What is a method?

A method is a collection of statements that are grouped together to perform an operation.

```java
public static int methodName(int a, int b) {
    // body
}
```

- method definition
  - `public static` - modifier
  - `int` - return type
  - `methodName` - Name of the method
  - `int a, int b` – list of parameters
  - `// body` - method body

Method Example

Here is the method takes two parameters num1 and num2 and returns the minimum between the two.

```
/** the snippet returns the minimum between two numbers */

public static int minFunction(int n1, int n2) {
   int min;
   if (n1 > n2)
      min = n2;
   else
      min = n1;

   return min;
}
```

Method Calling

A method should be called to make use of it. There are two ways in which a method is called i.e., method returns a value or returning nothing (no return value).

The methods returning void is considered as call to a statement. Lets consider an example –

```
System.out.println("This is tutorialspoint.com!");
```

The method returning value can be understood by the following example –

```
int result = sum(6, 9);
```

Method Calling Example

```java
public class ExampleMinNumber {
   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }
   /** returns the minimum of two numbers */
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;

      return min;
   }
}
```

This will produce the following result – `6`

The void Keyword

The void keyword allows us to create methods which do not return a value.

```java
public class ExampleVoid {

    public static void main(String[] args) {
        methodRankPoints(255.7);
    }
    public static void methodRankPoints(double points) {
        if (points >= 202.5) {
            System.out.println("Rank:A1");
        }else if (points >= 122.4) {
            System.out.println("Rank:A2");
        }else {
            System.out.println("Rank:A3");
        }
    }
}
```

Passing Parameters by Value

While calling arguments is to be passed in the same order as their respective parameters in the method specification.

```
  public class swappingExample {

   public static void main(String[] args) {
      int a = 30; int b = 45;
      swapFunction(a, b); // Invoke the swap method
   }

   public static void swapFunction(int a, int b) {
      System.out.println("Before swapping(Inside), a = " + a +
      // Swap n1 with n2
      int c = a; a = b; b = c;
      System.out.println("After swapping(Inside), a = " + a + "
   }
}
```

Method Overloading

When a class has two or more methods by the same name but different parameters, it is known as method overloading.

Overloading Example

```
public class ExampleOverloading {

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      double c = 7.3;
      double d = 9.4;
      int result1 = minFunction(a, b);

      // same function name with different parameters
      double result2 = minFunction(c, d);
      System.out.println("Minimum Value = " + result1);
      System.out.println("Minimum Value = " + result2);
   }
```

Conti...

```java
    // for integer
    public static int minFunction(int n1, int n2) {
       int min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;

       return min;
    }

    // for double
    public static double minFunction(double n1, double n2) {
      double min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;

       return min;
    }
}
```

Using Command-Line Arguments

Sometimes you will want to pass some information into a program when you run it. This is accomplished by passing command-line arguments to main( ).

Example

```java
public class CommandLine {

   public static void main(String args[]) {
      for(int i = 0; i<args.length; i++) {
         System.out.println("args[" + i + "]: " +  args[i]);
      }
   }
}
```

Try executing this program as shown here –

```
$java CommandLine this is a command line
```

The Constructors

A constructor initializes an object when it is created. It has the same name as its class and is syntactically similar to a method.

Example

```java
// A simple constructor.
class MyClass {
   int x;

   // Following is the constructor
   MyClass() {
      x = 10;
   }
}
```

Cont..

Calling Constructors

You will have to call constructor to initialize objects as follows –

```
public class ConsDemo {

   public static void main(String args[]) {
      MyClass t1 = new MyClass();
      MyClass t2 = new MyClass();
      System.out.println(t1.x + " " + t2.x);
   }
}
```

Output `10 10`

Parameterized Constructor

Most often, you will need a constructor that accepts one or more parameters. Parameters are added to a constructor in the same way that they are added to a method.

Example

Here is a simple example that uses a constructor with a parameter

```
// A simple constructor.
class MyClass {
   int x;

   // Following is the constructor
   MyClass(int i ) {
      x = i;
   }
}
```

Parameterized Parameterized Constructor

You will need to call a constructor to initialize objects as follows

```
public class ConsDemo {

   public static void main(String args[]) {
      MyClass t1 = new MyClass( 10 );
      MyClass t2 = new MyClass( 20 );
      System.out.println(t1.x + " " + t2.x);
   }
}
```

Output `10 20`

Reference list

1. https://docs.oracle.com/javase/tutorial/java/javaOO/index.html
2. http://web.mit.edu/1.00/www/definitions.htm
3. https://www.tutorialspoint.com/java/java_methods.htm