DAY-32

------

EXAMPLE:1

----------

class Student

{

private String name;

private int age;

Student(String sname,int sage)

{

name=sname;

age=sage;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class StudentApp

{

public static void main(String[] args)

{

Student s1= new Student("ramu",21);

System.out.println(s1.getName());

System.out.println(s1.getAge());

}

}

OUTPUT:

--------

ramu

21

EXAMPLE:2

---------

// SHADOWING PROBLEM

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

class Student

{

private String name;

private int age;

Student(String name,int age)

{

name=name;

age=age;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class StudentApp

{

public static void main(String[] args)

{

Student s1= new Student("ramu",21);

System.out.println(s1.getName());

System.out.println(s1.getAge());

}

}

OUTPUT:

--------

null

0

Whenever local variable name is same as the instance variable then the priority is given to local variable. Because of this problem object initilization

will not happen in the proper way.

This problem is called as Shadowing problem.

Solution to the above problem is :

1. use the 'this' keyword to differeniate b/w instance and local variable.

EXAMPLE:

--------

// solution to the SHADOWING PROBLEM by using 'this' keyword.

class Student

{

private String name;

private int age;

Student(String name,int age)

{

this.name=name;

this.age=age;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class StudentApp

{

public static void main(String[] args)

{

Student s1= new Student("ramu",21);

System.out.println(s1.getName());

System.out.println(s1.getAge());

Student s2= new Student("somu",22);

System.out.println(s2.getName());

System.out.println(s2.getAge());

}

}

OUTPUT:

-------

ramu

21

somu

22

NOTE: 'this' keyword is used to solve the shadowing probelm.

this keyword always points the currently executing object and also brings the difference b/w local and instance variable.

TYPES OF CONSTRUCTOR

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

The types of constructors are:

1. default constructor

2. user defined default constructor

3. parameterized constructor

DEFAULT CONSTRUCTOR:

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

refer fig:1

-----------

In a class if constructor is not present during the compilation compiler will add default constructor.

The default constructor is no argument or zero parameterized in nature.

The only one line of code present within the default constructor is super() method.

refer fig:2

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

Whenever compiler adds the default constructor the access modifier of the constructor will be same as that of class.

refer fig:3

-----------

In the above example void Demo() is not a constructor it is a method.

refer fig:4

-----------

If user has provided the constructor within a class then during compilation compiler will add a super() method to the body of the Constructor.

USER DIFIEND DEFAULT CONSTRUCTOR:

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

In a class if a programmer adds a constructor then compiler will not add the default constructor.

UDC is always added by the user with no args/zero parameter.

In a class there would be either UDC or DC. Both of them will never occur simultaneoulsy in the class.

EXAMPLE:

--------

class Student

{

String name;

int age;

Student()

{

name="ramu"; // USER DIFIEND CONSTRUCTOR

age=21;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class Demo

{

public static void main(String args[])

{

Student s1 = new Student();

System.out.println(s1.getName());

System.out.println(s1.getAge());

}

}

OUTPUT:

-------

ramu

21

THE DIFFERENCE B/W CONSTRUCTOR AND METHOD

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

EXAMPLE:

--------

class A

{

A()

{

System.out.println("Inside the constructor");

}

void A()

{

System.out.println("Inside the method");

}

}

class Demo1

{

public static void main(String[] args)

{

A a1= new A();

a1.A();

a1.A();

}

}

OUTPUT:

-------

Inside the constructor

Inside the method

Inside the method

CONSTRUCTOR METHOD

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

1. Constructor should be same as class name 1. method name can be anything.

2. They dont have return type 2. return type is compulsory

3. Constructor calling is implicit 3. method calling is explicit

4. Constructor will be called only one during the object creation 4. method can be called n number times

5. access modifier permitted are : public,private,protected,default 5. we can also use static,final ...etc

6. Constructors can not be inherited 6. methods can be inherited

PARAMETERIZED CONSTRUCTOR:

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

Constructors with the parameter is called as parameterized constructor. It is a user difiend constructor.

PC is used to initilize the object by the user input.

EXAMPLE:

--------

class Student

{

String name;

int age;

Student(String name)

{

this.name=name; // PARAMETERIZED CONSTRUCTOR

}

public String getName()

{

return name;

}

}

class Demo

{

public static void main(String args[])

{

Student s1 = new Student("RAMU");

System.out.println(s1.getName());

}

}

OUTPUT:

--------

RAMU

EXAMPLE:2

---------

class Student

{

String name;

int age;

Student(String name)

{

this.name=name; // PARAMETERIZED CONSTRUCTOR

}

public String getName()

{

return name;

}

}

class Demo

{

public static void main(String args[])

{

Student s1 = new Student();

System.out.println(s1.getName());

}

}

OUTPUT:

-------

COMPILATION ERROR

NOTE: In the above program we are trying to call the paramterized constructor with out passing the argument hence we will get compilation error.