**Java OOPs Concepts**

Object means a real word entity such as pen, chair, table etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

Object

Class

Inheritance

Polymorphism

Abstraction

Encapsulation

**Object in Java**

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tangible and intangible). The example of intangible object is banking system.

An object has three characteristics:

**state**: represents data (value) of an object.

**behavior**: represents the behavior (functionality) of an object such as deposit, withdraw etc.

**identity**: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

**What is Class:**

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

A class in Java can contain:

fields

methods

constructors

blocks

Syntax to declare a class:

class <class\_name>{

field;

method;

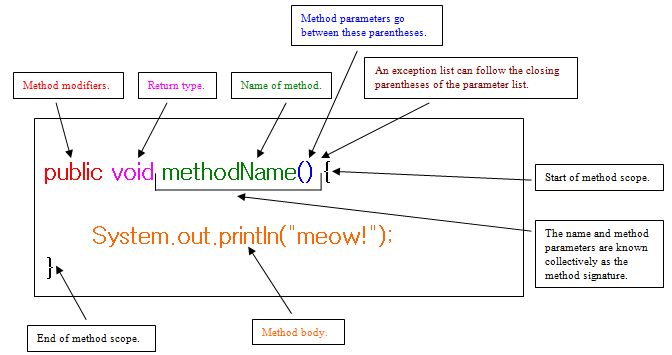
}

====================================================================

**Methods in Java**

A method is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name.

Each method has its own name.  When that name is encountered in a program, the execution of the program branches to the body of that method.  When the method is finished, execution returns to the area of the program code from which it was called, and the program continues on to the next line of code.



**new keyword in Java**

The new keyword is used to allocate memory at run time. All objects get memory in Heap memory area.

===================================================================

**Object and Class Example: main within class**

class Student{

int id; //field or data member or instance variable

String name;

public static void main(String args[]){

Student s1=new Student(); //creating an object of Student

System.out.println(s1.id); //accessing member through reference variable

System.out.println(s1.name);

}

}

====================================================================

**Object and Class Example: main outside class(TestStudent1.java)**

class Student{

int id;

String name;

}

class TestStudent1{

public static void main(String args[]){

Student s1=new Student();

System.out.println(s1.id);

System.out.println(s1.name);

}

}

====================================================================

**There are 3 ways to initialize object in java.**

**1)By reference variable**

**2)By method**

**3)By constructor**

**1) Object and Class Example: Initialization through reference variable**

Initializing object simply means storing data into object. Let's see a simple example where we are going to initialize object through reference variable.

class Student{

int id;

String name;

}

class TestStudent2{

public static void main(String args[]){

Student s1=new Student();

s1.id=101;

s1.name="Sonoo";

System.out.println(s1.id+" "+s1.name);//printing members with a white space

}

}

==================================================================

**2) Object and Class Example: Initialization through method**

class Student{

int rollno;

String name;

void insertRecord(int r, String n){

rollno=r;

name=n;

}

void displayInformation()

{

System.out.println(rollno+" "+name);}

}

class TestStudent4{

public static void main(String args[]){

Student s1=new Student();

Student s2=new Student();

s1.insertRecord(111,"Alok");

s2.insertRecord(222,"Prashant");

s1.displayInformation();

s2.displayInformation();

}

}

===================================================================

**Object and Class Example: Rectangle**

class Rectangle{

int length;

int width;

void insert(int l, int w)

{

length=l;

width=w;

}

void calculateArea()

{

System.out.println(length\*width);

}

}

class TestRectangle1{

public static void main(String args[]){

Rectangle r1=new Rectangle();

Rectangle r2=new Rectangle();

r1.insert(11,5);

r2.insert(3,15);

r1.calculateArea();

r2.calculateArea();

}

}

=====================================================================================

**Constructor in java**

Constructor in java is a special type of method that is used to initialize the object.Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.

**Rules for creating java constructor**

There are basically two rules defined for the constructor.

**1)**Constructor name must be same as its class name

**2)**Constructor must have no explicit return type

There are two types of constructors:

1)Default constructor (no-arg constructor)

2)Parameterized constructor

**Example of default constructor:**

class Player

{

Player()

{

System.out.println("Player is created");

}

public static void main(String args[])

{

Player p =new Player();

}

}

=====================================================================================

**Note:** If there is no constructor in a class, compiler automatically creates a default constructor. Default constructor provides the default values to the object like 0, null etc. depending on the type.

================================================================================================

**Example of default constructor that displays the default values:**

class Student{

int id;

String name;

void display()

{

System.out.println(id+" "+name);

}

public static void main(String args[]){

Student s1=new Student();

Student s2=new Student();

s1.display();

s2.display();

}

}

================================================================================================

**Example of parameterized constructor**

Parameterized constructor is used to provide different values to the distinct objects.

class Student4{

int id;

String name;

Student4(int i,String n){

id = i;

name = n;

}

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){

Student4 s1 = new Student4(111,"Karan");

Student4 s2 = new Student4(222,"Aryan");

s1.display();

s2.display();

}

}

=================================================================================================

**Constructor Overloading in Java**

Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

class Student5{

int id;

String name;

int age;

Student5(int i,String n){

id = i;

name = n;

}

Student5(int i,String n,int a){

id = i;

name = n;

age=a;

}

void display(){System.out.println(id+" "+name+" "+age);}

public static void main(String args[]){

Student5 s1 = new Student5(111,"Karan");

Student5 s2 = new Student5(222,"Aryan",25);

s1.display();

s2.display();

}

}

=================================================================================================

**Difference between constructor and method in java**

|  |  |
| --- | --- |
| **Java Constructor** | **Java Method** |
| Constructor is used to initialize the state of an object. | Method is used to expose behaviour of an object. |
| Constructor must not have return type. | Method must have return type. |
| Constructor is invoked implicitly. | Method is invoked explicitly. |
| The java compiler provides a default constructor if you don't have any constructor. | Method is not provided by compiler in any case. |
| Constructor name must be same as the class name. | Method name may or may not be same as class name. |

**Java Copy Constructor**

There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.

class Student6{

int id;

String name;

Student6(int i,String n){

id = i;

name = n;

}

Student6(Student6 s){

id = s.id;

name =s.name;

}

void display()

{

System.out.println(id+" "+name);

}

public static void main(String args[]){

Student6 s1 = new Student6(111,"Karan");

Student6 s2 = new Student6(s1);

s1.display();

s2.display();

}

}

====================================================================================