Method:

- ✓ Method is a sub block of a class that contains logic of that class.

- ✓ Logic must be placed inside a method, not directly at class level, if we place logic at class level compiler throws an error.

- ✓ So class level we are allowed to place variables and methods.

- ✓ The logical statements such as method calls, calculations and printing related statements must be placed inside method, because these statements are considered as logic.

```
class sample
{
int a;
int b;

System.out.println("instance of java"); // compiler throws an error.

}
```

```
class sample
{
static int a=10;
public static void main(String args[])
{
System.out.println(a); // works fine,prints a value:10
 }
}
```

Method Terminology:

1. Method Prototype:

✓ The head portion of the method is called method prototype.

```
class sample{

static int b=10;

public static void main(String args[]) // ->method prototype.
{

System.out.println(b);

 }
```

2. Method body and logic:

❖ The "{ }" region is called method body, and the statements placed inside method body is called logic.

```
class sample{

public static void main(String args[]) // ->method prototype.
{

  int a=10,b=20; // method logic
  int c=a+b;  // method logic
  System.out.println(c);  // method logic
```

```
 }
}
```

3. Method parameters and arguments:

- The variables declared in method parenthesis "( )" are called parameters.
- We can define method with 0 to n number of parameters.
- The values passing to those parameters are called arguments.
- In method invocation we must pass arguments according to the parameters order and type.

```
class sample
{
public void add(int a, int b) // ->method prototype. int a, int b are parameters
  {

      int c=a+b;  // method logic
      System.out.println(c);  // method logic

       }

      public static void main(String args[]) // ->method prototype.
```

```
{

 sample obj= new sample();

 obj.add(1,2); //method call , here 1, 2 are arguments.

}
}
```

4: Method signature:

- The combination of method "name + parameters " is called method signature

```
class sample{

 public void add(int a, int b) // ->method prototype. int a, int b are parameters
{

 int c=a+b;  // method logic
 System.out.println(c);  // method logic

 }

public static void main(String args[]) // ->method prototype.
{
```

```
 sample obj= new sample();

 obj.add(1,2); //method call , here 1, 2 are arguments.

}
}
```

- In the above program add(int a, int b) and main(String args[]) are method signatures.

5. Method return type:

- The keyword that is placed before method name is called method return type.
- It tells to compiler and JVM about the type of the value is returned from this method after its execution
- If nothing is return by method then we can use "void" keyword which specifies method returns nothing.

```
class sample{
```

```
    public int add(int a, int b) // ->method prototype. int a, int b are
parameters
    {

      int c=a+b;  // method logic
      return c;
     }

    public static void main(String args[]) // ->method prototype.
    {

      sample obj= new sample();

    int x= obj.add(1,2); //method call , here 1, 2 are arguments.
    System.out.println(x);

    }
    }
```

Type of declaration of methods based on return type and arguments:

1.Method with out return type and without arguments.

```
    class sample{

   static public void add(){

    int a=10;
    int b=10;
    int c=a+b;
```

```
System.out.println(c);

}

public static void main(String args[]) // ->method prototype.
{
     add();
 }
}
```

2.Method with out return type and with arguments.

```
package com.instanceofjava;
class sample{

public void add(int a, int b){

int c=a+b;
System.out.println(c);

}
```

```
public static void main(String args[]) // ->method prototype.
{

sample obj= new sample();
obj.add(1,2);



 }
}
```

3.Method with return type and without arguments.

```
package com.instanceofjava;
class sample{

public int add(){
int a=20;
int b=30;
int c=a+b;
return c;
}

public static void main(String args[]) // ->method prototype.
{

sample obj= new sample();
int x=obj.add();
System.out.println(x);
```

```
 }
}
```

4.Method with return type and with arguments.

```
package com.instanceofjava;
class sample{

public int add(int a, int b){

int c=a+b;
return c;
}

public static void main(String args[]) // ->method prototype.
{

sample obj= new sample();

int x=obj.add(1,2);
System.out.println(x);

 }
}
```

Method terminology : main method

Lets take main method and identify method parts

javamethods

1.Method Creation with body:

The process of creating method with body is called method definition. Technically this method called concrete method.

```
class A{

public void add(){

}
```

}

2.Method Creation without body:

Creating a method without body is called "Declaring a method" or "Method declaration".

Technically this method called "abstract method".

In method declaration the modifier "abstract" is mandatory and also should be terminated with ";".

Example : public void add();

And only abstract class can have abstract method. (in interface we can use , by default all methods are abstract)

```
abstract class A{

abstract  public void add();

}
```

Static and non static methods:

If a method has static keyword in its definition then that method called static method.

We can call static method directly from main method.

If method does not have static keyword in its definition then it is a non static method.

If we want to call a non static method from main method we need object of that class

if we want to call static method outside the class we can call by using classname.method();

```
class A{

public static void show(){    // static method
 System.out.println("static method");
}

public void display(){      //non static method
 System.out.println("non static method");
}

public static void main(String[] args){
show(); // static method call
A obj= new A();
obj.display(); // non static method call
}
}
```