Unit :1

Advance java.

Input and output classes.

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
Let's see the code to print **output and an error** message to the console.

1. System.out.println("simple message");
2. System.err.println("error message");

Let's see the code to get **input** from console.

1. **int** i=System.in.read();//returns ASCII code of 1st character
2. System.out.println((**char**)i);//will print the character

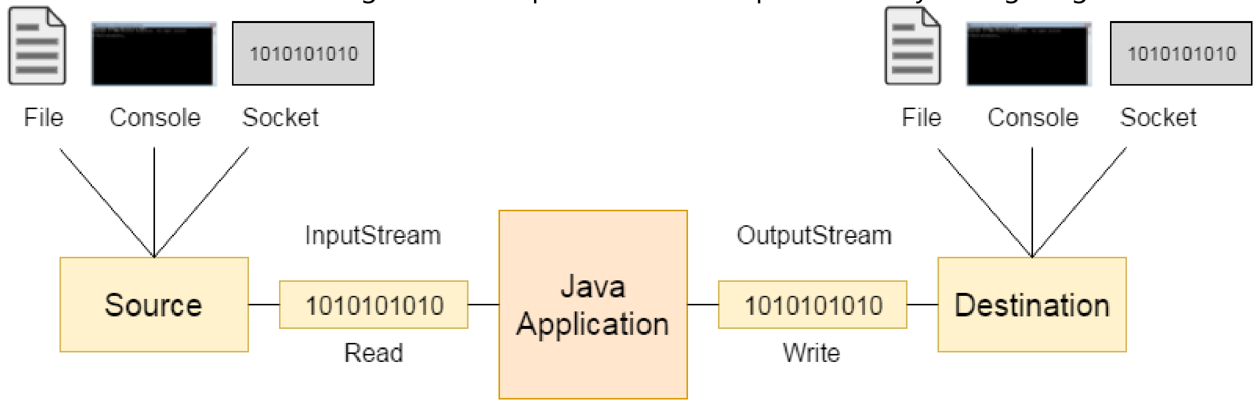
OutputStream vs InputStream

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.



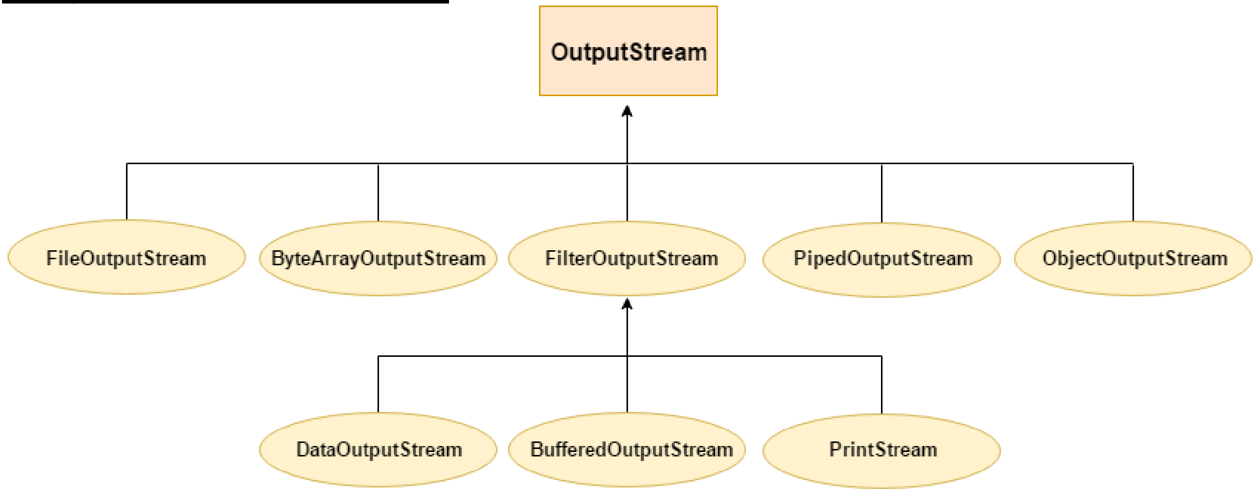
OutputStream class

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.

Useful methods of OutputStream

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.

OutputStream Hierarchy



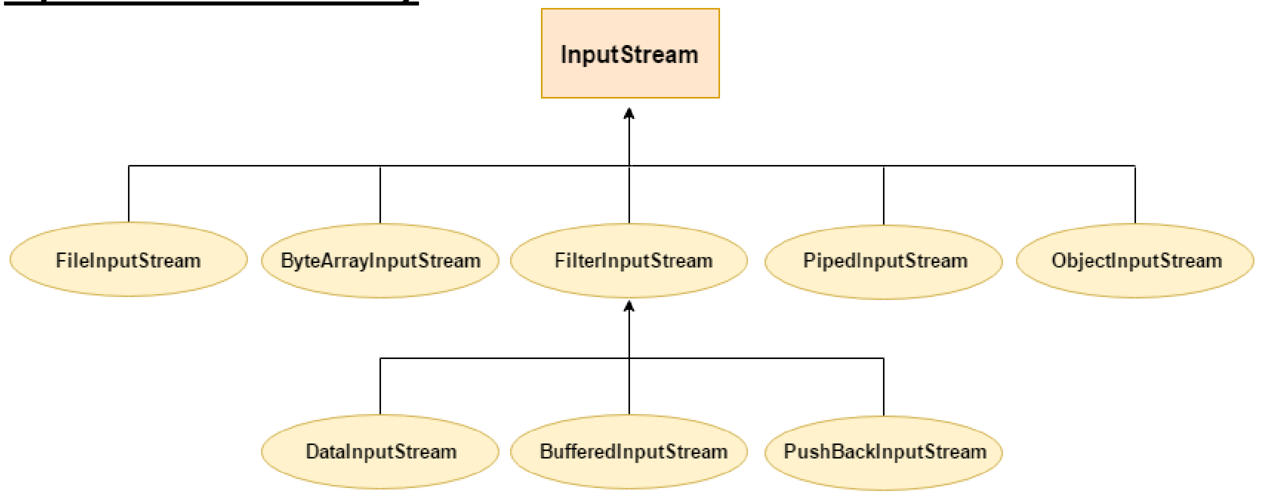
InputStream class

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

Useful methods of InputStream

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.

InputStream Hierarchy



File class

Java File Class

The File class is an abstract representation of file and directory pathname. A pathname can be either absolute or relative.

The File class have several methods for working with directories and files such as creating new directories or files, deleting and renaming directories or files, listing the contents of a directory etc.

Fields

Modifier	Type	Field	Description
static	String	pathSeparator	It is system-dependent path-separator character, represented as a string for convenience.
static	char	pathSeparatorChar	It is system-dependent path-separator character.
static	String	separator	It is system-dependent default name-separator character, represented as a string for convenience.
static	char	separatorChar	It is system-dependent default name-separator character.

Constructors

Constructor	Description
File(File parent, String child)	It creates a new File instance from a parent abstract pathname and a child pathname string.
File(String pathname)	It creates a new File instance by converting the given pathname string into an abstract pathname.
File(String parent, String	It creates a new File instance from a parent pathname string and a

child)	child pathname string.
File(URI uri)	It creates a new File instance by converting the given file: URI into an abstract pathname.

Useful Methods

Modifier and Type	Method	Description
static File	createTempFile(String prefix, String suffix)	It creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.
boolean	createNewFile()	It atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
boolean	canWrite()	It tests whether the application can modify the file denoted by this abstract pathname.String[]
boolean	canExecute()	It tests whether the application can execute the file denoted by this abstract pathname.
boolean	canRead()	It tests whether the application can read the file denoted by this abstract pathname.
boolean	isAbsolute()	It tests whether this abstract pathname is absolute.
boolean	isDirectory()	It tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	It tests whether the file denoted by this abstract pathname is a normal file.
String	getName()	It returns the name of the file or directory denoted by this abstract pathname.
String	getParent()	It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
Path	toPath()	It returns a java.nio.file.Path object constructed from the this abstract path.

URI	toURI()	It constructs a file: URI that represents this abstract pathname.
File[]	listFiles()	It returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname
long	getFreeSpace()	It returns the number of unallocated bytes in the partition named by this abstract path name.
String[]	list(FilenameFilter filter)	It returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
boolean	mkdir()	It creates the directory named by this abstract pathname.

Class example:

```
import java.io.*;
public class FileDemo {
   public static void main(String[] args) {

      try {
         File file = new File("javaFile123.txt");
         if (file.createNewFile()) {
            System.out.println("New File is created!");
         } else {
            System.out.println("File already exists.");
         }
      } catch (IOException e) {
         e.printStackTrace();
      }

   }
}
```

ByteStream Classes in Java

ByteStream classes are used to read bytes from the input stream and write bytes to the output stream. In other words, we can say that ByteStream classes read/write the data of 8-bits. We can store video, audio, characters, etc., by using ByteStream classes. These classes are part of the java.io package.

The ByteStream classes are divided into two types of classes, i.e., InputStream and OutputStream. These classes are abstract and the super classes of all the Input/Output stream classes.

InputStream Class

The InputStream class provides methods to read bytes from a file, console or memory. It is an abstract class and can't be instantiated; however, various classes inherit the InputStream class and override its methods. The subclasses of InputStream class are given in the following table.

SN	Class	Description
1	BufferedInputStream	This class provides methods to read bytes from the buffer.
2	ByteArrayInputStream	This class provides methods to read bytes from the byte array.
3	DataInputStream	This class provides methods to read Java primitive data types.
4	FileInputStream	This class provides methods to read bytes from a file.
5	FilterInputStream	This class contains methods to read bytes from the other input streams, which are used as the primary source of data.
6	ObjectInputStream	This class provides methods to read objects.
7	PipedInputStream	This class provides methods to read from a piped output stream to which the piped input stream must be connected.
8	SequenceInputStream	This class provides methods to connect multiple Input Stream and read data from them.

The InputStream class contains various methods to read the data from an input stream. These methods are overridden by the classes that inherit the InputStream class. However, the methods are given in the following table.

SN	Method	Description
1	int read()	This method returns an integer, an integral representation of the next available byte of the input. The integer -1 is returned once the end of the input is encountered.
2	int read (byte buffer [])	This method is used to read the specified buffer length bytes from the input and returns the total number of bytes successfully read. It returns -1 once the end of the input is encountered.
3	int read (byte buffer [], int loc, int nBytes)	This method is used to read the 'nBytes' bytes from the buffer starting at a specified location, 'loc'. It returns the total number of bytes successfully read from the input. It returns -1 once the end of the input is encountered.
4	int available ()	This method returns the number of bytes that are available to read.
5	Void mark(int nBytes)	This method is used to mark the current position in the input stream until the specified nBytes are read.
6	void reset ()	This method is used to reset the input pointer to the previously set mark.
7	long skip (long	This method is used to skip the nBytes of the input stream and

	nBytes)	returns the total number of bytes that are skipped.
8	void close ()	This method is used to close the input source. If an attempt is made to read even after the closing, IOException is thrown by the method.

OutputStream Class

The OutputStream is an abstract class that is used to write 8-bit bytes to the stream. It is the superclass of all the output stream classes. This class can't be instantiated; however, it is inherited by various subclasses that are given in the following table.

SN	Class	Description
1	BufferedOutputStream	This class provides methods to write the bytes to the buffer.
2	ByteArrayOutputStream	This class provides methods to write bytes to the byte array.
3	DataOutputStream	This class provides methods to write the java primitive data types.
4	FileOutputStream	This class provides methods to write bytes to a file.
5	FilterOutputStream	This class provides methods to write to other output streams.
6	ObjectOutputStream	This class provides methods to write objects.
7	PipedOutputStream	It provides methods to write bytes to a piped output stream.
8	PrintStream	It provides methods to print Java primitive data types.

The OutputStream class provides various methods to write bytes to the output streams. The methods are given in the following table.

SN	Method	Description
1	void write (int i)	This method is used to write the specified single byte to the output stream.
2	void write (byte buffer [])	It is used to write a byte array to the output stream.
3	Void write(bytes buffer[],int loc, int nBytes)	It is used to write nByte bytes to the output stream from the buffer starting at the specified location.
4	void flush ()	It is used to flush the output stream and writes the pending buffered bytes.
5	void close ()	It is used to close the output stream. However, if we try to close the already closed output stream, the IOException will be thrown by this method.

Example:

The following example uses the ByteArrayInputStream to create an input stream from a byte array "content". We use the read() method to read the content from an input stream. We have also used the write() method on a FileOutputStream object to write the byte array content in the file.

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;


public class InputOutputStreamExample {


public static void main(String[] args) throws IOException {
// TODO Auto-generated method stub
byte content[] = "Jtp is the best website to learn new technologies".getBytes();
ByteArrayInputStream inputStream = new ByteArrayInputStream(content);

inputStream.read(content);

File newFile = new File("/Users/MyUser/Desktop/MyNewFile.doc");
FileOutputStream outputStream = new FileOutputStream(newFile);
outputStream.write(content);

}


}
```

Java FileInputStream Class

Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

ava FileInputStream class declaration

Let's see the declaration for java.io.FileInputStream class:

```
public class FileInputStream extends InputStream
```

Java FileInputStream class methods

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the

	input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannel getChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptor getFD()	It is used to return the FileDescriptor object.
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

Java FileOutputStream Class

Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

FileOutputStream class declaration

Let's see the declaration for Java.io.FileOutputStream class:

1. **public class** FileOutputStream **extends** OutputStream

FileOutputStream class methods

Method	**Description**
protected void finalize()	It is used to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.

void close()	It is used to closes the file output stream.

Java DataInputStream Class

Java DataInputStream class allows an application to read primitive data from the input stream in a machine-independent way.

Java application generally uses the data output stream to write data that can later be read by a data input stream.

Java DataInputStream class declaration

Let's see the declaration for java.io.DataInputStream class:

public class DataInputStream **extends** FilterInputStream **implements** DataInput

Java DataInputStream class Methods

Method	Description
int read(byte[] b)	It is used to read the number of bytes from the input stream.
int read(byte[] b, int off, int len)	It is used to read **len** bytes of data from the input stream.
int readInt()	It is used to read input bytes and return an int value.
byte readByte()	It is used to read and return the one input byte.
char readChar()	It is used to read two input bytes and returns a char value.
double readDouble()	It is used to read eight input bytes and returns a double value.
boolean readBoolean()	It is used to read one input byte and return true if byte is non zero, false if byte is zero.
int skipBytes(int x)	It is used to skip over x bytes of data from the input stream.
String readUTF()	It is used to read a string that has been encoded using the UTF-8 format.
void readFully(byte[] b)	It is used to read bytes from the input stream and store them into the buffer array.
void readFully(byte[] b, int off, int len)	It is used to read **len** bytes from the input stream.

Java FileOutputStream Class

Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

FileOutputStream class declaration

Let's see the declaration for Java.io.FileOutputStream class:

1. **public class** FileOutputStream **extends** OutputStream

FileOutputStream class methods

Method	**Description**
protected void finalize()	It is used to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.
void close()	It is used to closes the file output stream.

Java ObjectInputStream class

The objectinputstream class is mainly used to deserialize the primitive data and objects which are written by using ObjectOutputStream. ObjectInputStream can also be used to pass the objects between hosts by using a SocketStream. The objects which implement Serializable or Externalizable interface can only be read using ObjectInputStream. Consider the following line of code to read an object from the file which was previously written by using ObjectOutputStream.

```
FileInputStream fis = new FileInputStream("t.tmp");
   ObjectInputStream ois = new ObjectInputStream(fis);
   int i = ois.readInt();
         String today = (String) ois.readObject();
   Date date = (Date) ois.readObject();
         ois.close();
```

Java ObjectInputStream class declaration

1. **public class** ObjectInputStream **extends** InputStream **implements** ObjectInput, ObjectStream Constants

List of ObjectInputStream Methods

NO	**Method**	**Description**
1	public int available() throws IOException	The available() method returns the number of bytes that can be read without blocking.

2	public void close() throws IOException	The close() method closes the input stream. It must be called to release any resource associated with the stream.
3	public void defaultReadObject() throws IOException, ClassNotFoundException	The default ReadObject() method reads the non-static and non-transient fields of the current class from this stream. i.e one cannot use defaultReadObject() to read the static fields. This may only be called from the readObject method of the class being desterilized. You cannot use defaultReadObject() directly.
4	public int read() throws IOException	The read() method of java.io.ObjectInputStream reads a byte of data. This method will block if no input is available. i.e., some data should be present to read in inputstream otherwise method will block. The actual number of bytes read or -1 is returned when the end of the stream is reached.
5	public int read(byte[] buf, int off, int len) throws IOException	The read() method read into an array of bytes. There should be some bytes present to be read otherwise the method will block. This method takes 3 parameters, a byte array buff into which the bytes are stored, starting offset off to start reading, Length of bytes len.
6	public boolean readBoolean() throws IOException	This method reads a Boolean from the stream. It returns true if the byte is non-zero otherwise false
7	public byte readByte() throws IOException	The readByte() method reads an 8-bit byte.
8	public char readChar() throws IOException	The readChar() method reads a 16-bit char.
9	Public double readDouble() throws IOException	
10	public float readFloat() throws IOException	The readFloat() method reads a 32-bit float.
11	Public int readInt() throws IOException	The readInt() method reads a 32-bit int.
12	public long readLong() throws IOException	The readLong() method reads a 64-bit float.
13	Public short readShort() throws IOException	The readShort() method read 16 bit short.

14	public String readLine() throws IOException	The readLine() method reads in a line that has been terminated by a \n, \r, \r\n or EOF.
15	public void readFully(byte[] buf) throws IOException public void readFully(byte[] buf, int off, int len) throws IOException	The readFully() method of ObjectInputStream class Reads bytes until all the bytes in objectinputstream are read. This method is overloaded as It takes 3 parameters, the buffer into which data is stored, from where to start reading, and how many bytes to read.
16	public ObjectInputStream.GetField readFields() throws IOException, ClassNotFoundException	The readField() method of ObjectInputStream class reads the persistent fields from the stream and makes them available by name.
17	public final Object readObject() throws IOException, ClassNotFoundException	The readObject() method of ObjectInputStream class is used to read an object from objectinputstresm.
18	public String readUTF() throws IOException	The readUTF() method of ObjectInputStream class reads a String in modified UTF-8 format. It returns String.
19	public int readUnsignedShort() throws IOException	The readUnsignedShort() method of ObjectInputStream class reads an unsigned 16 bit short.
20	public int readUnsignedByte() throws IOException	The readUnsignedByte() method of ObjectInputStream class reads an unsigned 8-bit byte.
21	public int skipBytes(int len) throws IOException	The skipBytes() method of ObjectInputStream class skips bytes by the number passed as a parameter.
22	protected Object readObjectOverride() throws IOException, ClassNotFoundException	The readObjectOverride() method of ObjectInputStream class is called by the trusted subclasses of ObjectOutputStream that constructed ObjectOutputStream using the protected no-arg constructor. The subclass is expected to provide an override method with the modifier "final."
23	public Object readUnshared() throws IOException, ClassNotFoundException	The readUnshared() method of ObjectInputStream class reads an "unshared" object from the ObjectInputStream.
24	public	The registerValidation() method of

	void registerValidation(ObjectInputValidation obj, int prio) throws NotActiveException, InvalidObjectException	ObjectInputStream class registers an object to be validated before the graph is returned.
25	protected boolean enableResolveObject(boolean enable) throws SecurityException	The enableResolveObject () method of ObjectInputStream enables the stream to do replacement of objects read from the stream. When enabled, the resolveObject(java.lang.Object) method is called for every object being deserialized.
26	protected Class<?> resolveClass(ObjectStreamClass desc) throws IOException, ClassNotFoundException	The resolveClass() method loads the local class equivalent of the specified stream class description. Subclasses may implement this method to allow classes to be fetched from an alternate source.
27	protected void readStreamHeader() throws IOException, StreamCorruptedException	The readStreamHeader() method of ObjectInputStream is provided to allow subclasses to read and verify their stream headers. It reads and verifies the magic number and version number.

Working of ObjectOutputStream

Basically, the `ObjectOutputStream` encodes Java objects using the class name and object values. And, hence generates corresponding streams. This process is known as serialization.

Those converted streams can be stored in files and can be transferred among networks.

Note: The ObjectOutputStream class only writes those objects that implement the Serializable interface. This is because objects need to be serialized while writing to the stream

---

Create an ObjectOutputStream

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

```
// Creates a FileOutputStream where objects from ObjectOutputStream are written
FileOutputStream fileStream = new FileOutputStream(String file);

// Creates the ObjectOutputStream
ObjectOutputStream objStream = new ObjectOutputStream(fileStream);
```

In the above example, we have created an object output stream named `objStream` that is linked with the file output stream named `fileStream`.

Methods of ObjectOutputStream

The `ObjectOutputStream` class provides implementations for different methods present in the `OutputStream` class.

write() Method

- `write()` - writes a byte of data to the output stream
- `writeBoolean()` - writes data in boolean form
- `writeChar()` - writes data in character form
- `writeInt()` - writes data in integer form
- `writeObject()` - writes object to the output stream

Java - RandomAccessFile

This class is used for reading and writing to random access file. A random access file behaves like a large array of bytes. There is a cursor implied to the array called file pointer, by moving the cursor we do the read write operations. If end-of-file is reached before the desired number of byte has been read than EOFException is thrown. It is a type of IOException.

Constructor

Constructor	**Description**
RandomAccessFile(File file, String mode)	Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument.
RandomAccessFile(String name, String mode)	Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Method

Modifier and Type	**Method**	**Method**
void	close()	It closes this random access file stream and releases any system resources associated with the stream.
FileChannel	getChannel()	It returns the unique FileChannel object associated with this file.
Int	readInt()	It reads a signed 32-bit integer from this file.

String	readUTF()	It reads in a string from this file.
void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
void	writeDouble(double v)	It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first.
void	writeFloat(float v)	It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first.
void	write(int b)	It writes the specified byte to this file.
Int	read()	It reads a byte of data from this file.
long	length()	It returns the length of this file.
void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

CharacterStream Classes in Java

The java.io package provides CharacterStream classes to overcome the limitations of ByteStream classes, which can only handle the 8-bit bytes and is not compatible to work directly with the Unicode characters. CharacterStream classes are used to work with 16-bit Unicode characters. They can perform operations on characters, char arrays and Strings.

However, the CharacterStream classes are mainly used to read characters from the source and write them to the destination. For this purpose, the CharacterStream classes are divided into two types of classes, I.e., Reader class and Writer class.

Reader Class

Reader class is used to read the 16-bit characters from the input stream. However, it is an abstract class and can't be instantiated, but there are various subclasses that inherit the Reader class and override the methods of the Reader class. All methods of the Reader class throw an IOException. The subclasses of the Reader class are given in the following table.

SN	**Class**	**Description**
1.	BufferedReader	This class provides methods to read characters from the buffer.
2.	CharArrayReader	This class provides methods to read characters from the char array.
3.	FileReader	This class provides methods to read characters from the file.
4.	FilterReader	This class provides methods to read characters from the underlying

		character input stream.
5	InputStreamReader	This class provides methods to convert bytes to characters.
6	PipedReader	This class provides methods to read characters from the connected piped output stream.
7	StringReader	This class provides methods to read characters from a string.

The Reader class methods are given in the following table.

Prime Ministers of India | List of Prime Minister of India (1947-2020)

SN	Method	Description
1	int read()	This method returns the integral representation of the next character present in the input. It returns -1 if the end of the input is encountered.
2	int read(char buffer[])	This method is used to read from the specified buffer. It returns the total number of characters successfully read. It returns -1 if the end of the input is encountered.
3	int read(char buffer[], int loc, int nChars)	This method is used to read the specified nChars from the buffer at the specified location. It returns the total number of characters successfully read.
4	void mark(int nchars)	This method is used to mark the current position in the input stream until nChars characters are read.
5	void reset()	This method is used to reset the input pointer to the previous set mark.
6	long skip(long nChars)	This method is used to skip the specified nChars characters from the input stream and returns the number of characters skipped.
7	boolean ready()	This method returns a boolean value true if the next request of input is ready. Otherwise, it returns false.
8	void close()	This method is used to close the input stream. However, if the program attempts to access the input, it generates IOException.

Writer Class

Writer class is used to write 16-bit Unicode characters to the output stream. The methods of the Writer class generate IOException. Like Reader class, Writer class is also an abstract class that cannot be instantiated; therefore, the subclasses of the Writer class are used to write the characters onto the output stream. The subclasses of the Writer class are given in the below table.

SN	Class	Description

1	BufferedWriter	This class provides methods to write characters to the buffer.
2	FileWriter	This class provides methods to write characters to the file.
3	CharArrayWriter	This class provides methods to write the characters to the character array.
4	OutpuStreamWriter	This class provides methods to convert from bytes to characters.
5	PipedWriter	This class provides methods to write the characters to the piped output stream.
6	StringWriter	This class provides methods to write the characters to the string.

To write the characters to the output stream, the Write class provides various methods given in the following table.

SN	Method	Description
1	void write()	This method is used to write the data to the output stream.
2	void write(int i)	This method is used to write a single character to the output stream.
3	Void write(char buffer[])	This method is used to write the array of characters to the output stream.
4	void write(char buffer [],int loc, int nChars)	This method is used to write the nChars characters to the character array from the specified location.
5	void close ()	This method is used to close the output stream. However, this generates the IOException if an attempt is made to write to the output stream after closing the stream.
6	void flush ()	This method is used to flush the output stream and writes the waiting buffered characters.