Today we are going to learn following topics

Regular expressions
Basics of I/O
Recursion

Let Us start Basics Of I/O

Java I/O (Input and Output) is used to process the input and produce the output.

Java uses the concept of a stream to make I/O operation fast.

The java.io package contains all the classes required for input and output operations.

We can perform file handling in Java by Java I/O API.

Stream

A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

In Java, 3 streams are created for us automatically. All these streams are attached with the console.

1) **System.out**: standard output stream
2) **System.in**: standard input stream
3) **System.err**: standard error stream

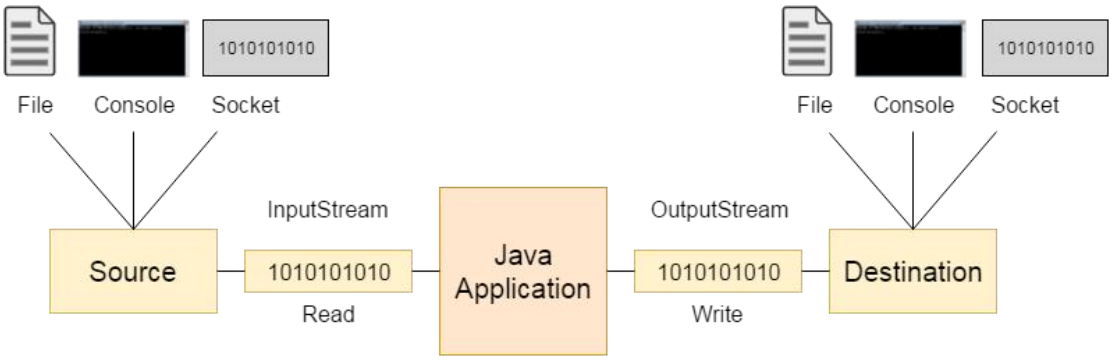
OutputStream

Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.

InputStream

Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

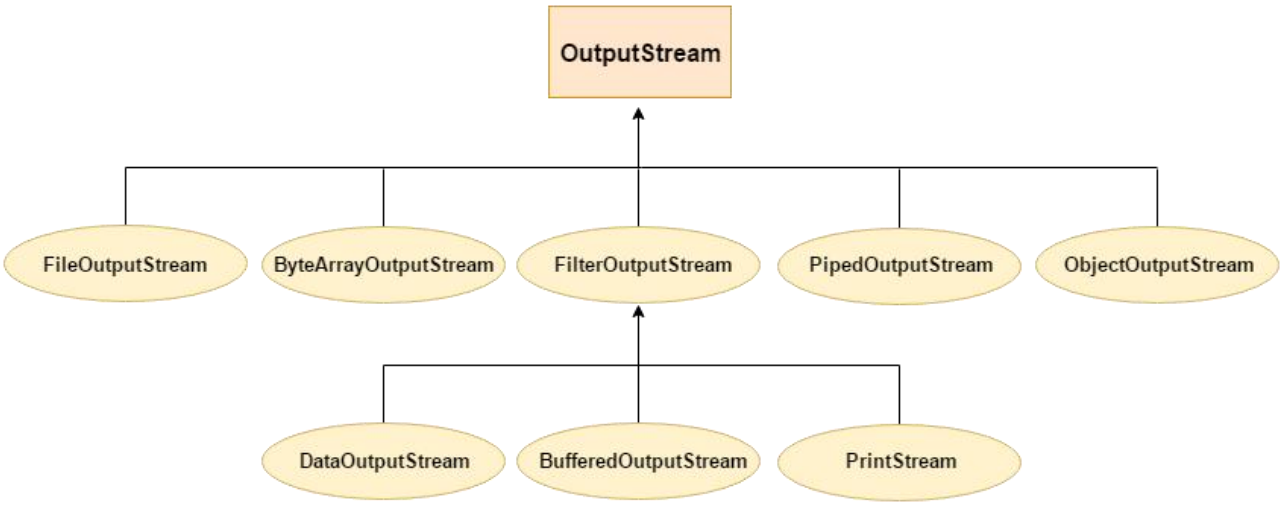
Let's understand the working of Java OutputStream and InputStream by the figure given below.



OutputStream class

OutputStream class is an abstract class. It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Java OutputStream Hierarchy



Useful Methods of OutPutStream Class

Method	Description
1) public void write(int)throws IOException	is used to write a byte to the current output stream.
2) public void write(byte[])throws IOException	is used to write an array of byte to the current output stream.
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

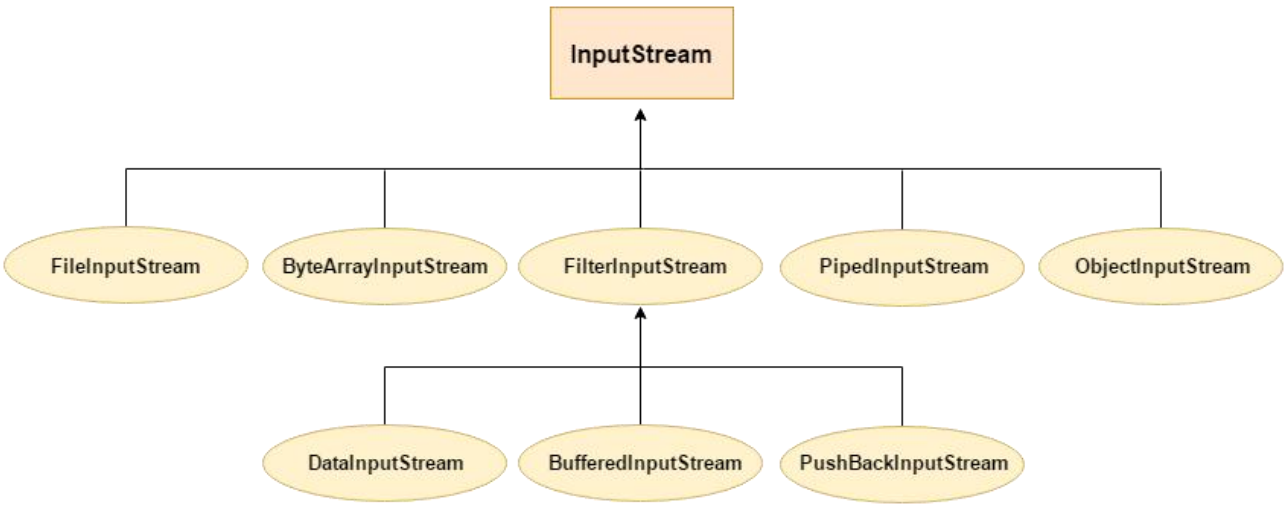
InputStream class

InputStream class is an abstract class. It is the superclass of all classes representing an input stream of bytes.

Useful Methods of InputStream Class

Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -1 at the end of the file.
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read from the current input stream.
3) public void close()throws IOException	is used to close the current input stream.

Java InputStream Hierarchy



What is Regular Expressions

In Theoretical term :- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations.

Java does not have a built-in Regular Expression class, but we can import the java.util.regex package to work with regular expressions. The package includes the following classes:

Pattern Class - Defines a pattern (to be used in a search)
Matcher Class - Used to search for the pattern
PatternSyntaxException Class - Indicates syntax error in a regular expression pattern

Pattern Class - Defines a pattern (to be used in a search)

The first parameter of the Pattern.compile() method is the pattern. It describes what is being searched for.
Brackets are used to find a range of characters:

Expression Description
[abc] Find one character from the options between the brackets
[^abc] Find one character NOT between the brackets
[0-9] Find one character from the range 0 to 9

Matcher Class - Used to search for the pattern

The matcher() method is used to search for the pattern in a string. It returns a Matcher object which contains information about the search that was performed. The find() method returns true if the pattern was found in the string and false if it was not found.

PatternSyntaxException Class - Indicates syntax error in a regular expression pattern

A PatternSyntaxException is an unchecked exception that indicates a syntax error in a regular expression pattern. The PatternSyntaxException class provides the following methods to help you determine what went wrong:

public String getDescription(): Retrieves the description of the error.
public int getIndex(): Retrieves the error index.
public String getPattern(): Retrieves the erroneous regular expression pattern.
public String getMessage(): Returns a multi-line string containing the description of the syntax error and its index, the erroneous regular-expression pattern, and a visual indication of the error index within the pattern.

This was theoretical explanation of Regular Expression , Let Me give you a life based practical example

Java Recursion

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Recursion may be a bit difficult to understand. The best way to figure out how it works is to experiment with it.

Recursion Example

Adding two numbers together is easy to do, but adding a range of numbers is more complicated. In the following example, recursion is used to add a range of numbers together by breaking it down into the simple task of adding two numbers:

Example :- Use recursion to add all of the numbers up to 10.

```
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

Example Explained

When the sum() function is called, it adds parameter k to the sum of all numbers smaller than k and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0

Result = 55

Halting Condition

Just as loops can run into the problem of infinite looping, recursive functions can run into the problem of infinite recursion. Infinite recursion is when the function never stops calling itself. Every recursive function should have a halting condition, which is the condition where the function stops calling itself. In the previous example, the halting condition is when the parameter k becomes 0.

It is helpful to see a variety of different examples to better understand the concept. In this example, the function adds a range of numbers between a start and an end. The halting condition for this recursive function is when end is not greater than start:

Example

Use recursion to add all of the numbers between 5 to 10.

```
public class Main {
  public static void main(String[] args) {
    int result = sum(5, 10);
    System.out.println(result);
  }
  public static int sum(int start, int end) {
    if (end > start) {
      return end + sum(start, end - 1);
    } else {
      return end;
    }
  }}
```

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
…
…
10 + 9 + 8 + 7 + 6 + 5

Result = 45

Question On I/O

- How to write a common data to multiple files using a single stream only?
- How can we access multiple files by a single stream?
- How can we improve the performance of Input and Output operation?
- How many ways can we read data from the keyboard?
- What does the console class?
- How to compress and uncompress the data of a file?

Question On Regular Expressions

1. Write a Java program to check whether a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

2. Write a Java program that matches a string that has a p followed by zero or more q's.

3. Write a Java program to find sequences of lowercase letters joined with a underscore.

4. Write a Java program to find the sequences of one upper case letter followed by lower case letters.

5. Write a Java program that matches a string that has a 'p' followed by anything, ending in 'q'.

6. Write a Java program to check a word contains the character 'g' in a given string.

7. Write a Java program that matches a word containing 'g', not at the start or end of the word.

8. Write a Java program to match a string that contains only upper and lowercase letters, numbers, and underscores.

9. Write a Java program where a string starts with a specific number.

10. Write a Java program to remove leading zeros from a given IP address.

11. Write a Java program to check for a number at the end of a given string. **12.** Write a Java program to replace Python with Java and code with coding in a given string.

13. Write a Java program to find the word Python in a given string, if the word Python present in the string return Java otherwise return C++. Ignore case sensitive.

14. Write a Java program to count number of vowels in a given string using regular expression.

15. Write a Java program to remove all the vowels of a given string. Return the new string.

16. Write a Java program to replace all the vowels in a given string with a specified character.

17. Write a Java program to count the number of decimal places in a given number.

18. Write a Java program to validate a personal identification number (PIN). Assume the length of a PIN number is 4, 6 or 8.

19. Write a Java program to remove the specific letters from a string and return the new string. Specific letters: "p", "q", or "r".

20. Write a Java program that takes a number and set thousands separator in that number.

21. Write a Java program to remove all non-alphanumeric characters from a given string.

22. Write a Java program to validate a given phone number. **23.** Write a Java program to move all lower case letters to the front of a given word keeping the relative position all the letters(both upper and lower case) same.

24. Write a Java program to separate consonants and vowels from a given string.

25. Write a Java program to get last n vowels of a given string.

26. Write a Java program to check whether a given string is a valid hex code or not.

27. Write a Java program to add a dash before and after every vowel in a given string.

28. Write a Java program to reverse the words of length higher than 3 in a given string.

29. Write a Java program to check if a given string is a Mathematical Expression or not.

30. Write a Java program to insert a dash (-) between an upper case letter and a lower case letter in a given string.

Question On Java Recursion

1. Write a program in Java to calculate the sum of all prime numbers between 1-100.
2. Write a program in Java to calculate the sum of all even numbers between 1-100.
3. Write a program in Java to calculate the sum of all odd numbers between 1-100.
4. Write a program in Java to calculate the sum of all numbers that are dividable by 5 from 1-100.
5. Write a program in Java to to print the following pattern in console

```
*
**
***
****
*****
```

6. Write a program in Java to to print the following pattern in console

```
    *
   **
  ***
 ****
*****
```

7. Write a program in Java to to print the following pattern in console

```
*****
****
***
**
*
```

8. Write a program in Java to to print the following pattern in console

```
*****
 ****
  ***
   **
    *
```