Lecture – 3 (Java Input/Output)

Java Output

You can simply use System.out.println(), System.out.print() or System.out.printf() to send output to standard output (screen).

System is a class and out is a public static field which accepts output data. Don't worry if you don't understand it. Classes, public, and static will be discussed in later chapters.

Let's take an example to output a line.

```
public class AssignmentOperator {
   public static void main(String[] args) {

            System.out.println("Java programming is interesting.");
   }
}
```

When you run the program, the output will be:

Java programming is interesting.

Here, println is a method that displays the string inside quotes.

What's the difference between println(), print() and printf()?

- print() - prints string inside the quotes.
- println() - prints string inside the quotes similar like print() method. Then the cursor moves to the beginning of the next line.
- printf() - it provides string formatting (similar to printf in C programming).

Example 2: print() and println()

```
public class Output {
   public static void main(String[] args) {

            System.out.println("1. println ");
```

```
            System.out.println("2. println ");

            System.out.print("1. print ");
            System.out.print("2. print");
    }
}
```

When you run the program, the output will be:

```
1. println
2. println
1. print 2. print
```

To display integers, variables and so on, do not use quotation marks.

Example 3: Printing Variables and Literals

```
public class Variables {
    public static void main(String[] args) {

            Double number = -10.6;

            System.out.println(5);
            System.out.println(number);
    }
}
```

When you run the program, the output will be:

```
5
-10.6
```

You can + operator to concatenate strings and print it.

Example 4: Print Concatenated Strings

```
public class PrintVariables {
    public static void main(String[] args) {
```

```
        Double number = -10.6;

        System.out.println("I am " + "student.");
        System.out.println("Number = " + number);
    }
}
```

When you run the program, the output will be:

```
I am student.
Number = -10.6
```

Java Input

There are several ways to get input from the user in Java. You will learn to get input by using Scanner object in this article.

For that, you need to import Scanner class using:

```
import java.util.Scanner;
```

Learn more about Java import

Then, we will create an object of Scanner class which will be used to get input from the user.

```
Scanner input = new Scanner(System.in);
int number = input.nextInt();
```

Example 5: Get Integer Input From the User

```
import java.util.Scanner;

public class Input {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int number = input.nextInt();
```

```
            System.out.println("You entered " + number);
    }
}
```

When you run the program, the output will be:

```
Enter an integer: 23
You entered 23
```

Here, input object of Scanner class is created. Then, the nextInt() method of the Scanner class is used to get integer input from the user.

To get long, float, double and String input from the user, you can use nextLong(), nextFloat(), nextDouble() and next() methods respectively.

Example 6: Get float, double and String Input

```
import java.util.Scanner;

public class Input {
    public static void main(String[] args) {
            Scanner input = new Scanner(System.in);
        // Getting String input
            System.out.print("Enter text line: ");
            String myStrings = input.nextLine();
            System.out.println("Text line entered = " + myStrings);

            // Getting float input
            System.out.print("Enter float: ");
            float myFloat = input.nextFloat();
            System.out.println("Float entered = " + myFloat);

            // Getting double input
            System.out.print("Enter double: ");
            double myDouble = input.nextDouble();
            System.out.println("Double entered = " + myDouble);
```

```
        // Getting String input
        System.out.print("Enter text: ");
        String myString = input.next();
        System.out.println("Text entered = " + myString);
    }
}
```

When you run the program, the output will be:

```
Enter text line: R-Creation Track of IT
Text entered = R-Creation Track of IT
Enter float: 2.2
Float entered = 2.2
Enter double: 3.9998855
Double entered = 3.9998855
Enter text: Java
Text entered = Java
```

Java Command Line Arguments

The java command-line argument is an argument i.e. passed at the time of running the java program.

The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behavior of the program for the different values. You can pass **N** (1,2,3 and so on) numbers of arguments from the command prompt.

Simple example of command-line argument in java

In this example, we are receiving only one argument and printing it. To run this java program, you must pass at least one argument from the command prompt.

```
class CommandLineExample{
public static void main(String args[]){
```

```
System.out.println("Your first argument is: "+args[0]);
}
}
compile by > javac CommandLineExample.java
run by > java CommandLineExample R-Creation
```

Output:

```
Your first argument is: R-Creation
```