Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No.2	
Accepting Input Through Keyboard	
Date of Performance:	
Date of Submission:	

Aim: To apply basic programing for accepting input through keyboard.

Objective: To use the facility of java to read data from the keyboard for any program

Theory:

Java brings various Streams with its I/O package that helps the user perform all the Java input-output operations. These streams support all types of objects, data types, characters, files, etc. to fully execute the I/O operations. Input in Java can be with certain methods mentioned below in the article.

Methods to Take Input in Java

There are two ways by which we can take Java input from the user or from a file

1. BufferedReader Class

2. Scanner Class

Using BufferedReader Class for String Input In Java

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 characters so that it can be read as BufferedReader expects a stream of characters. BufferedReader can throw checked Exceptions.

Using Scanner Class for Taking Input in Java

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 of Scanner class

Scanner scn = new Scanner(System.in);

Code:

<u>Using BufferedReader Class :</u>

```
import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;


public class Buff{

public static void main(String args[])

{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

try

{

System.out.print("Enter a line of text: ");

String input = reader.readLine();

System.out.println("You entered: " + input);

} catch (IOException e)

{

System.err.println("An error occurred while reading input: " + e.getMessage());

}
```

```java
finally

 {

try {

reader.close();



} catch (IOException e) {



System.err.println("An error occurred while closing the BufferedReader: " +



e.getMessage());

        }

    }

  }

}
```

```
Output

java -cp /tmp/hXQsrp9UkP/Buff
Enter a line of text: iamvcetcollegestudent
You entered: iamvcetcollegestudent

=== Code Execution Successful ===
```

Using Scanner Class:

import java.util.Scanner;

```java
public class Add{

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the first number: ");

        int num1 = sc.nextInt();

        System.out.print("Enter the second number: ");

        int num2 = sc.nextInt();

        System.out.print("Enter the third number: ");

        int num3 = sc.nextInt();
        int sum = num1 + num2 + num3;

        System.out.println("The sum of the three numbers is: " + sum);

        sc.close();

    }

}
```

Output

```
java -cp /tmp/dqVIQr4oU2/Add
Enter the first number: 25
Enter the second number: 35
Enter the third number: 20
The sum of the three numbers is: 80

=== Code Execution Successful ===
```

Conclusion:

Buffered Reader:

More efficient for large data or text reading, better suited for file and stream operations.

Scanner:

User-friendly for various input types, best for simple and interactive input

scenarios.