How to Take Input From User in Java?

Difficulty Level : Basic Last Updated : 20 Jan, 2022

Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files, etc to fully execute the I/O operations. There are **two ways** by which we can take input from the user or from a file

- BufferedReader Class
- Scanner Class

1. BufferedReader

It is a simple class that is used to read a sequence of characters. It has a simple function that reads a character another read which reads, an array of characters, and a readLine() function which reads a line.

InputStreamReader() is a function that converts the input stream of bytes into a stream of character so that it can be read as BufferedReader expects a stream of character.

BufferedReader can throw checked Exceptions

```
// Java Program for taking user
// input using BufferedReader Class
import java.io.*;

class GFG

{

  // Main Method
  public static void main(String [] args) throws IOException
  {
    // Creating BufferedReader Object
    // InputStreamReader converts bytes to
    // stream of character
    BufferedReader bfn = new BufferedReader(new InputStreamReader(System.in));

      // Asking for input from user
      System.out.println("Enter String : ");
      System.out.println("Enter Integer : ");
```

```
        // String reading internally
        String str = bfn.readLine();

        // Integer reading internally
        int it = Integer.parseInt(bfn.readLine());

        // Printing String
        System.out.println("Entered String : "+ str);

        // Printing Integer
        System.out.println("Entered Integer : "+ bfn);

      }

    }
```

Input:

```
GFG
123
```

Output:

```
Enter String :
Enter Integer :
Entered String : GFG
Entered Integer : 123
```

2. Scanner

It is an advanced version of BufferedReader which was added in later versions of Java. The scanner can read formatted input. It has different functions for different types of data types.

- The scanner is much easier to read as we don't have to write throws as there is no exception thrown by it.
- It was added in later versions of Java
- It contains predefined functions to read an Integer, Character, and other data types as well.

Syntax for Scanner

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

Syntax for importing Scanner Class: To use the Scanner we need to import the Scanner Class

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

Inbuilt Scanner functions are as follows

- Integer: nextInt()
- Float: nextFloat()
- String : readLine()

Hence, in the case of Integer and String in Scanner, we don't require parsing as we did require in BufferedReader.

```
// Java Program to show how to take
// input from user using Scanner Class

import java.util.*;

class GFG {

public static void main( String[] args )
{

    // Scanner definition
    Scanner scn= new Scanner(System.in);

    // input is a string
    // read by nextLine()function
    String str= scn.nextLine();

    // print string
    System.out.println("Entered String : "+ str);

    // input is an Integer
    // read by nextInt() function
    int x= scn.nextInt();

    // print integer
    System.out.println("Entered Integer : "+ x);

    // input is a floatingValue
    // read by nextFloat() function
```

```
        float f = scn.nextFloat();

        // print floating value
        System.out.println("Entered FloatValue : "+ f);
    }
    }
```

Input:

```
GFG
123
123.090
```

Output :

```
Entered String : GFG
Entered Integer : 123
Entered FloatValue : 123.090
```

Differences Between BufferedReader and Scanner

- BufferedReader is a very basic way to read the input generally used to read the stream of characters. Its gives an edge over Scanner as it is faster than Scanner because Scanner does lots of post-processing for parsing the input; as seen in nextInt(), nextFloat()
- BufferedReader is more flexible as we can specify the size of stream input to be read. (In general, it is there that BufferedReader reads lager input than Scanner)
- These two factors come into play when we are reading larger input. In general, the Scanner Class serves the input.
- BufferedReader is preferred as it is synchronized. While dealing with multiple threads it is preferred.
- For decent input, easy readability. The Scanner is preferred over BufferedReader.