

6. the inherited method from the parent class is also called as 'concrete method'.

EXAMPLE:

--------

abstract class Bank

{

abstract void roi(); // 100% abstraction

}

class Sbi extends Bank

{

void roi()

{

System.out.println("intrest rate is 7.5%");

}

}

class Hdfc extends Bank

{

void roi()

{

System.out.println("intrest rate is 8.5%");

}

}

class Icici extends Bank

{

void roi()

{

System.out.println("intrest rate is 9.5%");

}

}

class Demo1

{

public static void main(String[] args)

{

//Bank b =new Bnak();

Bank b;

b = new Sbi(); // upcasting

b.roi();

b = new Hdfc(); // upcasting

b.roi();

b = new Icici(); // upcasting

b.roi();

}

}

NOTE:2

-------

1. For abstrat class we can not create object but we can have refence

2. by parent reference we can perform upcasting and polymorphism can be achived

3. abstract class which consist of only abstract methods is considered as 100% abstraction.

DIFFERENCE B/W NORMAL CLASS AND ABSTRACT CLASS.

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

EXAMPLE FOR NORMAL CLASS WITHOUT OBJECT OREIENTATION FEATURES:

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

class Calc

{

int a;

int b;

void add() // concrete method

{

a=10;

b=20;

int c=a+b;

System.out.println(c);

}

void sub() // concrete method

{

a=20;

b=10;

int c=a-b;

System.out.println(c);

}

}

class Demo2

{

public static void main(String[] args)

{

Calc c = new Calc();

c.add();

c.sub();

}

}

EXAMPLE FOR ABSTRACTION CLASS:

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

// normal class

abstract class Calc

{

int a;

int b;

abstract void add(); // abstract method

abstract void sub(); // abstract method

}

class Demo2

{

public static void main(String[] args)

{

//Calc c = new Calc();

//c.add();

//c.sub();

}

}

ABSTRACTION USING THE ABSTRACT CLASS

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

abstract class remote

{

abstract void On();

abstract void Off();

}

class Tv extends remote

{

void On()

{

System.out.println("the TV is on");

}

void Off()

{

System.out.println("the TV is off");

}

}

class Demo3

{

public static void main(String[] args)

{

remote r;

r = new Tv();

r.On();

r.Off();

}

}

RULES W.R.T ABSTRACTION:

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

RULE:1

-------

If a method is made as abstract then compulsory the child class should override the method.

If the child class doesnot override the abstract method then error is displaced during compilation

EXAMPLE:

--------

CASE:1

-------

abstract class Sample

{

abstract void fun1();

}

class Demo4 extends Sample

{

void fun1()

{

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

}

public static void main(String[] args)

{

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

}

}

CASE:2

-------

abstract class Sample

{

abstract void fun1();

}

class Demo4 extends Sample

{

public static void main(String[] args)

{

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

}

}

NOTE: Abstract keyword can be used with following members:

1. class

2. methods

3. Interface

4. Inner class

NOTE: Illegal combination of using abstraction

1. static

2. final

3. private

legal combinations are:

1. Strictfp

2. Synchronized

3. native