Java InputStream Class

In this tutorial, we will learn about the Java InputStream class and its methods with the help of an example.

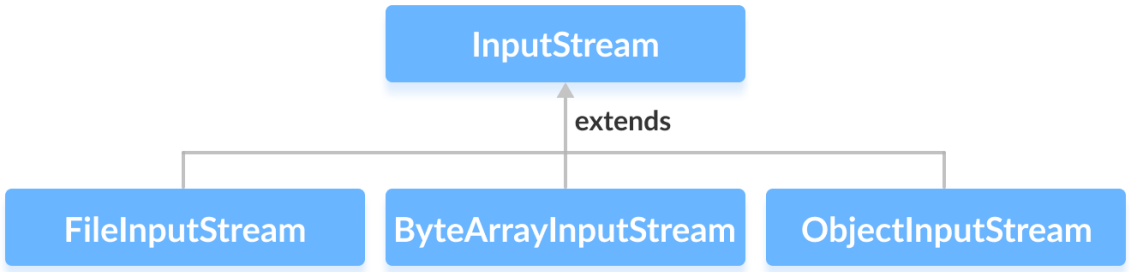
The `InputStream` class of the `java.io` package is an abstract superclass that represents an input stream of bytes.

Since `InputStream` is an abstract class, it is not useful by itself. However, its subclasses can be used to read data.

Subclasses of InputStream

In order to use the functionality of `InputStream`, we can use its subclasses. Some of them are:

- FileInputStream
- ByteArrayInputStream
- ObjectInputStream



We will learn about all these subclasses in the next tutorial.

Create an InputStream

In order to create an InputStream, we must import the `java.io.InputStream` package first. Once we import the package, here is how we can create the input stream.

```
// Creates an InputStream
InputStream object1 = new FileInputStream();
```

Here, we have created an input stream using `FileInputStream`. It is because `InputStream` is an abstract class. Hence we cannot create an object of `InputStream`.

Note: We can also create an input stream from other subclasses of `InputStream`.

Methods of InputStream

The `InputStream` class provides different methods that are implemented by its subclasses. Here are some of the commonly used methods:

- `read()` - reads one byte of data from the input stream
- `read(byte[] array)` - reads bytes from the stream and stores in the specified array
- `available()` - returns the number of bytes available in the input stream
- `mark()` - marks the position in the input stream up to which data has been read
- `reset()` - returns the control to the point in the stream where the mark was set
- `markSupported()` - checks if the `mark()` and `reset()` method is supported in the stream
- `skips()` - skips and discards the specified number of bytes from the input stream
- `close()` - closes the input stream

Example: InputStream Using FileInputStream

Here is how we can implement `InputStream` using the `FileInputStream` class.

Suppose we have a file named **input.txt** with the following content.

```
This is a line of text inside the file.
```

Let's try to read this file using `FileInputStream` (a subclass of `InputStream`).

```java
import java.io.FileInputStream;
import java.io.InputStream;

public class Main {
    public static void main(String args[]) {

        byte[] array = new byte[100];

        try {
            InputStream input = new FileInputStream("input.txt");

            System.out.println("Available bytes in the file: " + input.available());

            // Read byte from the input stream
            input.read(array);
            System.out.println("Data read from the file: ");

            // Convert byte array into string
            String data = new String(array);
            System.out.println(data);

            // Close the input stream
            input.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

Output

```
Available bytes in the file: 35
Data read from the file:
This is a line of text inside the file
```

In the above example, we have created an input stream using the `FileInputStream` class. The input stream is linked with the file **input.txt**.

```
InputStream input = new FileInputStream("input.txt");
```

To read data from the **input.txt** file, we have implemented these two methods.

```
input.read(array);      // to read data from the input stream
input.close();             // to close the input stream
```