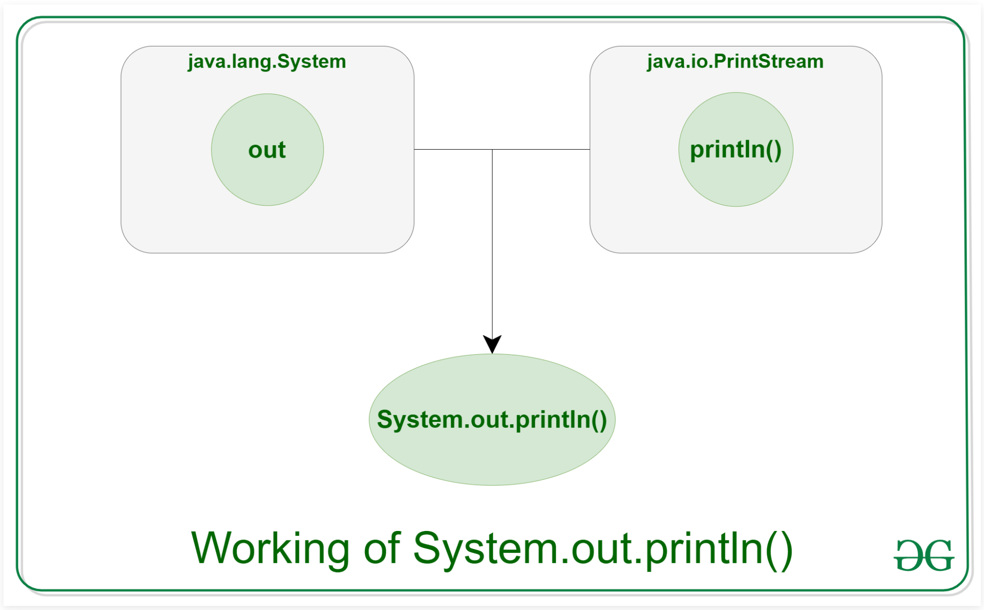
System.out.println in Java

Difficulty Level : Easy Last Updated : 28 Nov, 2019

Java **System.out.println()** is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood separately as:

1. System**:** It is a final class defined in the java.lang package.
2. **out:** This is an instance of PrintStream type, which is a public and static member field of the System class.
3. println()**:** As all instances of PrintStream class have a public method println(), hence we can invoke the same on out as well. This is an upgraded version of print(). It prints any argument passed to it and adds a new line to the output. We can assume that System.out represents the Standard Output Stream.



Syntax:

```
System.out.println(parameter)
```

Parameters: The parameter might be anything that the user wishes to print on the output screen.

Example 1:

```
// Java code to illustrate
// System.out.println();

import java.io.*;
```

```
class GFG {
    public static void main(String[] args)
    {
        System.out.println("Welcome");
        System.out.println("To");
        System.out.println("GeeksforGeeks");
    }
}
```

Output:

```
Welcome
To
GeeksforGeeks
```

Example 2:

```
// Java code to illustrate
// System.out.println();

import java.io.*;

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

        // Declaring variable
        int num1 = 10, num2 = 20, sum;

        // Printing the variables
        System.out.print("The addition of ");
        System.out.println(
            num1 + " and " + num2 + " is:");

        // Printing the result after operation
        System.out.println(num1 + num2);
    }
}
```

Output:

```
The addition of 10 and 20 is:
30
```

Just like **System.out**, Java provides us with two other standard or default **input-output streams**:

1. **System.in**: This is the standard input stream that is used to read characters from the keyboard or any other standard input device.
   Example:

   ```
   InputStreamReader inp = new InputStreamReader(System.in);
   ```

2. **System.err**: This is the standard error stream that is used to output all the error data that a program might throw, on a computer screen or any standard output device.
   Example:

   ```
   System.err.print("Error");
   ```

Overloads of println() method

As we know, Method Overloading in Java allows different methods to have the same name, but different signatures or parameters where each signature can differ by the number of input parameters or type of input parameters or both. From the use of println() we observed that it is a single method of PrintStream class that allows the users to print various types of elements by accepting different type and number of parameters.

For example:

```
System.out.println(),
System.out.println(int),
System.out.println(double),
System.out.println(string),
System.out.println(character),
etc.
```

PrintStream has around **10 different overloads of println() method** that are invoked based on the type of parameters passed by the user.

Example:

```
// Java code to illustrate method
```

```
// overloading in println()

import java.io.*;

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

        // Declaring different datatypes
        int num = 10;
        char ch = 'G';
        String str = "GeeksforGeeks";
        double d = 10.2;
        float f = 13.5f;
        boolean bool = true;

        // Various overloads of println() method
        System.out.println();
        System.out.println(num);
        System.out.println(ch);
        System.out.println(str);
        System.out.println(d);
        System.out.println(f);
        System.out.println(bool);
        System.out.println("Hello");
    }
}
```

Output:

```
10
G
GeeksforGeeks
10.2
13.5
true
Hello
```

Difference between System.out.print() and System.out.println()

System.out.print(): This method prints the text on the console and the cursor remains at the end of the text at the console. The next printing takes place from just here. This method must take atleast one parameter else it will throw an error.

System.out.println(): This method prints the text on the console and the cursor remains at the start of the next line at the console. The next printing takes place from the next line. This method may or may not take any parameter.

Example:

```java
// Java code to illustrate difference
// between print() and println()

import java.io.*;

class Demo_print {
    public static void main(String[] args)
    {
        System.out.println("Using print()");

        // using print()
        // all are printed in the
        // same line
        System.out.print("GfG! ");
        System.out.print("GfG! ");
        System.out.print("GfG! ");

        System.out.println();
        System.out.println();
        System.out.println("Using println()");

        // using println()
        // all are printed in the
        // different line
        System.out.println("GfG! ");
        System.out.println("GfG! ");
        System.out.println("GfG! ");
    }
}
```

Output:

```
Using print()
GfG! GfG! GfG!


Using println()
GfG!
GfG!
GfG!
```

Performance Analysis of System.out.println()

println() is a method that helps display output on a console. This might be dependent on various factors that drives the performance of this method. The message passed using println() is passed to the server's console where **kernel time** is required to execute the task. Kernel time refers to the **CPU time**. Since println() is a synchronized method, so when multiple threads are passed could lead to the **low-performance issue**. System.out.println() is a **slow operation** as it incurs heavy overhead on the machine compared to most IO operations.

There is an alternative way of performing output operations by invoking PrintWriter or the BufferedWriter class.

They are **fast as compared to the println()** of the PrintStream class.

Related Articles:

1. Difference between print() and println()
2. Input-Output in Java
3. PrintStream println() method in Java with Examples
4. Redirecting System.out.println() output to a file in Java