**Constructor-**

Its name is same as class name called as constructor.

It is invoked by JVM automatically when you create the object of class.

It does not return anything even void also.

There are two types of constructor are as

* Default constructor(No- argument constructor)
* Parameterized constructor

**Default constructor (No- argument constructor)**

A constructor that does not accept any arguments called as default constructor.

**Program for no argument constructor**

**package** com.test;

**public** **class** Test2 {

String name;

**public** Test2() {

name = "velocity";

}

**public** **static** **void** main(String[] args) {

Test2 test2 = **new** Test2();

System.***out***.println("Name is>>" + test2.name);

}

}

Output-

Name is>>velocity

**Parameterized constructor-**

A constructor with arguments called as parameterized constructor.

**Program for parameterized constructor**

**package** com.test;

**public** **class** Test2 {

**int** id;

String name;

String city;

**public** Test2(**int** userId, String userName, String userCity) {

id = userId;

name = userName;

city = userCity;

System.***out***.println("id>>" + id);

System.***out***.println("name>>" + name);

System.***out***.println("city>>" + city);

}

**public** **static** **void** main(String[] args) {

Test2 test2 = **new** Test2(10, "ram", "pune");

}

}

Output-

id>>10

name>>ram

city>>pune

Note-

1. When you don’t write any constructor in the class then default constructor will be added by JVM automatically at the compile time.
2. When you write any constructor in the class then default constructor will not added by JVM.
3. Constructor does not have any return type. If you write any return type then it will be called as method.
4. Java constructor cannot be abstract, static, final, and synchronized.

**What is the use of constructor-?**

If you want to execute some code at object creation. In other words, it is used for generally initialization of global variables.

There are five ways to calling the constructor as

Employee e= new Employee();

super();

this();

new Employee();

class.forName(“com.test”).newInstance();

Constructor can be overloaded because we can write same name with different arguments.

**Program for overloading the constructor.**

**package** com.test;

**package** com.test;

**public** **class** Employee {

**int** employeeId;

String employeeName;

String employeeCountry;

**public** Employee(**int** id, String name, String country) {

employeeId = id;

employeeName = name;

employeeCountry = country;

}

**public** Employee(**int** id, String name) {

employeeId= id;

employeeName = name;

}

@Override

**public** String toString() {

**return** "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", employeeCountry="

+ employeeCountry + "]";

}

**public** **static** **void** main(String[] args) {

Employee e = **new** Employee(1, "shyam", "pune");

Employee e1 = **new** Employee(1, "ram");

System.***out***.println(e);

System.***out***.println(e1);

}

}

Note-

1. Constructor cannot be overridden because we can’t write multiple constructors with same arguments. If you are tried to write it then you will get compiler time error “Duplicate method Employee”.
2. Private Constructor is mostly used in singleton design pattern in java.