Java Methods

User-defined Method

You can also define methods inside a class as per your wish. Such methods are called user-defined methods.

How to create a user-defined method?

Before you can use (call a method), you need to define it.

Here is how you define methods in Java.

```
public static void myMethod() {
        System.out.println("My Function called");
}
```

Here, a method named `myMethod()` is defined.

You can see three keywords `public`, `static` and `void` before the function name.

- The `public` keyword makes `myMethod()` method `public`. Public members can be accessed from outside of the class.
    - The `static` keyword denotes that the method can be accessed without creating the object of the class. The `void` keyword signifies that the method doesn't return any value.

Standard Library Methods

The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (*.jar) file with JVM and JRE.

For example,

- `print()` is a method of `java.io.PrintSteam`. The `print("...")` prints the string inside quotation marks.
- `sqrt()` is a method of Math class. It returns square root of a number.

Here's an working example:

```
public class Numbers {
   public static void main(String[] args) {
      System.out.print("Square root of 4 is: " + Math.sqrt(4));
   }
}
```

When you run the program, the output will be:

```
Square root of 4 is: 2.0
```

Program 01

```
public class Method01 {
public static void method(String name, int age){
System.out.println("The formal parameters have value: " + name + " and " + age);
name = "XXX";
age = -1;
System.out.println("The formal parameters have value: " + name + " and " + age);
}

public static void main(String[] args){
// Call the method passing in literal values
method("Sarah", 20);
// Call the method passing in variables
String name = "Anne";
int age = 19;
System.out.println("The actual inputs have value: " + name + " and " + age);
method(name, age);
System.out.println("The actual inputs have value: " + name + " and " + age);
}
}
```

Program 02

Example: Complete Program of Java Method

```
class Method_02 {

  public static void main(String[] args) {
    System.out.println("About to encounter a method.");

    // method call
    myMethod();

    System.out.println("Method was executed successfully!");
  }

  // method definition
  private static void myMethod(){
    System.out.println("Printing from inside myMethod()!");
  }
}
```

Program 03

```
class Method_03 {

  public static void main(String[] args) {

    Output obj = new Output();
    System.out.println("About to encounter a method.");

    // calling myMethod() of Output class
    obj.myMethod();

    System.out.println("Method was executed successfully!");
  }
}

class Output {

  // public: this method can be called from outside the class
  public void myMethod() {
    System.out.println("Printing from inside myMethod().");
  }
}
```

Program 04

Example: Method Accepting Arguments and Returning Value

```
public class SquareMain_me {

   public static void main(String[] args) {
      int result, n;

      n = 3
      result = square(n);
      System.out.println("Square of 3 is: " + result);

      n = 4
      result = square(n);
      System.out.println("Square of 4 is: " + result);
   }

            static int square(int i) {
      return i * i;
   }
}
```