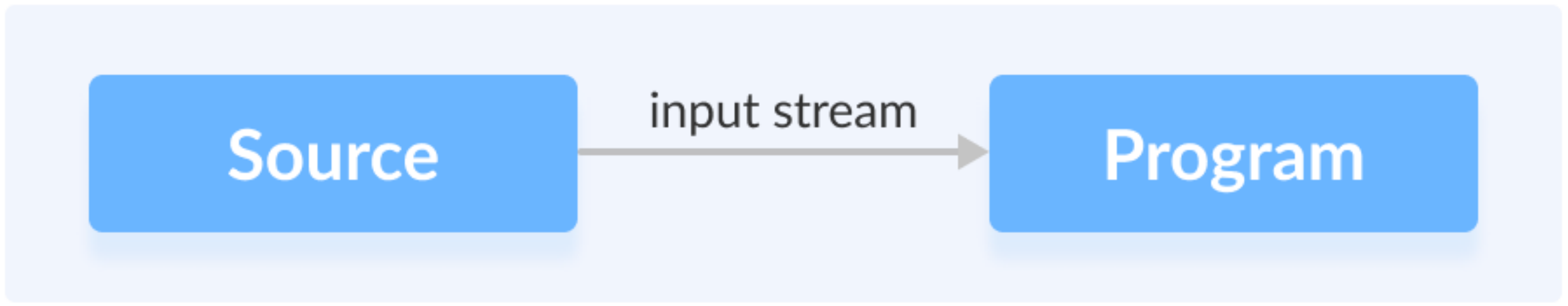
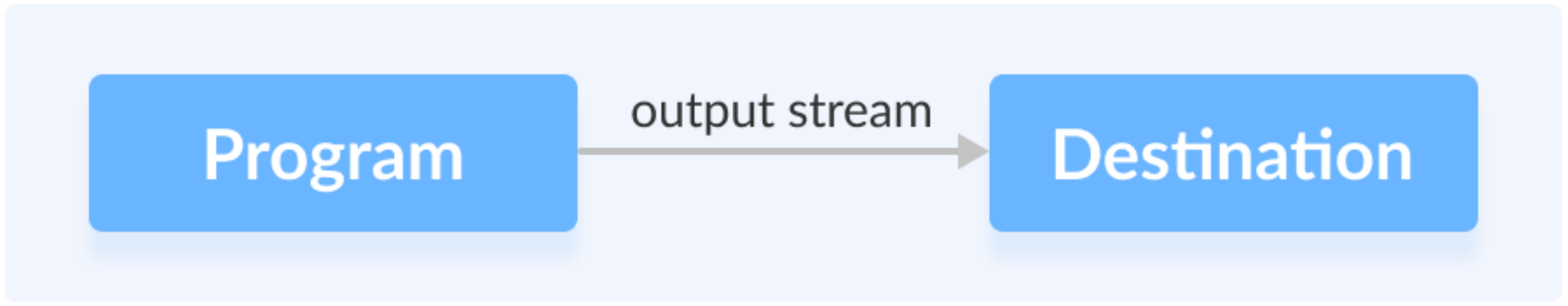
File, Stream and I/O

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.

Reading data from source



Writing data to destination



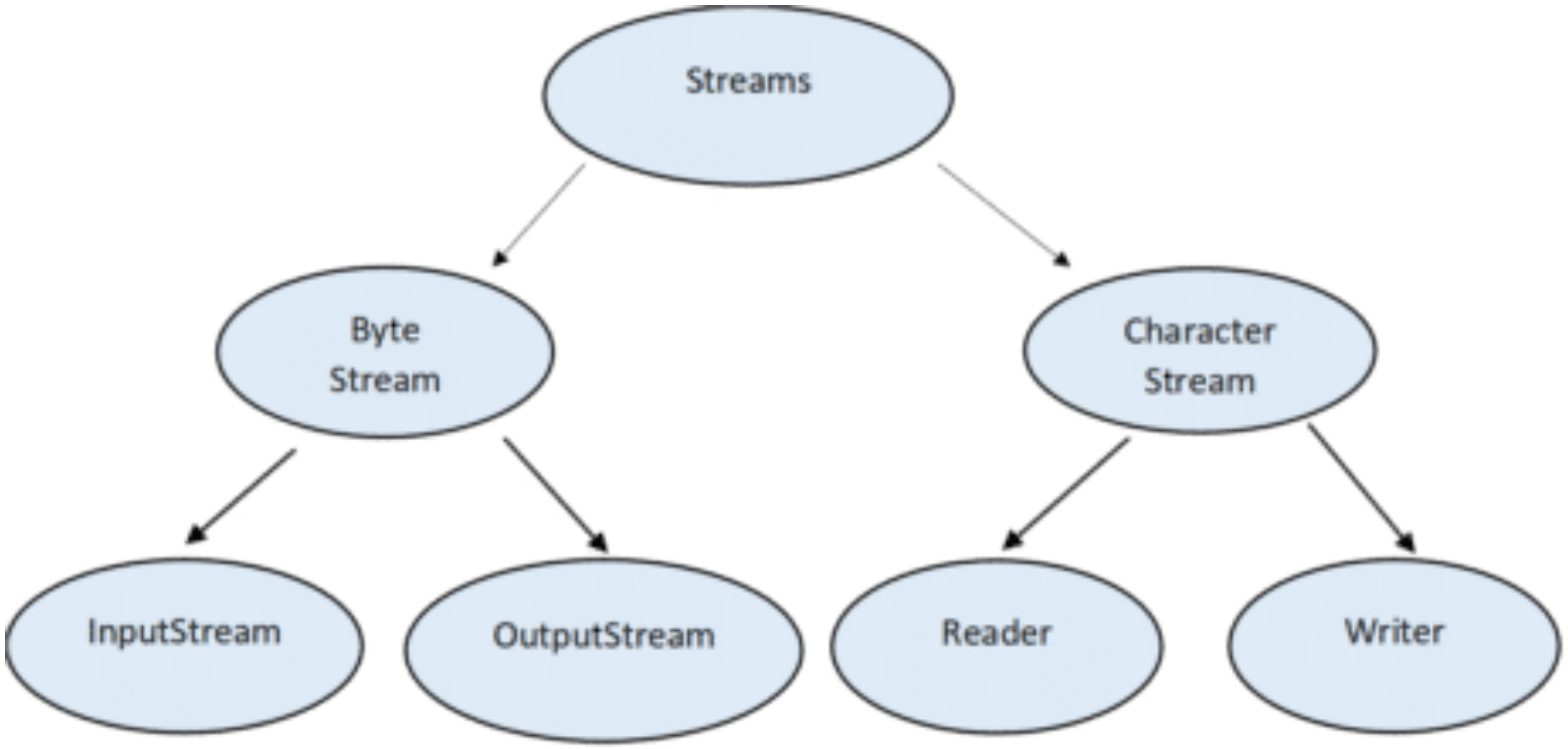
Streams are the sequence of bits(data).
There are two types of streams:

- Input Streams
- Output Streams

Input Streams:
Input streams are used to read the data from various input devices like keyboard, file, network, etc.
Output Streams:
Output streams are used to write the data to various output devices like monitor, file, network, etc.

There are two types of streams based on data:

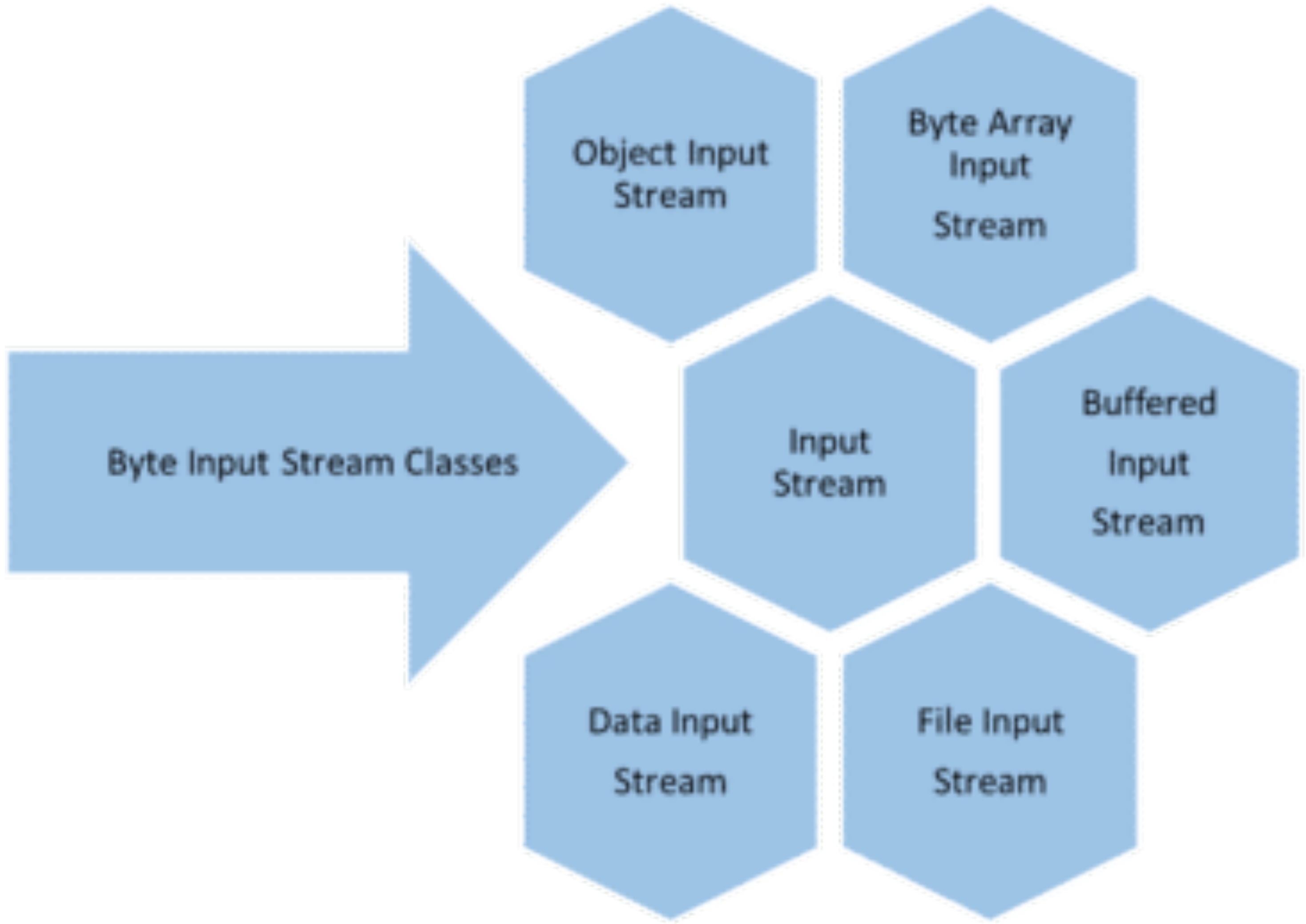
- **Byte Stream**: used to read or write byte data.
- **Character Stream**: used to read or write character data.



-

Byte Input Stream:

- These are used to read byte data from various input devices.
- InputStream is an abstract class and it is the super class of all the input byte streams.



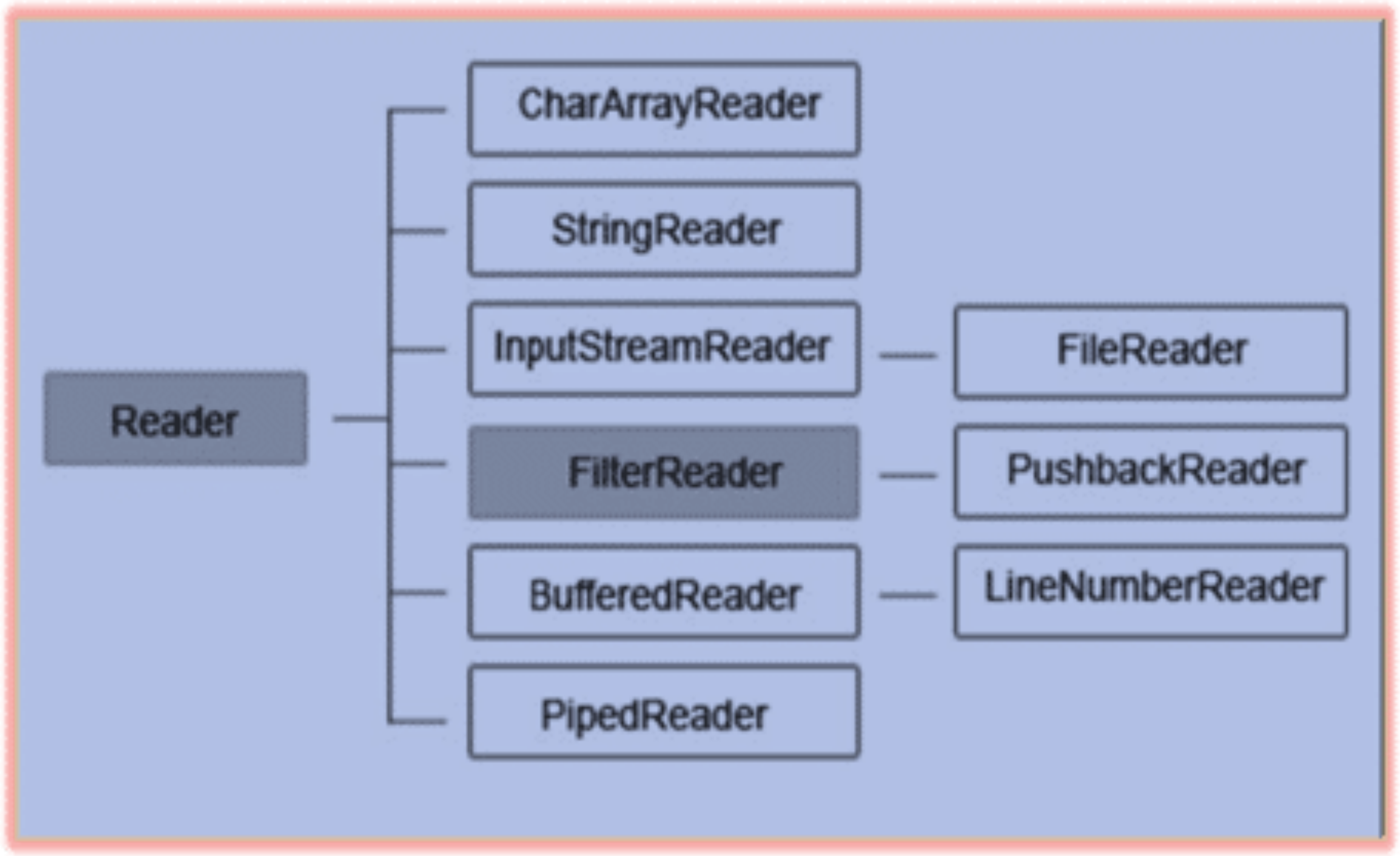
-

Byte Streams

```
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();
         }
      }
      }
      }
```

Character Input Stream:

- These are used to read char data from various input devices.
- Reader is an abstract class and is the super class for all the character input streams.



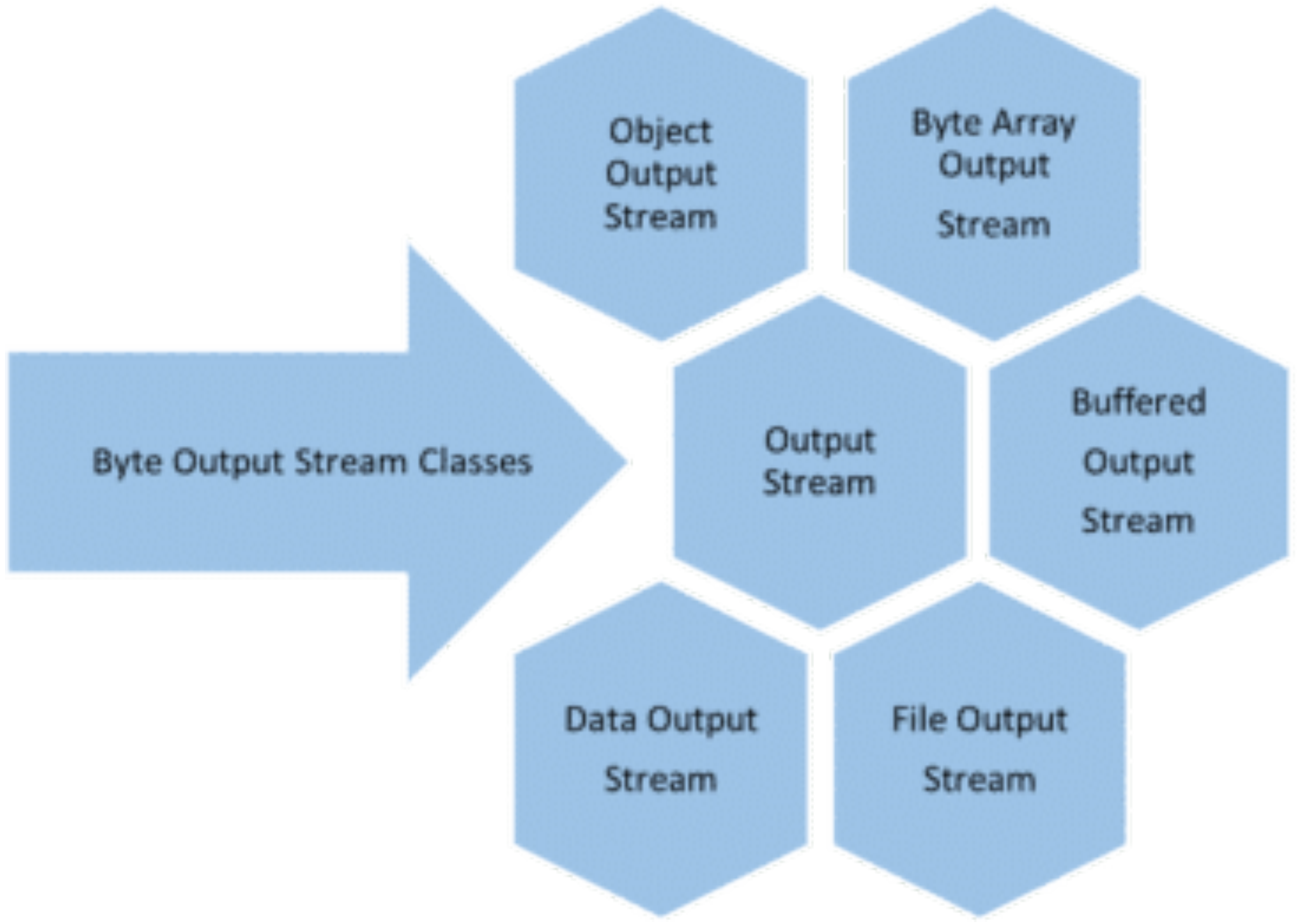
-

Character Stream

```
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("Press Enter to quit.");
         char c;
         do {
            c = (char) cin.read();
            System.out.print(c);
         }
         while(c != ‘\n');
      }
      finally {
         if (cin != null) {
         cin.close();

         }
      }
   }
}
```

Byte Output Stream:

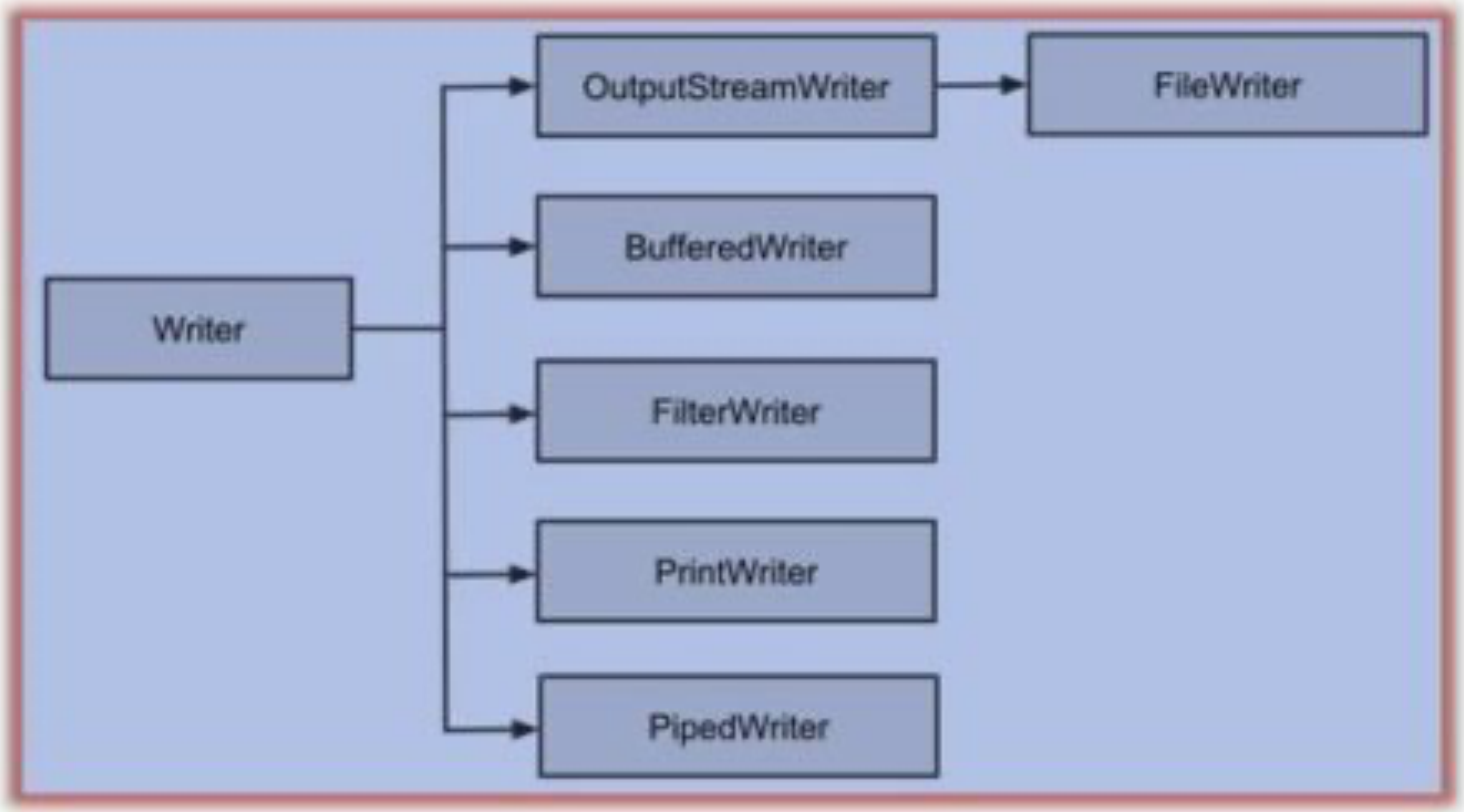
- These are used to write byte data to various output devices.
- Output Stream is an abstract class and it is the superclass for all the output byte streams.



-

Character Output Stream:

- These are used to write char data to various output devices.
- Writer is an abstract class and is the super class of all the character output streams.



-

File Navigation and I/O:

File Class:

This class is used for creation of files and directories, file searching, file deletion etc.

SN	Methods with Description
1	**public String getName()** Returns the name of the file or directory denoted by this abstract pathname.
2	**public String getParent()** Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
3	**public File getParentFile()** Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.
4	**public String getPath()** Converts this abstract pathname into a pathname string.

```java
import java.io.File;

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

   File f = null;
   String[] strs = {"test1.txt", "test2.txt"};
   try{
    // for each string in string array
    for(String s:strs )
    {
     // create new file
     f= new File(s);

     // true if the file is executable
     boolean bool = f.canExecute();

     // find the absolute path
     String a = f.getAbsolutePath();

     // prints absolute path
     System.out.print(a);

     // prints
     System.out.println(" is executable: "+ bool);
    }
   }catch(Exception e){
    // if any I/O error occurs
    e.printStackTrace();
   }
```

This class inherits from the InputStreamReader class. FileReader is used for reading streams of characters.

```
import java.io.*;

public class FileRead{

 public static void main(String args[])throws IOException{

   File file = new File("Hello1.txt");
   // creates the file
   file.createNewFile();

   // creates a FileWriter Object
   FileWriter writer = new FileWriter(file);

   // Writes the content to the file
   writer.write("This\n is\n an\n example\n");
   writer.flush();
   writer.close();

   //Creates a FileReader Object
   FileReader fr = new FileReader(file);
   char [] a = new char[50];
   fr.read(a); // reads the content to the array
   for(char c : a)
     System.out.print(c); //prints the characters one by one
     fr.close();
 }
}
```

FileWriterClass:

This class inherits from the OutputStreamWriter class. The class is used for writing streams of characters.

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

    File file = new File(“outputstream.txt");
    // creates the file
    file.createNewFile();

    // creates a FileWriter Object
    FileWriter writer = new FileWriter(file);
    // Writes the content to the file
    writer.write("This\n is\n an\n example\n");
    writer.flush();
    writer.close();

    //Creates a FileReader Object
    FileReader fr = new FileReader(file);
    char [] a = new char[50];
    fr.read(a); // reads the content to the array
    for(char c : a)
      System.out.print(c); //prints the characters one by one
      fr.close();
  }
}
```

Directories :

Creating Directories:

- **The mkdir( ) method creates a directory, returning true on success and false on failure.**
- **The mkdirs() method creates both a directory and all the parents of the directory.**

```
import java.io.File;

public class CreateDir {
  public static void main(String args[]) {
  String dirname = "/tmp/user/java/bin";
  File d = new File(dirname);
    // Create directory now.
    d.mkdirs();
  }
}
```

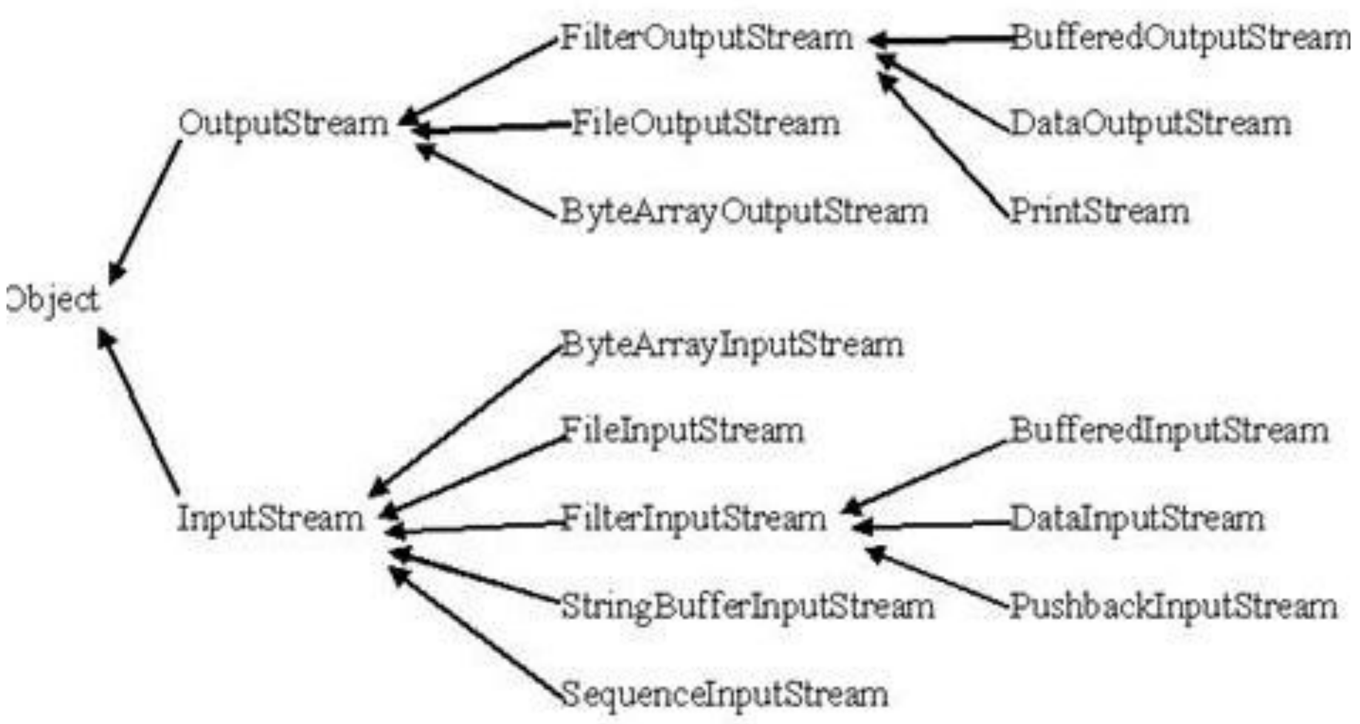
Listing Directories:

```
   File file = null; String[] paths;
   try{
     // create new file object
     file = new File("/tmp");
     // array of files and directory
     paths = file.list();
     // for each name in the path array
     for(String path:paths)
     {
       // prints filename and directory name
       System.out.println(path);
     }
   }catch(Exception e){
     // if any error occurs
     e.printStackTrace();
   }
  }
}
```

Standard Stream:

- 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

Reading and writing Files



FileInputStream

```
File f = new File("C:/java/hello");
InputStream f = new FileInputStream(f);
```

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 finalize()throws 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(int r)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{}** Gives the number of bytes that can be read from this file input stream. Returns an int.

ByteArrayInputStream

```
import java.io.*;

public class ByteStreamTest {
  public static void main(String args[])throws IOException {
  ByteArrayOutputStream bOutput = new ByteArrayOutputStream(12);
  while( bOutput.size()!= 10 ) {
     // Gets the inputs from the user
     bOutput.write(System.in.read());
    }

    byte b [] = bOutput.toByteArray();
    System.out.println("Print the content");
    for(int x= 0 ; x < b.length; x++) {
     // printing the characters
     System.out.print((char)b[x] + "   ");
    }
    System.out.println(" "); int c;

    ByteArrayInputStream bInput = new ByteArrayInputStream(b);

    System.out.println("Converting characters to Upper case " );
    for(int y = 0 ; y < 1; y++ ) {
     while(( c= bInput.read())!= -1)
      { System.out.println(Character.toUpperCase((char)c));
     }
     bInput.reset();
    }
   }
 }
```

DataInputStream

The DataInputStream is used in the context of DataOutputStream and can be used to read primitives.

```
import java.io.*;

public class Test{
 public static void main(String args[])throws IOException{

   DataInputStream d = new DataInputStream(new
               FileInputStream("test.txt"));

   DataOutputStream out = new DataOutputStream(new
               FileOutputStream("test1.txt"));

   String count;
   while((count = d.readLine()) != null){
   String u = count.toUpperCase();
   System.out.println(u); out.writeBytes(u + "  ,");
   }
   d.close();
   out.close();
  }
}
```

FileOutputStream:

FileOutputStream is used to create a file and write data into it.

```
File f = new File("C:/java/hello");
OutputStream f = new FileOutputStream(f);
```

`ByteArrayOutputStream` – The ByteArrayOutputStream class stream creates a buffer in memory and all the data sent to the stream is stored in the buffer.

```
import java.io.*;

public class ByteStreamTest {
 public static void main(String args[])throws IOException {
 ByteArrayOutputStream bOutput = new ByteArrayOutputStream(12);
 while( bOutput.size()!= 10 ) {
    // Gets the inputs from the user
    bOutput.write(System.in.read());
   }

   byte b [] = bOutput.toByteArray();
   System.out.println("Print the content");
   for(int x= 0 ; x < b.length; x++) {
    //printing the characters
    System.out.print((char)b[x]  + "   ");
   }
   System.out.println(" "); int c;

   ByteArrayOutputStream bInput = new ByteArrayOutputStream(b);

   System.out.println("Converting characters to Upper case " );
   for(int y = 0 ; y < 1; y++ ) {
    while(( c= bInput.read())!= -1) {
    System.out.println(Character.toUpperCase((char)c));
    }
    bInput.reset();
   }
```

DataOutputStream:

The DataOutputStream stream let you write the primitives to an output source.

```
import java.io.*;

public class Test{
 public static void main(String args[])throws IOException{

   DataInputStream d = new DataInputStream(new
                FileInputStream("test.txt"));

   DataOutputStream out = new DataOutputStream(new
                FileOutputStream("test1.txt"));

   String count;
   while((count = d.readLine()) != null){
   String u = count.toUpperCase();
   System.out.println(u);
   out.writeBytes(u + " ,");
   }
   d.close();
   out.close();
  }
}
```