Que. Explain the concept of constructor and different types of constructor with suitable example?

Answer:

Constructor: Constructor is special method of class that is automatically invoked whenever the object is created. It is used to initialize the member variables of object.

Rules to create constructor

1. Constructor name should exactly match with class name.
2. Access specifier of constructor could be private/public or protected.
3. Constructor should not have any return type not even void.
4. Constructor should not contain any return statement.
5. Overloading of constructor is allowed.
6. Implicit return value of constructor is always an object.

Types of constructor

There are only two types of constructor available in java language as given below.

1. Default Constructor: Constructor with no parameter or zero parameters is known as default constructor.
2. Parameterized Constructor: Constructor with one or more than one parameters is known as parameterized constructor.

Note: copy constructor is not available in java language, but we can perform operation of copy constructor using the parameterized constructor.

Example:

```
class Data
{
        private int a;
        public Data()
        {
        a=10;
        }

         public Data(int n)
```

```
    {
    a=n;
    }
    public void show()
    {
    System.out.println("Value="+a);
    }
}
class UseMe
{
    public static void main(String args[])
    {
        Data d1=new Data();
        Data d2=new Data(100);
        d1.show();
        d2.show();
    }
}
```

Output:

Value=10

Value=100

In the above example, Data class contain two types of constructor i.e. default and parameterized constructor. Default constructor is initializing the value of member variable 'a' by 10 where as parameterized constructor is initializing the value of member variable by the value of given parameter.

During the execution of main program, object d1 will create memory and initialize the value of member variable by executing the default constructor where as object d2 will create memory and initialize the value of member variable by using parameterized constructor.

Note: if constructors are defined in the class then we can access only the defined constructors but if the class is not containing any constructor then we can access default constructor.

Que. Explain the concept of garbage collection?

or

Que.Explain the finalize method of Java language?

Answer:

Garbage Collection: Automatic de-allocation of memory of object whenever execution control comes out of the block in which the object is declared is known as garbage collection.

Whenever any object is declared or created within the block then scope of that object will be local to the block and life of object will end after the complete execution of block. Java does not provide destructor or any type of delete operator to delete or free the allocated memory but the de-allocation operation is automatically performed by java interpreter whenever the execution control comes out of the block in which the object is declared. To perform the de-allocation operation, java interpreter always invokes finalize() method whenever the scope of object ends. finalize() method is native of Object class that is always available in all the classes. To perform resource free operation in user defined class, java developer can override the finalize() method and clean or free the resources of class.

Java can also convert the unused blocks of memory into free space by using the static method gc() of System class. After the garbage collection operation using gc() method, free memory space of program will automatically increased.

Que.Explain this keyword of java language with suitable example?

Answer:

'this' is standard keyword of java language that represents the reference of currently invoking object and it is used to access member variable, methods and constructors of invoking object. 'this' keyword is accessible only in non-static method and constructors where as it is not allowed in static block or static method.

Note: In java language, this is not a pointer.

this variable

Example:

```
class Data
{
        private int a;
        public Data(int a)
        {
        a=a;
        }
        public void show()
        {
        System.out.println("Value="+a);
        }
}
class UseMe
{
        public static void main(String args[])
        {
        Data d1=new Data(10);
        d1.show();
        }
}
```

Output:

Value=0

In the above example, Data class contain parameterized constructor initializing the value of member variable 'a' but both the source and target variable will represent the local variables only. During the execution of parameterized constructor of Data class, execution control will access the value of variable 'a' from the local variable and not from the class level variable i.e. constructor will not initialize the member variable of object d1.

To solve the above problem, we have to use the 'this' keyword as show below.

Example:

```
class Data
{
        private int a;
        public Data(int a)
        {
        this.a=a;
        }
        public void show()
        {
        System.out.println("Value="+a);
        }
}
class UseMe
{
        public static void main(String args[])
        {
        Data d1=new Data(10);
        d1.show();
        }
}
```

Output:

Value=10

In the above example, this keyword represents the reference of invoking object and accessing the member variable of object.

this method

Example:

```
class Data
{
```

```
    private int a;
    public Data(int a)
    {
    this.a=a;
    this.show();
    }
    public void show()
    {
    System.out.println("Value="+a);
    }
}
class UseMe
{
        public static void main(String args[])
        {
        Data d1=new Data(10);
        d1.show();
        }
  }
```

Output:

```
        Value=10
        Value=10
```

this constructor

Example:

```
class Data
{
        private int a;
        public Data()
        {
        this(10);
        }
```

```
    public Data(int a)
    {
    this.a=a;
    }
     public void show()
     {
      System.out.println("Value="+a);
      }
}
class UseMe
{
        public static void main(String args[])
        {
         Data d1=new Data();
         d1.show();
         }
  }
```

Output:

Value=10