DAY-45

------

RULE-2:

If a child class is not in a position to override the abstract method then in the child class the method can be re -declared as abstract and

corresponding child class should also be made as abstract.

EXAMPLE:

---------

abstract class Bird

{

abstract void fly();

abstract void eat();

}

class Pigeon extends Bird

{

void fly()

{

System.out.println("pigeon fly very low");

}

void eat()

{

System.out.println("pigeon eats grains");

}

}

abstract class Eagle extends Bird

{

void fly()

{

System.out.println("eagle fly very high");

}

abstract void eat();

}

class MountainEagle extends Eagle

{

void eat()

{

System.out.println("Mountain eagles hunt and eat over mounatins");

}

}

class GoldenEagle extends Eagle

{

void eat()

{

System.out.println("Golden eagles hunt and eat over oceans");

}

}

class sky

{

void allow(Bird b)

{

b.fly();

b.eat();

}

}

class Rule2

{

public static void main(String[] args)

{

MountainEagle m = new MountainEagle();

GoldenEagle g = new GoldenEagle();

Pigeon p = new Pigeon();

sky s = new sky();

s.allow(m);

s.allow(g);

s.allow(p);

}

}

OUTPUT:

--------

eagle fly very high

Mountain eagles hunt and eat over mounatins

eagle fly very high

Golden eagles hunt and eat over oceans

pigeon fly very low

pigeon eats grains

RULE 3:

-------

An abstract class can have constructor and this gets called during the object creation of its child class.

EXAMPLE:

---------

abstract class Demo

{

public Demo()

{

System.out.println("parent constructor");

}

abstract void fun1();

}

class Demo1 extends Demo

{

/\* public Demo1()

{

super();

}

\*/

void fun1()

{

System.out.println("inside the child class fun1 method");

}

}

class Sample

{

public static void main(String[] args)

{

Demo1 d =new Demo1();

d.fun1();

}

}

OUTPUT:

-------

parent constructor

inside the child class fun1 method

RULE:4

-------

An abstract class can also have the final methods

EAMPLE:

--------

abstract class Demo

{

final void fun1()

{

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

}

abstract void fun2();

}

class Demo1 extends Demo

{

void fun2()

{

System.out.println("inside child fun2");

}

}

class Final

{

public static void main(String[] args)

{

Demo1 d = new Demo1();

d.fun2();

}

}

OUTPUT:

--------

inside child fun2

WHY CAN'T STATIC METHODS BE ABSTRACT?

--> static method will not participate in inheritance but abstratct method should be override by the child class hence abstract and static

together we can't use.

EXAMPLE:

--------

abstract class Demo

{

abstract static void fun1();

}

class Demo1 extends Demo

{

void fun1()

{

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

}

}

class Static

{

public static void main(String[] args)

{

Demo1 d = new Demo1();

d.fun1();

//System.out.println("Hello World!");

}

}

OUTPUT:

-------

compilation error --> ILLEGAL COMBINATION

NOTE:

-----

Can the abstract method in the abstract class be final?

--> NO

EXAMPLE:

--------

abstract class Demo

{

abstract final void fun1();

}

OUTPUT:

-------

ILLEGAL COMBINATION

NOTE:

-----

Can the abstract class be final?

--> NO

EXAMPLE:

--------

abstract final class Demo

{

abstract void fun1();

}

OUTPUT:

-------

ILLEGAL COMBINATION

EXAMPLE:

--------

// program without object oriented features

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

import java.util.\*;

class Square

{

float length;

float area;

public void input()

{

Scanner s = new Scanner(System.in);

System.out.println("enter the length: ");

length = s.nextFloat();

}

public void compute()

{

area= length \* length;

}

public void disp()

{

System.out.println("the area of the square is :"+area);

}

}

class Rectangle

{

float length;

float breadth;

float area;

public void input()

{

Scanner s = new Scanner(System.in);

System.out.println("enter the length: ");

length = s.nextFloat();

System.out.println("enter the breadth: ");

breadth = s.nextFloat();

}

public void compute()

{

area= length \* breadth;

}

public void disp()

{

System.out.println("the area of the rectangle is :"+area);

}

}

class Math

{

public static void main(String[] args)

{

Square s = new Square();

s.input();

s.compute();

s.disp();

Rectangle r = new Rectangle();

r.input();

r.compute();

r.disp();

}

}

OUTPUT:

-------

enter the length:

32.5

the area of the square is :1056.25

enter the length:

12.0

enter the breadth:

10.0

the area of the rectangle is :120.0

EXAMPLE:

---------

// program with object oriented features

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

import java.util.\*;

abstract class Shape

{

float area;

float length;

abstract public void input();

abstract public void compute();

abstract public void disp();

}

class Square extends Shape

{

public void input()

{

Scanner s = new Scanner(System.in);

System.out.println("enter the length: ");

length = s.nextFloat();

}

public void compute()

{

area= length \* length;

}

public void disp()

{

System.out.println("the area of the square is :"+area);

}

}

class Rectangle extends Shape

{

float breadth;

public void input()

{

Scanner s = new Scanner(System.in);

System.out.println("enter the length: ");

length = s.nextFloat();

System.out.println("enter the breadth: ");

breadth = s.nextFloat();

}

public void compute()

{

area= length \* breadth;

}

public void disp()

{

System.out.println("the area of the rectangle is :"+area);

}

}

class solve

{

void allow(Shape ref)

{

ref.input();

ref.compute();

ref.disp();

}

}

class Maths

{

public static void main(String[] args)

{

Square s = new Square();

Rectangle r = new Rectangle();

solve s1 = new solve();

s1.allow(s);

s1.allow(r);

}

}

OUTPUT:

-------

enter the length:

12.5

the area of the square is :156.25

enter the length:

15.4

enter the breadth:

16.6

the area of the rectangle is :255.64