BufferedReader Class

So far we have learnt to take input with the help of Scanner class. There is another way to take input: using BufferedReader class. Let us dive deep into it.

BufferedReader:

It can be used to read input from a file as well as a keyboard. Since, throughout our course, since our source of input will be keyboard only, therefore, we will limit our discussions to taking input from keyboard.

Using this method, we read characters from the input stream. As we know, we have three streams:

1. System.in
2. System.out
3. System.err

In this method, we will use a class called InputStreamReader, which takes input byte by byte and decodes it into a character stream. After reading data from the source keyboard, the decoded data (character stream) is stored in a buffer (storage meant for temporary usage) and then the object of class BufferedReader reads from this buffer. You will get to know about this in detail in the lecture of Object Oriented Programming (OOP). It is explained in the OOP lecture that non-static functions or methods cannot be accessed by class. For accessing and invoking non-static methods, we use objects of class.

Note:-

1. Since, methods of BufferedReader class and InputStreamReader deal in input and output operations, therefore, these methods may lead to errors in reading input or writing output. Therefore, the function must throw IOException.
2. BufferedReader and InputStreamReader are in the “io” package. Therefore, following statements must be included at the top of the code:

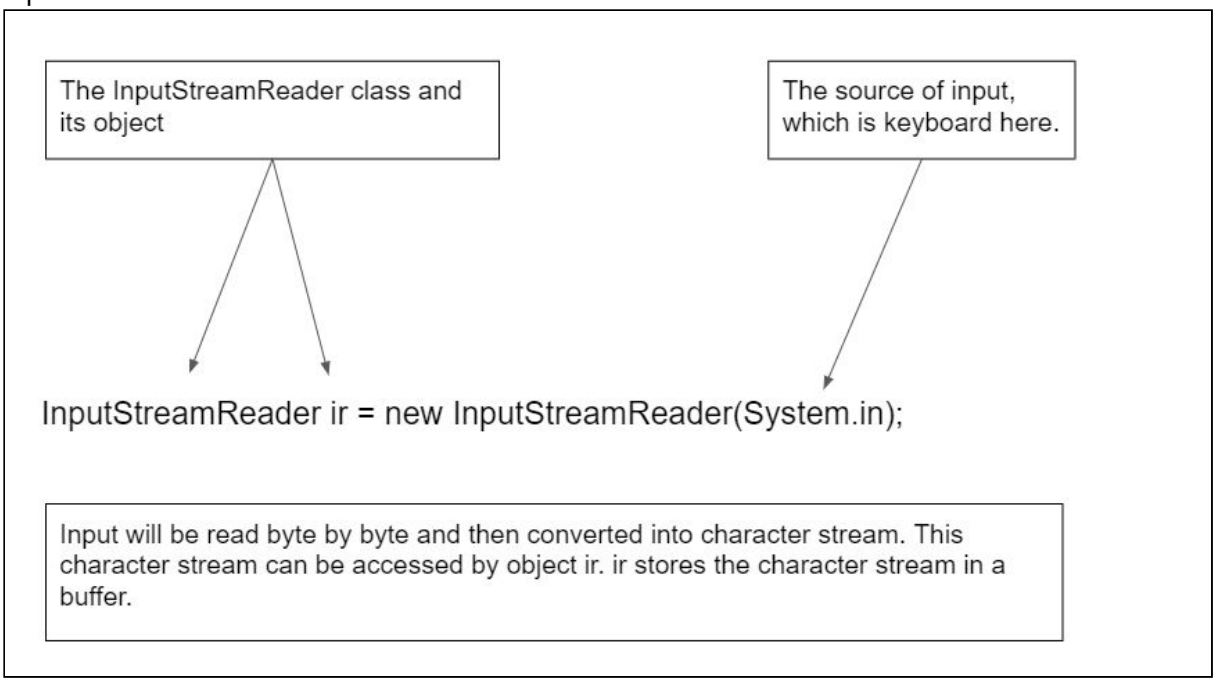
   ```
   import java.io.BufferedReader;
   import java.io.IOException;
   import java.io.InputStreamReader;
   ```

   Or alternatively, we can include: `import java.io.*;`

Process

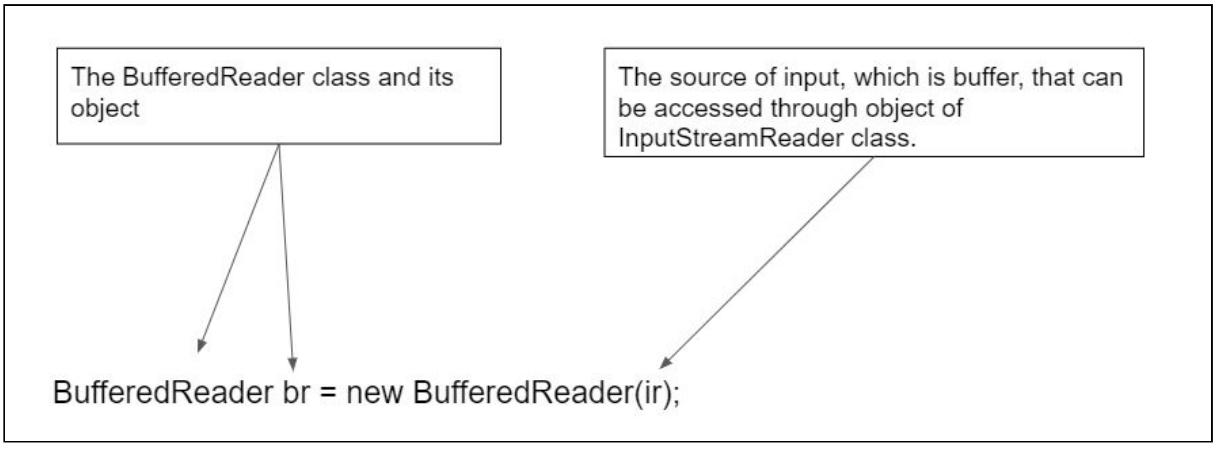
The complete can be divided into two steps:

1. In first step, we have to read the input or access the input using InputStreamReader



2. Now, in the second step, we have to read data from the buffer. This can be done using the object of the BufferedReader class. BufferedReader can read only characters or string. It does so using the following two methods:
    a. read(): reads only single character
    b. readLine(): reads multiple character or a string

    The following syntax is used to read character using BufferedReader class:



How to take input of integer and floating point numbers

As BufferedReader method can only be used to find reading characters or string, therefore, for reading integers or floating-point values, we have to first read the input in the form of characters and then typecast it into integer or floating-point values.

We will static function parseInt for type casting character into integers and similarly, parseFloat for floating point values.

Examples:

```
int a = Integer.parseInt(br.readLine());
float b = Float.parseFloat(br.readLine());
String str = br.readLine();
```

Before getting our hands dirty in a more complex input format, we have to discuss split function. This function splits the given string on a certain delimiter. For example, if the delimiter is space, then it will divide the string into smaller substrings, which are separated by space. It will return an array of those substrings.

Example:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

	public static void main(String[] args) throws NumberFormatException,
IOException {
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

		String str = br.readLine();
		String[] strNums = str.split(" ");

		for (int i = 0; i < strNums.length; i++) {
			System.out.print(strNums[i]+);
		}

	}

}
```

Example Input:

11 12 13 14

Example Output:

11 12 13 14

More Examples

Let us suppose that we have to read input with the following input format:
“The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains an integer 'N' representing the size of the array/list.
Second line contains 'N' single space separated integers representing the elements in the array/list.”

One Example of this input format is:

2
5
9 3 6 2 0
4
2 3 1 2

For reading this input, following code will be used:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
	/*
		The object of class BufferedReader is made static because it
		is being used by multiple functions.
	*/
	static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

	public static int[] takeInput() throws IOException {
		int size = Integer.parseInt(br.readLine());
		int[] input = new int[size];

		if (size == 0) {
			return input;
		}

		String[] strNums;
		strNums = br.readLine().split("\\s");

		for (int i = 0; i < size; ++i) {
			input[i] = Integer.parseInt(strNums[i]);
		}

		return input;
	}

	public static void printArray(int[] arr) {
		for (int i = 0; i < arr.length; i++) {
```

```
                System.out.print(arr[i] + " ");
            }

            System.out.println();
        }

        public static void main(String[] args) throws NumberFormatException,
IOException {
            int t = Integer.parseInt(br.readLine());

            while (t > 0) {

                int[] input = takeInput();
                printArray(input);

                t -= 1;
            }
        }

}
```