Java Basic Input and Output

Java Output

In Java, you can simply use

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

System.out.print(); or

System.out.printf();
```

to send output to standard output (screen).

Here,

- `System` is a class
- `out` is a `public` `static` field: it accepts output data.

Don't worry if you don't understand it. We will discuss `class`, `public`, and `static` in later chapters.

Let's take an example to output a line.

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

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

Run Code

Output:

```
Java programming is interesting.
```

Here, we have used the `println()` method to display the string.

Difference between println(), print() and printf()

- `print()` - It prints string inside the quotes.
- `println()` - It 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/C++ programming).

Example: print() and println()

```
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");
    }
}
```

Run Code

Output:

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

In the above example, we have shown the working of the `print()` and `println()` methods. To learn about the `printf()` method, visit Java printf().

Example: Printing Variables and Literals

```
class Variables {
```

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

        Double number = -10.6;

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

When you run the program, the output will be:

```
5
-10.6
```

Here, you can see that we have not used the quotation marks. It is because to display integers, variables and so on, we don't use quotation marks.

Example: Print Concatenated Strings

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

        Double number = -10.6;

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

Output:

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

In the above example, notice the line,

```
System.out.println("I am " + "awesome.");
```

Here, we have used the `+` operator to concatenate (join) the two strings: `"I am "` and `"awesome."`.
And also, the line,

```
System.out.println("Number = " + number);
```

Here, first the value of variable `number` is evaluated. Then, the value is concatenated to the string: `"Number = "`.

Java Input

Java provides different ways to get input from the user. However, in this tutorial, you will learn to get input from user using the object of `Scanner` class. In order to use the object of `Scanner`, we need to import `java.util.Scanner` package.

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

To learn more about importing packages in Java, visit Java Import Packages. Then, we need to create an object of the `Scanner` class. We can use the object to take input from the user.

```
// create an object of Scanner
Scanner input = new Scanner(System.in);

// take input from the user
int number = input.nextInt();
```

Example: Get Integer Input From the User

```
import java.util.Scanner;

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);

        // closing the scanner object
        input.close();
    }
}
```

Run Code

Output:

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

In the above example, we have created an object named `input` of the `Scanner` class. We then call the `nextInt()` method of the `Scanner` class to get an integer input from the user.

Similarly, we can use `nextLong()`, `nextFloat()`, `nextDouble()`, and `next()` methods to get `long`, `float`, `double`, and `string` input respectively from the user.

> **Note**: We have used the `close()` method to close the object. It is recommended to close the scanner object once the input is taken.

Example: Get float, double and String Input

```
import java.util.Scanner;

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

```
        Scanner input = new Scanner(System.in);

        // 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);
    }
}
```

Run Code

Output:

```
Enter float: 2.343
Float entered = 2.343
Enter double: -23.4
Double entered = -23.4
Enter text: Hey!
Text entered = Hey!
```

As mentioned, there are other several ways to get input from the user. To learn more about `Scanner`, visit Java Scanner.