Java Command-Line Arguments

The **command-line arguments** in Java allow us to pass arguments during the execution of the program.
As the name suggests arguments are passed through the command line.

Example: Command-Line Arguments

```
class Main {
  public static void main(String[] args) {
    System.out.println("Command-Line arguments are");

    // Print all arguments
    System.out.println(args[0]);
    System.out.println(args[1]);
    System.out.println(args[2]);
  }
}
```

Let's try to run this program using the command line.

1. To compile the code

```
javac Main.java
```

2. To run the code

```
java Main
```

Now suppose we want to pass some arguments while running the program, we can pass the arguments after the class name. For example,

```
java Main apple ball cat
```

Here `apple`, `ball`, and `cat` are arguments passed to the program through the command line. Now, we will get the following output.

```
Command-Line arguments are
Apple
Ball
Cat
```

In the above program, the `main()` method includes an array of string named `args` as its parameter.

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

The `String` array stores all the arguments passed through the command line.

> **Note**: Arguments are always stored as strings and always separated by **white-space**.

Passing Numeric Command-Line Arguments

The `main()` method of every Java program only accepts string arguments. Hence it is not possible to pass numeric arguments through the command line.
However, we can later convert string arguments into numeric values.

Example: Numeric Command-Line Arguments

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

    for(String str: args) {
      // convert into integer type
    int argument = Integer.parseInt(str);
    System.out.println("Argument in integer form: " + argument);
    }

  }
}
```

Let's try to run the program through the command line.

```
// compile the code
javac Main.java

// run the code
java Main 11 23
```

Here `11` and `23` are command-line arguments. Now, we will get the following output.

```
Arguments in integer form
11
23
```

In the above example, notice the line

```
int argument = Intege.parseInt(str);
```

Here, the `parseInt()` method of the `Integer` class converts the string argument into an integer.
Similarly, we can use the `parseDouble()` and `parseFloat()` method to convert the string into `double` and `float` respectively.

> **Note**: If the arguments cannot be converted into the specified numeric value then an exception named `NumberFormatException` occurs.