Department:- BCA
Faculty Name:- Er. Ashish Kumar
Subject:- JAVA(5th sem)
Topic:-Methods in java

Methods in Java

A method is a collection of statements that perform some specific task and return the result to the caller. A method can perform some specific task without returning anything. Methods allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class which is different from languages like C, C++, and Python. Methods are **time savers** and help us to **reuse** the code without retyping the code.

Syntax :-

```
<access_modifier> <return_type> <method_name>(<parameter_list1,....>)
        {
            //body of method
        }
```

EX :-

```
public int max(int n, int m)
 {
     if(n>m)
       return n;
     else
      return m;
 }
```

- **Modifier**-: Defines **access type** of the method i.e. from where it can be accessed in our application. In Java, there 4 type of the access modifiers.
  - **public:** accessible in all class in our application.

- **protected:** accessible within the class in which it is defined and in its **subclass(es)**
- **private:** accessible only within the class in which it is defined.
- **default:** (declared/defined without using any modifier) : accessible within same class and package within which its class is defined.

- **The return type** : The data type of the value returned by the method or void if does not return a value.
- **Method Name** : the rules for field names apply to method names as well, but the convention is a little different.
- **Parameter list** : Comma separated list of the input parameters are defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses ().
- **Exception list** : The exceptions we expect by the method can throw, we can specify these exception(s).
- **Method body** : it is enclosed between braces. The code we need to be executed to perform our intended operations.

Example:-

C:\Users\ashishjha\Desktop\java\TestNum.java - Notepad++

```java
//find the greatest value using return type method.
class Num
{
    int x,y;
    int max(int m,int n)
    {
        x=m;
        y=n;
    if(x>y)
     return x;
     else
     return y;
    }
}
class TestNum
{
    public static void main(String args[])
      {
     int i;
     Num obj =new Num();
     i=obj.max(10,20);
    System.out.println("the greatest value is:"+i);
      }
}
```

Command Prompt

```
C:\Users\ashishjha\Desktop\java>javac TestNum.java

C:\Users\ashishjha\Desktop\java>java TestNum
the greatest value is:20

C:\Users\ashishjha\Desktop\java>
```

Method Overloading in Java

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**. If we have to perform only one operation, having same name of the methods increases the readability of the program.

There are two ways to overload the method in java

- **By changing number of arguments**
- **By changing the data type**

- **By changing number of arguments:-**

Example:-

C:\Users\ashishjha\Desktop\java\TestAddNum.java - Notepad++

```java
class AddNum
{
 int add(int a,int b,int c)
  {
  return(a+b+c);
  }
  int add(int a,int b)
  {
  return(a+b);
  }
}
class TestAddNum
{
 public static void main(String args[])
  {
     int x;
     AddNum obj=new AddNum();
     x=obj.add(10,20,30);
     System.out.println("sum="+x);
     x=obj.add(10,20);
     System.out.println("sum="+x);
  }
}
```

Command Prompt

```
C:\Users\ashishjha\Desktop\java>javac TestAddNum.java

C:\Users\ashishjha\Desktop\java>java TestAddNum
sum=60
sum=30

C:\Users\ashishjha\Desktop\java>
```

- **By changing the data type**

Example:-

C:\Users\ashishjha\Desktop\java\TestAddDNum.java - Notepad++

```java
class AddNum
{
 int add(int a,int b)
  {
  return(a+b);
  }
  double add(double a,double b)
  {
  return(a+b);
  }
}
class TestAddDNum
{
 public static void main(String args[])
  {
     int x;
     double y;
     AddNum obj=new AddNum();
     x=obj.add(10,20);
     System.out.println("sum="+x);
     y=obj.add(10.5,20.7);
     System.out.println("sum="+y);
  }
}
```

Command Prompt

```
sum=60
sum=30

C:\Users\ashishjha\Desktop\java>javac TestAddDNum.java

C:\Users\ashishjha\Desktop\java>java TestAddDNum
sum=30
sum=31.2

C:\Users\ashishjha\Desktop\java>
```

Java source file | length : 388 lines : 25 | Ln : 18 Col : 30 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Note:-In java, method overloading is not possible by changing the return type of the method only because of ambiguity.

Q. Can we overload java main() method?

Ans:- Yes, by method overloading. we can have any number of main methods in a class by method overloading. But **JVM** calls main() method which receives string array as arguments only.

Example:-

C:\Users\ashishjha\Desktop\java\TestMainOverload.java - Notepad++

```java
class TestMainOverload
{


    public static void main(String[] args)
    {
    System.out.println("main with String[]");
    }
    public static void main(String args)

    {
    System.out.println("main with String");
    }
    public static void main()
    {
    System.out.println("main without args");
    }
}
```

Command Prompt

```
sum=31.2

C:\Users\ashishjha\Desktop\java>javac TestMainOverload.java

C:\Users\ashishjha\Desktop\java>java TestMainOverload

main with String[]

C:\Users\ashishjha\Desktop\java>
```