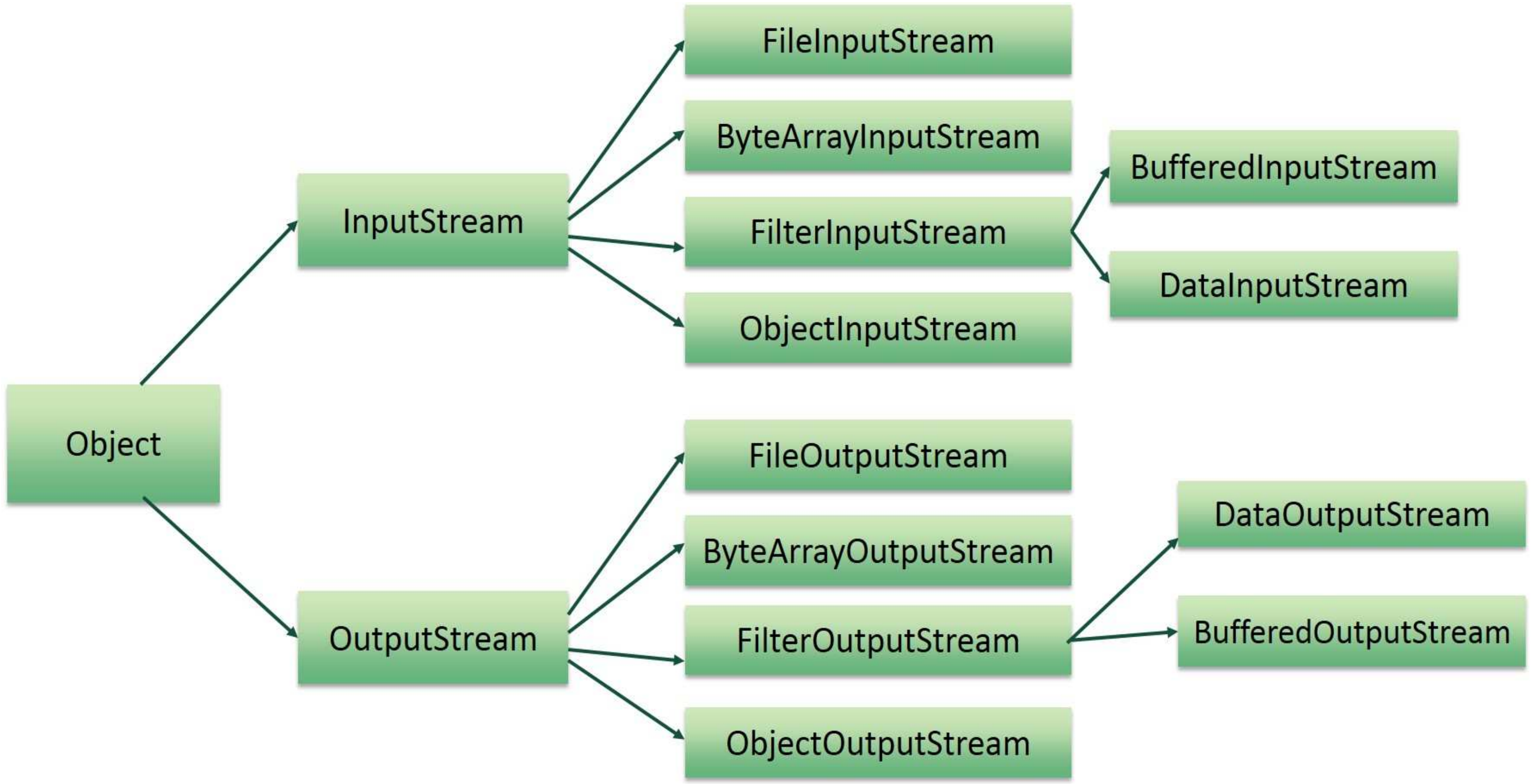
Streams and I/O

Introduction

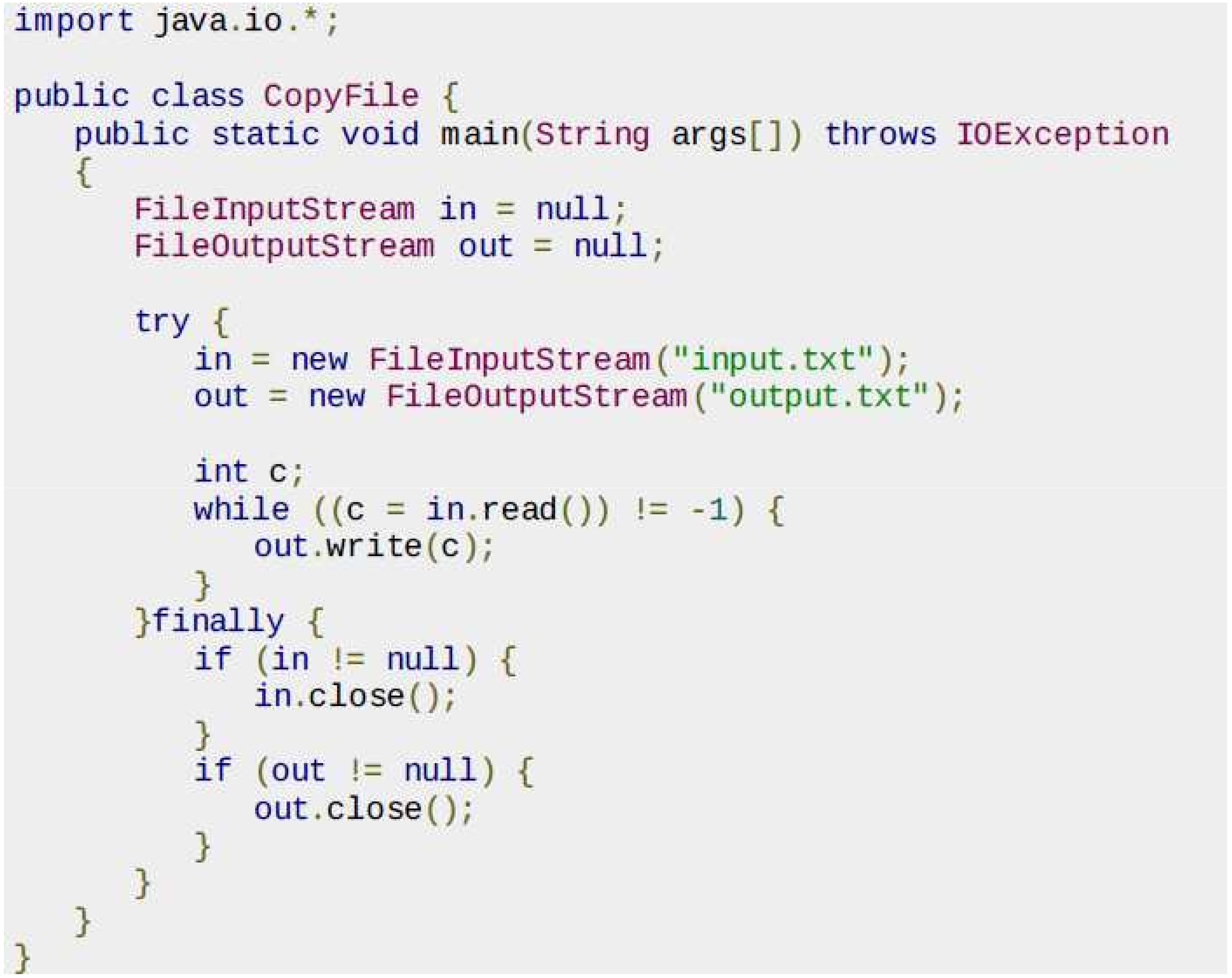
- A stream can be defined as a sequence of data.
- There are two kinds of Streams
  - InputStream: The InputStream is used to read data from a source.
  - OutputStream: The OutputStream is used for writing data to a destination.

Hierarchy of classes



Byte Streams

- Java byte streams are used to perform input and output of 8-bit bytes.
- Though there are many classes related to byte streams but the most frequently used classes are ,
  - **FileInputStream**
  - **FileOutputStream**



```java
import java.io.*;

public class CopyFile {
   public static void main(String args[]) throws IOException
   {
      FileInputStream in = null;
      FileOutputStream out = null;

      try {
         in = new FileInputStream("input.txt");
         out = new FileOutputStream("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Character Streams

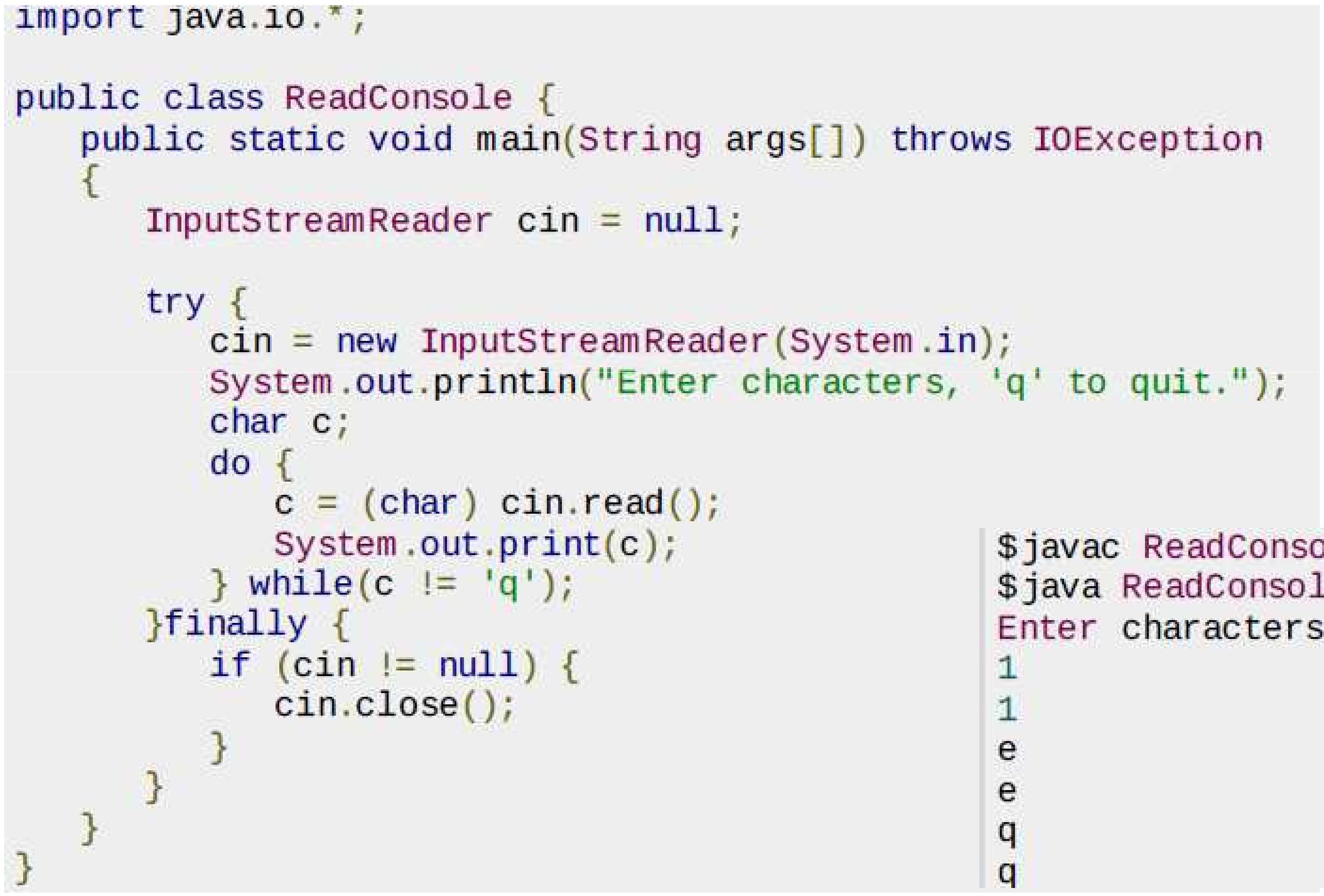
- Java **Character** streams are used to perform input and output for 16-bit unicode.
- Though there are many classes related to character streams but the most frequently used classes are
  - **FileReader**
  - **FileWriter**
- Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

```java
import java.io.*;

public class CopyFile {
   public static void main(String args[]) throws IOException
   {
      FileReader in = null;
      FileWriter out = null;

      try {
         in = new FileReader("input.txt");
         out = new FileWriter("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Standard streams

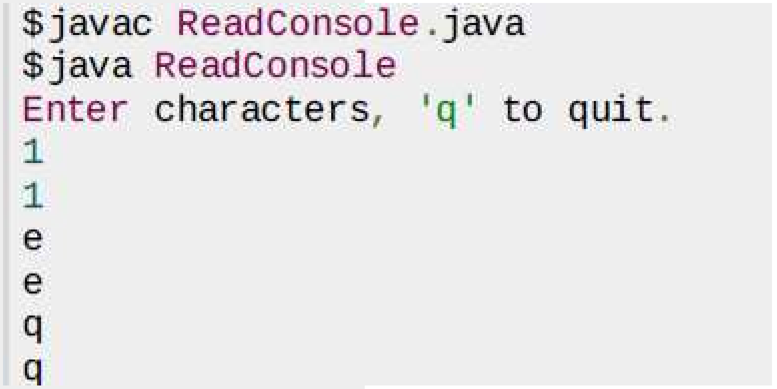
- **Standard Input:** This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as **System.in**.
- **Standard Output:** This is used to output the data produced by the user's program and usually a computer screen is used to standard output stream and represented as **System.out**.
- **Standard Error:** This is used to output the error data produced by the user's program and usually a computer screen is used to standard error stream and represented as **System.err**.

Example



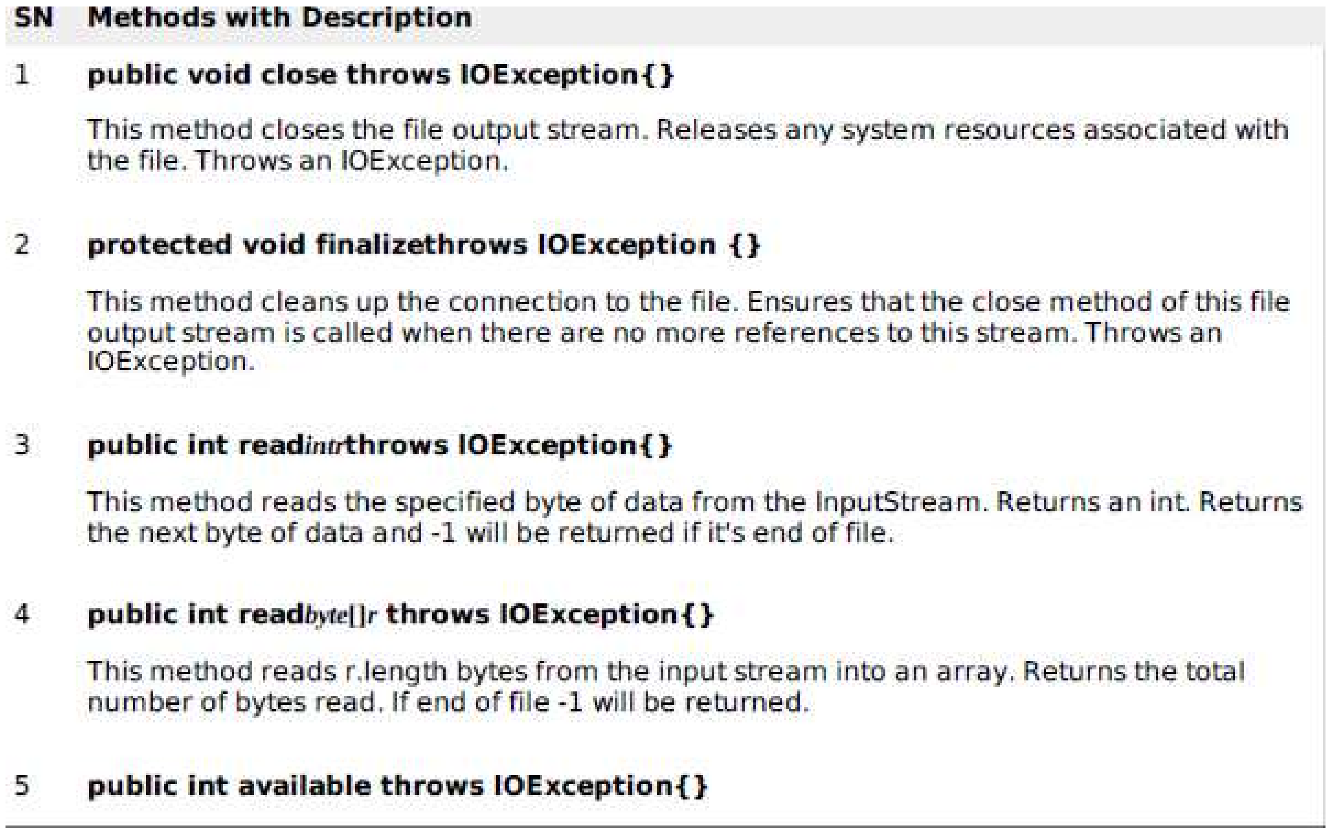
```java
import java.io.*;

public class ReadConsole {
   public static void main(String args[]) throws IOException
   {
      InputStreamReader cin = null;

      try {
         cin = new InputStreamReader(System.in);
         System.out.println("Enter characters, 'q' to quit.");
         char c;
         do {
            c = (char) cin.read();
            System.out.print(c);
         } while(c != 'q');
      }finally {
         if (cin != null) {
            cin.close();
         }
      }
   }
}
```

Output:



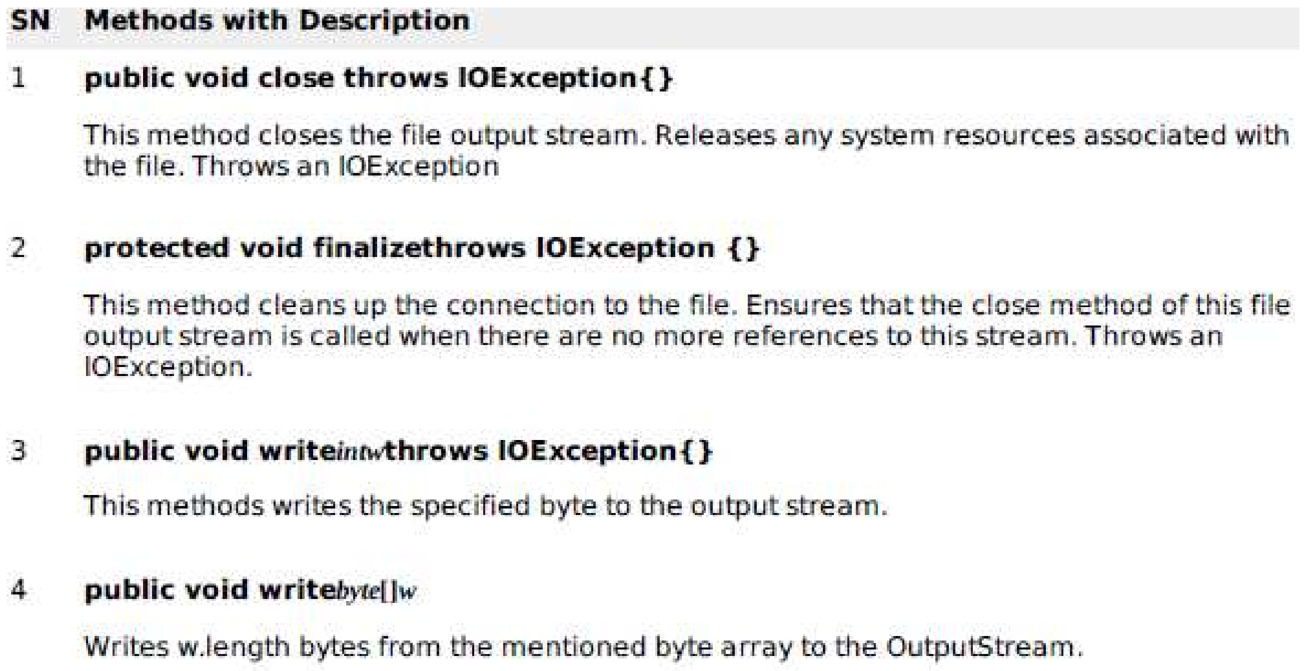
```
$javac ReadConsole.java
$java ReadConsole
Enter characters, 'q' to quit.
1
1
e
e
q
q
```

FileInputStream - methods



SN	Methods with Description
1	**public void close throws IOException{}** This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
2	**protected void finalizethrows IOException {}** This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3	**public int read*intr*throws IOException{}** This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's end of file.
4	**public int read*byte[]r* throws IOException{}** This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If end of file -1 will be returned.
5	**public int available throws IOException{}**

FileOutputStream - methods



SN	Methods with Description
1	**public void close throws IOException{}** This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException
2	**protected void finalizethrows IOException {}** This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3	**public void write***intw***throws IOException{}** This methods writes the specified byte to the output stream.
4	**public void write***byte[]w* Writes w.length bytes from the mentioned byte array to the OutputStream.