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.

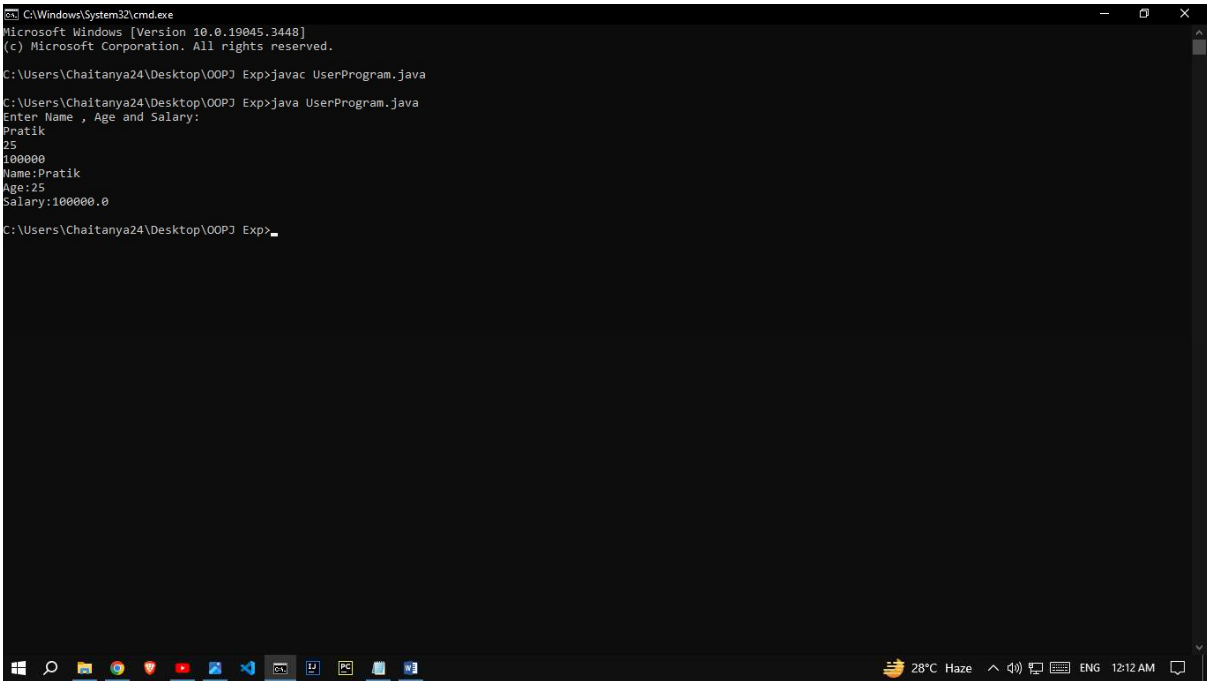
Syntax of Scanner class

Scanner scn = new Scanner(System.in);

Code:

1} Scanner class

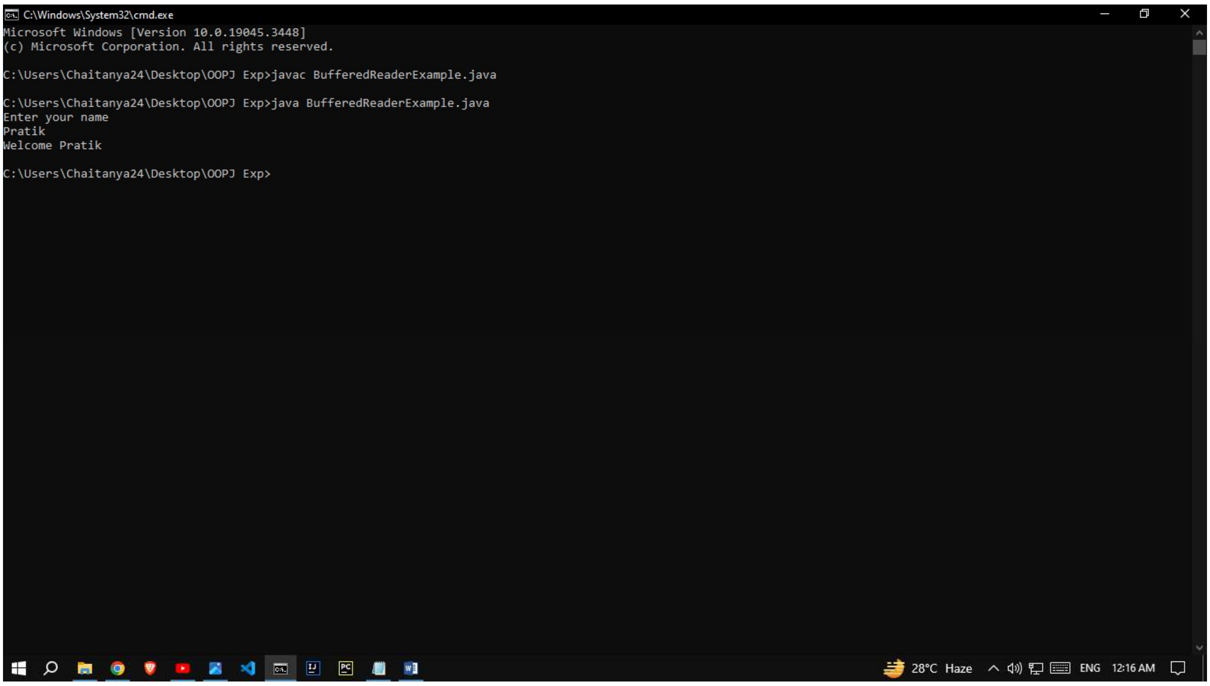
```
import java.util.Scanner;

class UserProgram

{

  public static void main(String args[])

   {

    Scanner a = new Scanner(System.in);

    System.out.println("Enter Name , Age and Salary:");

    String str = a.nextLine();

    int age = a.nextInt();

    Double salary = a.nextDouble();

    System.out.println("Name:" + str);

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

    System.out.println("Salary:" + salary);

   }

}
```



```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Chaitanya24\Desktop\OOPJ Exp>javac UserProgram.java

C:\Users\Chaitanya24\Desktop\OOPJ Exp>java UserProgram.java
Enter Name , Age and Salary:
Pratik
25
100000
Name:Pratik
Age:25
Salary:100000.0

C:\Users\Chaitanya24\Desktop\OOPJ Exp>
```

2} Buffer reader class

```
package com.javatpoint;
import java.io.*;
public class BufferedReaderExample{
public static void main(String args[])throws Exception{
    InputStreamReader r=new InputStreamReader(System.in);
    BufferedReader br=new BufferedReader(r);
    System.out.println("Enter your name");
    String name=br.readLine();
    System.out.println("Welcome "+name);
}
}
```



```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Chaitanya24\Desktop\OOPJ Exp>javac BufferedReaderExample.java

C:\Users\Chaitanya24\Desktop\OOPJ Exp>java BufferedReaderExample.java
Enter your name
Pratik
Welcome Pratik

C:\Users\Chaitanya24\Desktop\OOPJ Exp>
```

Conclusion:

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

- **BufferedReader:**BufferedReader is a class used for reading text from character-input streams. It's often used for reading user input from the console.It's typically wrapped around an InputStreamReader to read from standard input (System.in).BufferedReader is efficient for reading large volumes of text because it buffers the input, reducing the number of I/O operations.When you use BufferedReader, you can use its readLine() method to read a whole line of text as a String. This is useful for processing textual input.The readLine() method can throw an IOException, so it's essential to handle exceptions appropriately to ensure the program's robustness.
- **Scanner:** Scanner is a class that provides various methods for parsing different data types from an input source, including the console. It's convenient for

parsing user input into different data types like integers, doubles, and strings. Scanner can tokenize input, breaking it into smaller pieces based on delimiters (default delimiter is whitespace). You can create a Scanner object to read from standard input (System.in), and then use methods like nextInt(), nextDouble(), and nextLine() to read specific data types. Scanner can be resource-intensive for reading large volumes of text because it's not as efficient as BufferedReader. It provides more user-friendly input methods that don't require explicit parsing like BufferedReader.