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.

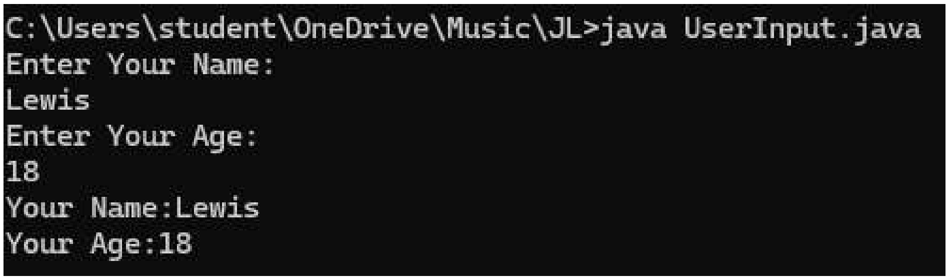
Syntax of Scanner class

Scanner scn = new Scanner(System.in);

Code:

```
import java.io.*;

import java.util.*;

public class UserInput

{

    public static void main(String[] args)

    {

        Scanner S = new Scanner(System.in);

        System.out.println("Enter Your Name:");

        String name = S.nextLine();

        System.out.println("Enter Your Age:");

        int age = S.nextInt();

        System.out.println("Your Name:"+ name);

        System.out.println("Your Age:"+ age);

    }
}
```

Output:-

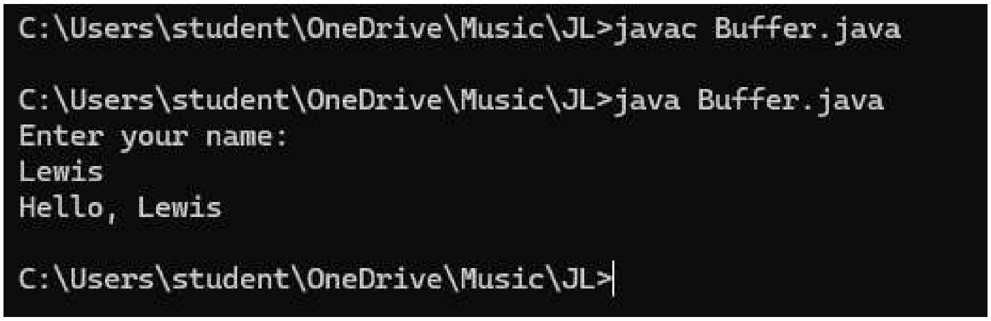


```
C:\Users\student\OneDrive\Music\JL>java UserInput.java
Enter Your Name:
Lewis
Enter Your Age:
18
Your Name:Lewis
Your Age:18
```

Code Using BufferClass:

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;


public class Buffer

{

    public static void main(String[] args) throws IOException

    {

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

        System.out.println("Enter your name:");


            String name = reader.readLine();

            System.out.println("Hello, " + name);

    }
}
```

Output:-



```
C:\Users\student\OneDrive\Music\JL>javac Buffer.java

C:\Users\student\OneDrive\Music\JL>java Buffer.java
Enter your name:
Lewis
Hello, Lewis

C:\Users\student\OneDrive\Music\JL>
```

Conclusion:

Comment on how you have used BufferedReader and Scanner Class for accepting user input:

Usage of Scanner Class:

- Initialization: The Scanner object A is created with new Scanner(System.in), which sets up the scanner to read input from the standard input stream (keyboard).
- Reading Input: The nextInt() method of the Scanner class is used to read two integers from the user. This method simplifies input handling as it directly parses and returns integer values.
- Comparison and Output: The program then compares the two integers and prints the larger one using basic conditional statements.

Comparison with BufferedReader:

- Ease of Use: The Scanner class provides a more straightforward and user-friendly approach for input handling compared to BufferedReader. It allows direct reading and parsing of various data types (e.g., integers, doubles, strings) without the need for additional conversion steps.
- Exception Handling: Unlike BufferedReader, which requires explicit handling of IOException, Scanner simplifies error management by not throwing checked exceptions, making the code cleaner and easier to write.
- Functionality: BufferedReader is efficient for reading large amounts of text and lines but requires manual conversion of strings to other data types (e.g., using Integer.parseInt() for integers). In contrast, Scanner provides built-in methods for parsing different data types directly, making it a more versatile tool for interactive applications.