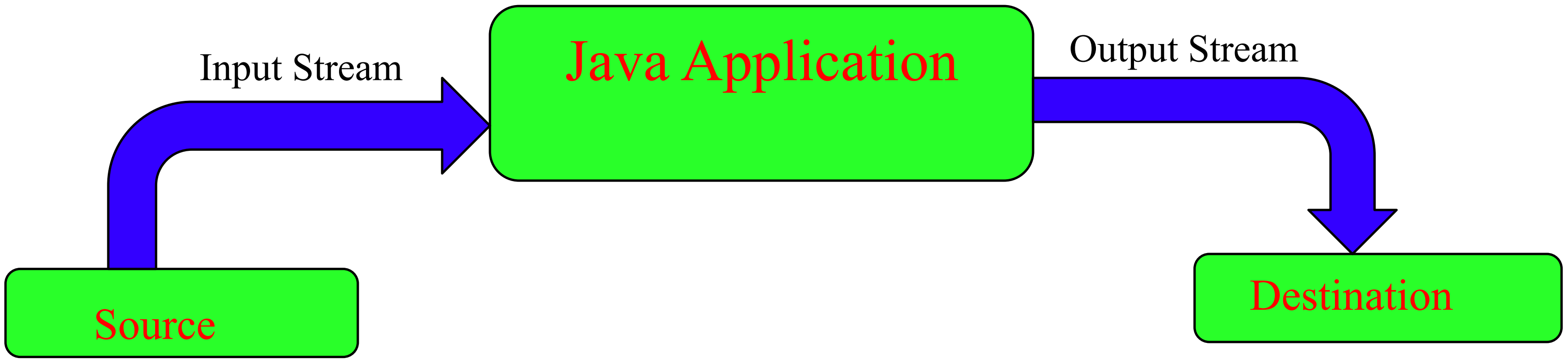
Java IO Streams

Streams

- ***Stream***: an object that either delivers data to its destination (screen, file, etc.) or that takes data from a source (keyboard, file, etc.)
    - it acts as a buffer between the data source and destination
- ***Input stream***: a stream that provides input to a program
    - `System.in` is an input stream
- ***Output stream***: a stream that accepts output from a program
    - `System.out` is an output stream
- A stream connects a program to an I/O object
    - `System.out` connects a program to the screen
    - `System.in` connects a program to the keyboard

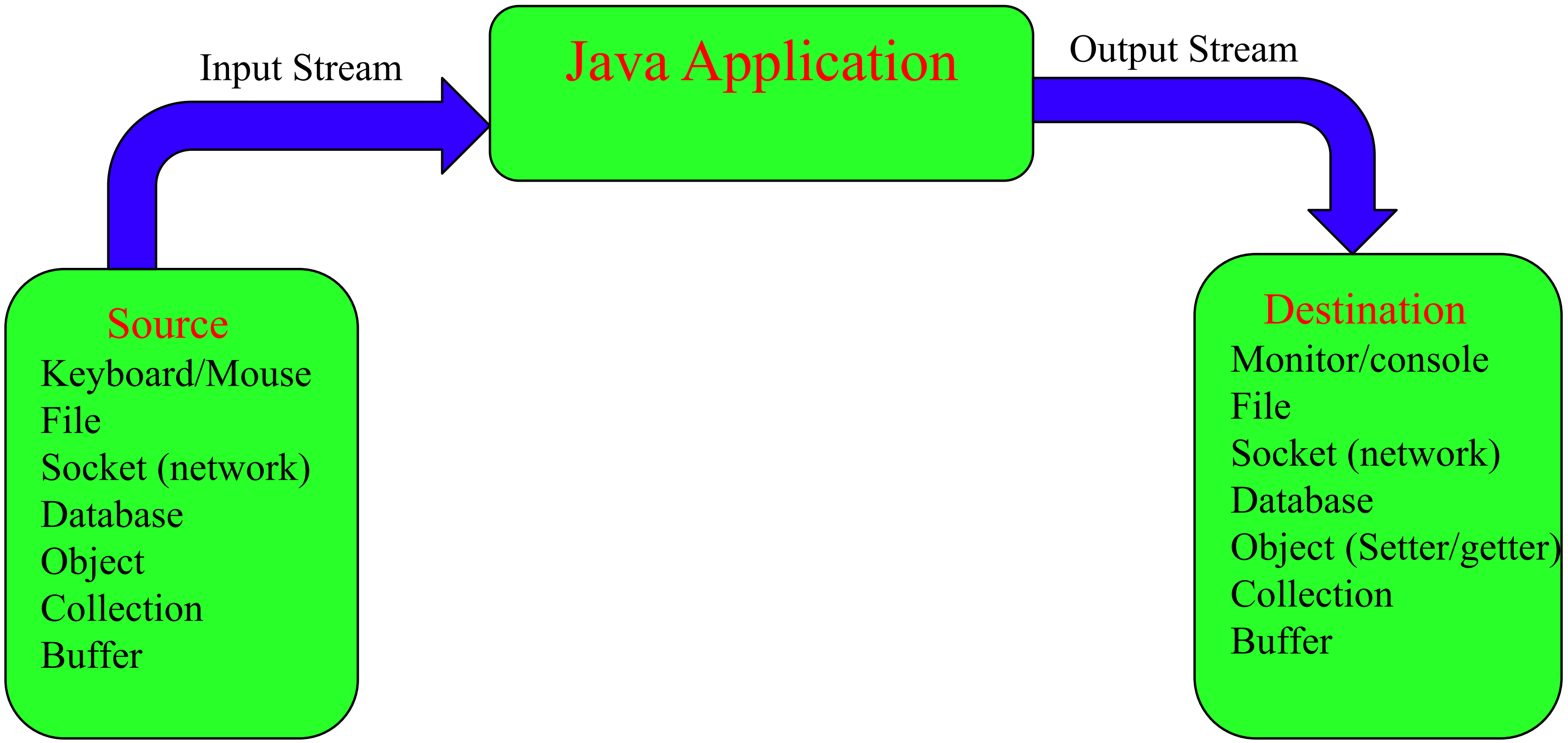
IO Streams



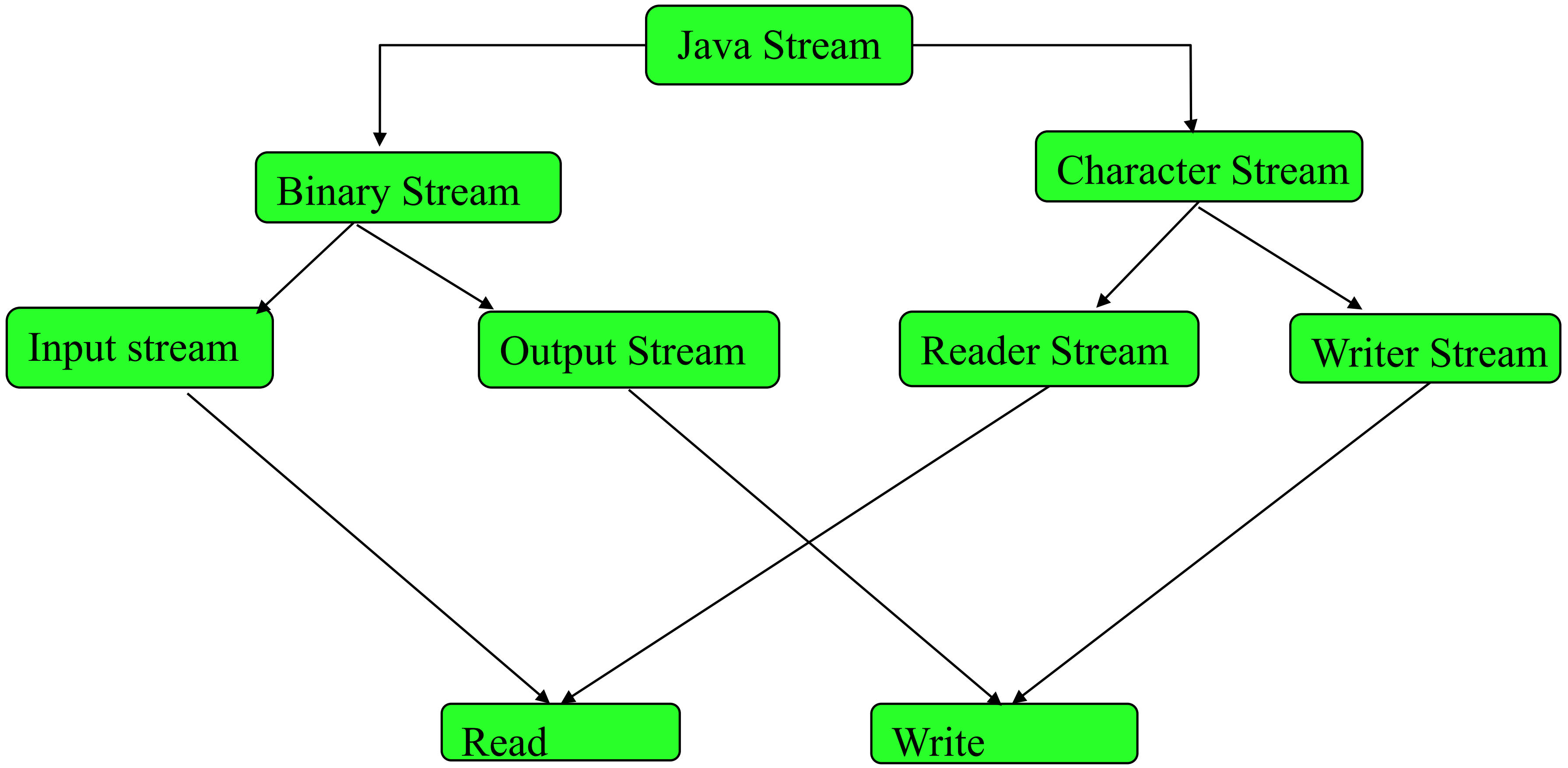
The input stream will be different based on the source

The output stream will be different based on the destination

IO Streams


Input Stream
Java Application
Output Stream
Source
Keyboard/Mouse
File
Socket (network)
Database
Object
Collection
Buffer
Destination
Monitor/console
File
Socket (network)
Database
Object (Setter/getter)
Collection
Buffer

Type of steams



Byte-Oriented Stream Classes

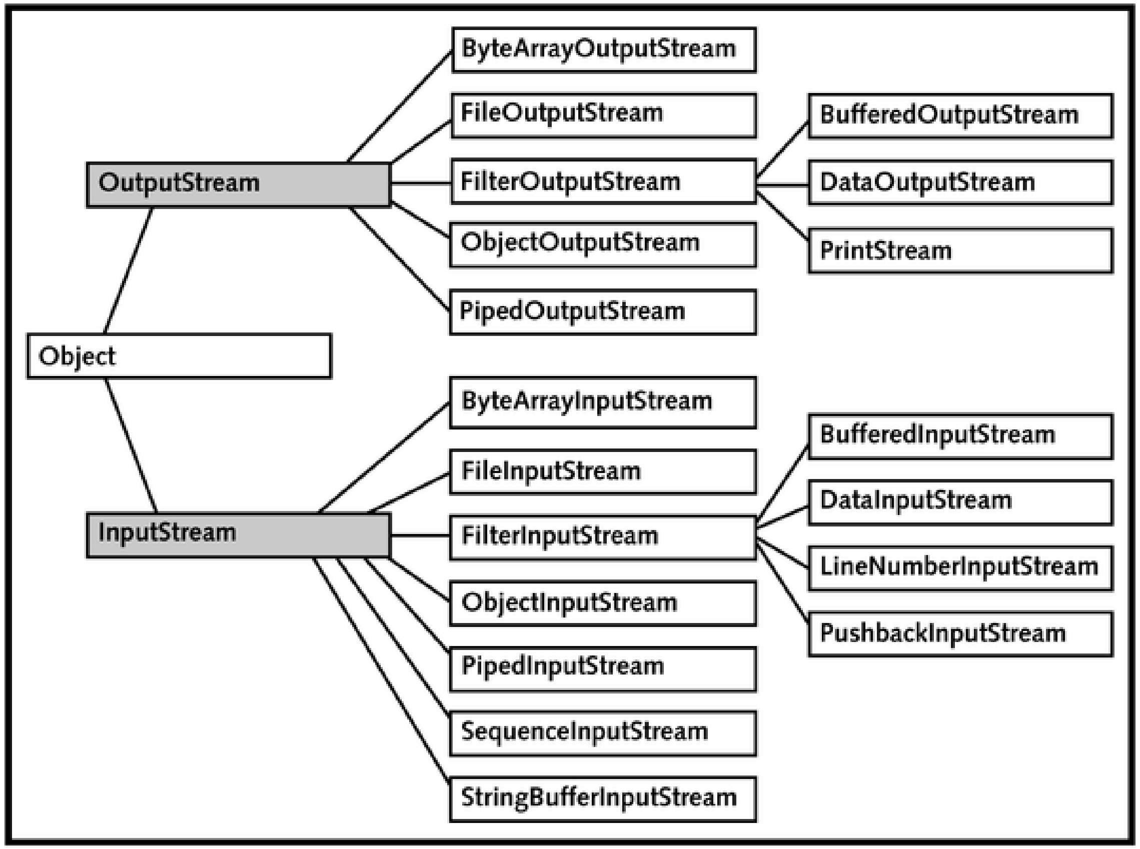


Figure 8-1 Byte-oriented stream classes

Character Stream Classes

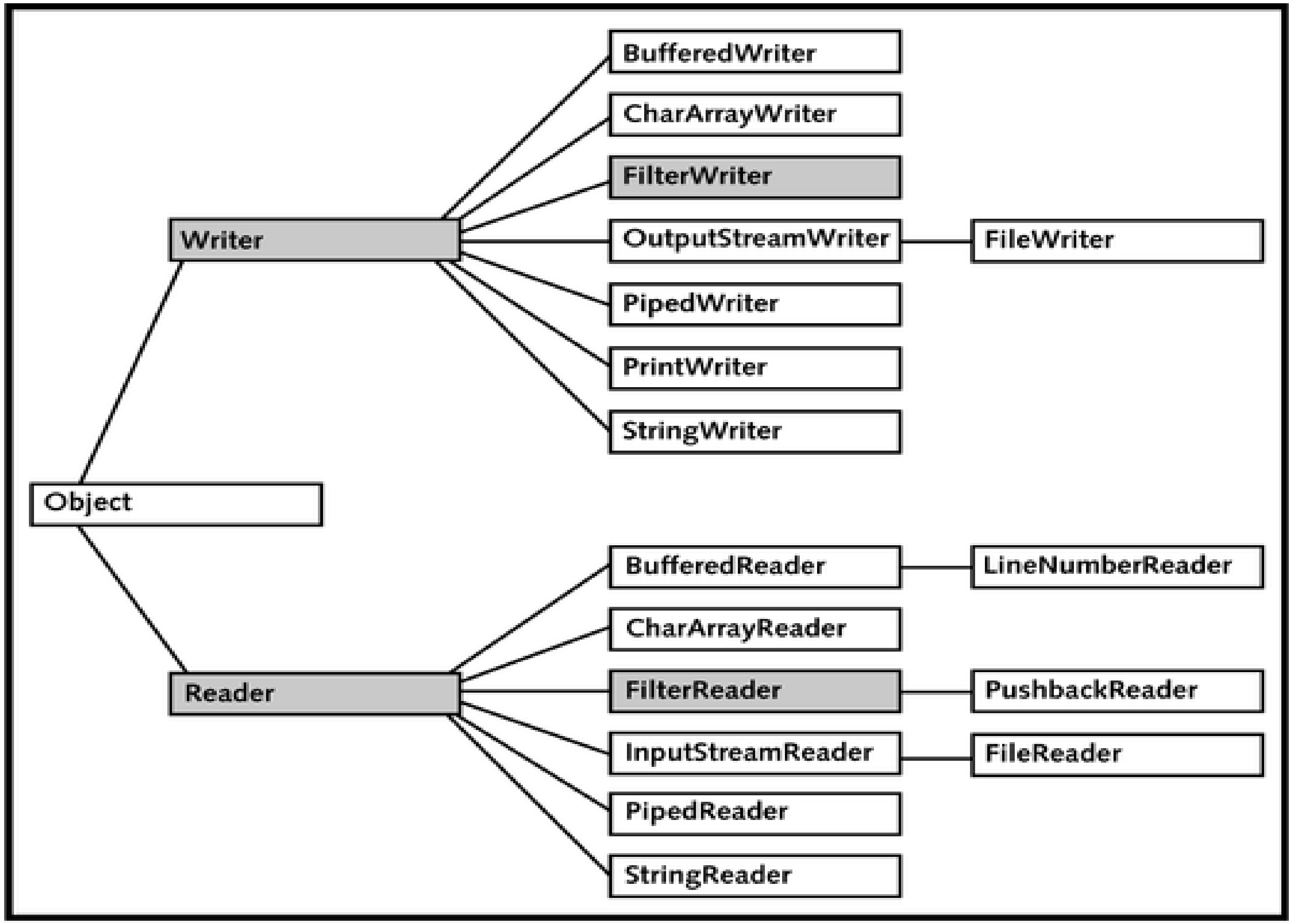


Figure 8-2 Character stream classes

File Class

- A `File` object can refer to either a file or a directory

```
File file1 = new File("data.txt");
File file1 = new File("C:\java");
```

  - To obtain the path to the current working directory use

```
System.getProperty("user.dir");
```

File Class

- Write code to create file with name demo.txt in the current working directory?
  - File f = new file (“demo.txt”)
  - F.createnewfile();

- Write code to create directory call “CECS277” in the current working directory and create file named with abc.txt in that directory.
    - File f = new File (“CECS277”);
    - f.makedir();
    - File f1 = ne File (“CECS277”, “abc.txt”);

  Or

    - File f1 = new File (f, “abc.txt”);
    - f1.createNewfile();

File Class

```
import java.io.*;

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

    File F = new File(“Test.txt”);
    F.createNewFile();
    File Dir = new File(“c:\\Test”);
    Dir.mkdirs();
    File F2 = new File(“c:\\Test”,“Test.txt”);
    F.createNewFile();

    }
  }
}
```

Useful File Methods

- createNewFile
  - To create a new file
- mkdir
  - Creates a new subdirectory
- isFile / isDirectory
- canRead / canWrite
- length
  - Length of the file in bytes (long) or 0 if nonexistent
- list
  - If the File object is a directory, returns a String array of all the files and directories contained in the directory; otherwise, null
- delete
  - Deletes the directory and returns true if successful
- toURL
  - Converts the file path to a URL object
- exists

File Class

Is the file going to be created?

```
import java.io.*;

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

    File dir = new File(System.getProperty("user.dir"));
     //File dir = new File("C:\\");
    if(dir.isDirectory()){
      System.out.println("Directory of " + dir);
      String[] listing = dir.list();
      for(int i=0; i<listing.length; i++) {
        System.out.println("\t" + listing[i]);
      }
    }
  }
}
```

How would you print
the number of files in folder?

File Writer

- Constructors:
  - FileWriter fw = new FileWriter(String name);
  - FileWriter fw = new FileWriter(File f);

  The above 2 constructors meant for overriding existing data.

- Instead of overriding, if you want to perform append operation then we have to use the following 2 constructers.
  - FileWriter fw = new FileWriter(String name, boolean append);
  - FileWriter fw = new FileWriter(File f, boolean append);

Note: if the specified file is not already available then all the above constructors will create the files

File Writer

- Methods of FileWriter class:
  - Writer (int ch); To write a single character to the file.
  - Write (char[] ch); To write an array of characters to the file.
  - Write (String s); To write a string to the file.
  - Flush(); to push the date to the file.
  - Close(); to close the file.

FileWriter

```
import java.io.*;

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


    try {
      FileWriter out = new FileWriter("book.txt");
      //FileWriter out = new FileWriter("book.txt“, true);
      System.out.println("Encoding: " + out.getEncoding());
      out.write("Core Java Programming");
      out.write("\n");
      out.flush();
      out.close();

    } catch(IOException ioe) {
      System.out.println("IO problem: " + ioe);
      ioe.printStackTrace();

      }
    }
  }
}
```

File Reader

- We can Use FileReader to read character data from the file.
- Constructor:
  - 1- FileReader fr = new FileReader (String fname);
  - 2- FileReader fr = new FileReader(File f);
- Methods:
  - in read();
    - It attempt to read next character from the file and returns it's Unicode value.
    - If there is no next character then we will get -1.
    - As this method returns Unicode value compulsory at the time of printing we should perform type –casting.

```
FileReader fr = new FileReader ("abc.txt");
int i = fr.read ();
While (i != -1){
            System.out.print((char) i);
            i = fr.read();
}
fr.close();
```

File Reader

- int read (char[] ch);

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

                    File f = new File ("Test.txt");
                    char[] ch = new char [(int) f.length()];
                    FileReader fr = new FileReader("Test.txt");
                    fr.read(ch);
                    for (char ch1:ch){
                              System.out.println(ch1);
                    }

                    System.out.println("******************************************");

                    FileReader fr1 = new FileReader("Test.txt");
                    int i = fr1.read();
                    while (i != -1){
                              System.out.print((char) i);
                              i =fr1.read();
                    }
          }
```

In java the maximum size of an array is int only. The length of a file is long

What println will give you?

Buffering

- Usage of FileWriter and FileReader is not recommended for the following reasons:
  - You have to insert line separator manually, which is varied from system to system. It’s difficulty to the programmer.
  - You have to read data character by character which is not convenient to the programmer and it’s very slow.

BufferedWriter

- We can use BufferedWriter to write Text (Character) data to the file.

- Constructors:
  - BufferedWriter bw = new BufferedWriter(Writer w);
  - BufferedWriter bw = new BufferedWriter(Writer w, int Bufferesize);

Note: BufferedWriter can't communicate directly with the file. It can communicate via writer object only.

BufferedWriter

- Which of the following is valid?

  - BufferedWriter bw = new BufferedWriter(“abc.txt”);
  - BufferedWriter bw = new BufferedWriter(new File(“abc.txt”));
  - BufferedWriter bw = new BufferedWriter(new FileWriter (“abc.txt));
  - BufferedWriter bw = new BufferedWriter(new BufferedWriter (new FileWriter(“abc.txt”)));

BufferedWriter

- Which of the following is valid?
  - BufferedWriter bw = new BufferedWriter(“abc.txt”); X
  - BufferedWriter bw = new BufferedWriter(new File(“abc.txt”)); X
  - BufferedWriter bw = new BufferedWriter(new FileWriter (“abc.txt)); √
  - BufferedWriter bw = new BufferedWriter(new BufferedWriter (new FileWriter(“abc.txt”))); √

Methods of BufferedWriter

- write (int ch)
- write (char[] ch)
- write (String s)
- flush()
- close()
- newline(): To insert a line separator.

Q:When compared with FileWriter which of the above method is newly introduced in the BufferedWriter?

BufferedWriter

```
import java.io.*;
public class BufferWriter {
       public static void main(String[] args) throws IOException{
              FileWriter fw = new FileWriter("test.txt");
              BufferedWriter bw = new BufferedWriter(fw);
              bw.write(100);
              bw.newLine();
              char[] ch1 = {'a', 'b', 'c', 'd'};
              bw.write(ch1);
              bw.newLine();
              bw.write("CECS277");
              bw.newLine();
              bw.write("CSULB");
              bw.flush();
              bw.close();
       }
}
```

This will print the Unicode of 100

Methods of BufferedWriter

Note: Closing the BufferedWriter will automatically close the FileWriter and its not required to close it explicitly.

BufferedWriter.close();	FileWriter.close	BufferedWriter.close();
		FileWriter.close();

(Recommended)	(Not Recommended)	(Not Recommended)
√	X	X

BufferedReader

- We can use BufferedReader to read character data (text data) from a file.
- *The main advantage of BufferedReader over Filereader is we can read data line by line in addition to character by character, which is more convenient to the programmer.*
- Constructors:
  - BufferedReader br = new BufferedReader(Reader r);
  - BufferedReader br = new BufferedReader(Reader r, int bufferesize);

Note: BufferedReader can not communicate directly with the files. It can only communicate via reader object only.

Methods of BufferedReader

- int read();
- int read (char[] ch);
- void close();
- string readLine();
  - It Attempts to read next line from the file and returns it if it’s available.
  - If the next line is not available, then it will return null.

BufferedReader

```
import java.io.*;
public class BufferReader {
	public static void main(String[] args) throws IOException{
		FileReader fr = new FileReader ("test.txt");
		BufferedReader br = new BufferedReader(fr);
		//BufferedReader br = new BufferedReader(new FileReader ("test.txt"));
		String line = br.readLine();
		while (line != null){
			System.out.println(line);
			line = br.readLine();
		}
		br.close();
	}
}
```

You can immediately call the file reader.

Note: Closing the BufferedWriter will automatically close the FileWriter and its not required to close it explicitly.

PrintWriter

- FileWriter
  - Insert “\n” after the end of each line.
  - If you add integer it will add Unicode value of the integer value.

    Ex: BufferWriter.add(100); this will add ‘d’ to the file
  - You can’t add bolean, double unless you convert them to String.
- BufferWriter
  - Call “newline ()” after the end of each line.
  - If you add integer it will add Unicode value of the integer value.
  - You can’t add bolean, double unless you convert them to String.

PrintWriter

- It’s the most enhanced writer to write character data to the file.

- The main advantage of PrintWriter is to allow us write any type of primitive type data directly to the file.

- Constructors:
  - PrintWriter pw = new PrintWriter (String filename);
  - PrintWriter pw = new PrintWriter (File f);
  - PrintWriter pw = new PrintWriter (Writer w);

Note: PrintWriter can communicate either directly to the file or via some writer object also

PrintWriter Methods

Write Methods	Print Methods	Println Methods
write (int ch);	print (char ch);	println (char ch);
write (char[] ch);	print (int i);	println (int i);
write (String s);	print (double d);	println (double d);
flush();	print (boolean b);	println (boolean b);
close();	print (String s);	println (String s);
	flush();	flush();
	close();	close();

PrintWriter Methods

What is the difference between pw.write(100); and pw.print(100)?

PrintWriter Methods

- In the case of PrintWriter.write(100); the corresponding character ‘d’ will be added to the file.

- But in case of PrintWriter.print(100); the int value 100 will be added directly to the file.

PrintWriter

```
import java.io.IOException;
import java.io.PrintWriter;

public class PrintWriterExample {
      public static void main(String[] args) throws
IOException{
            PrintWriter pw = new PrintWriter("test.txt");
            pw.write(100);
            pw.println(100);
            pw.println("CECS277");
            pw.println(true);
            pw.flush();
            pw.close();
      }
}
```

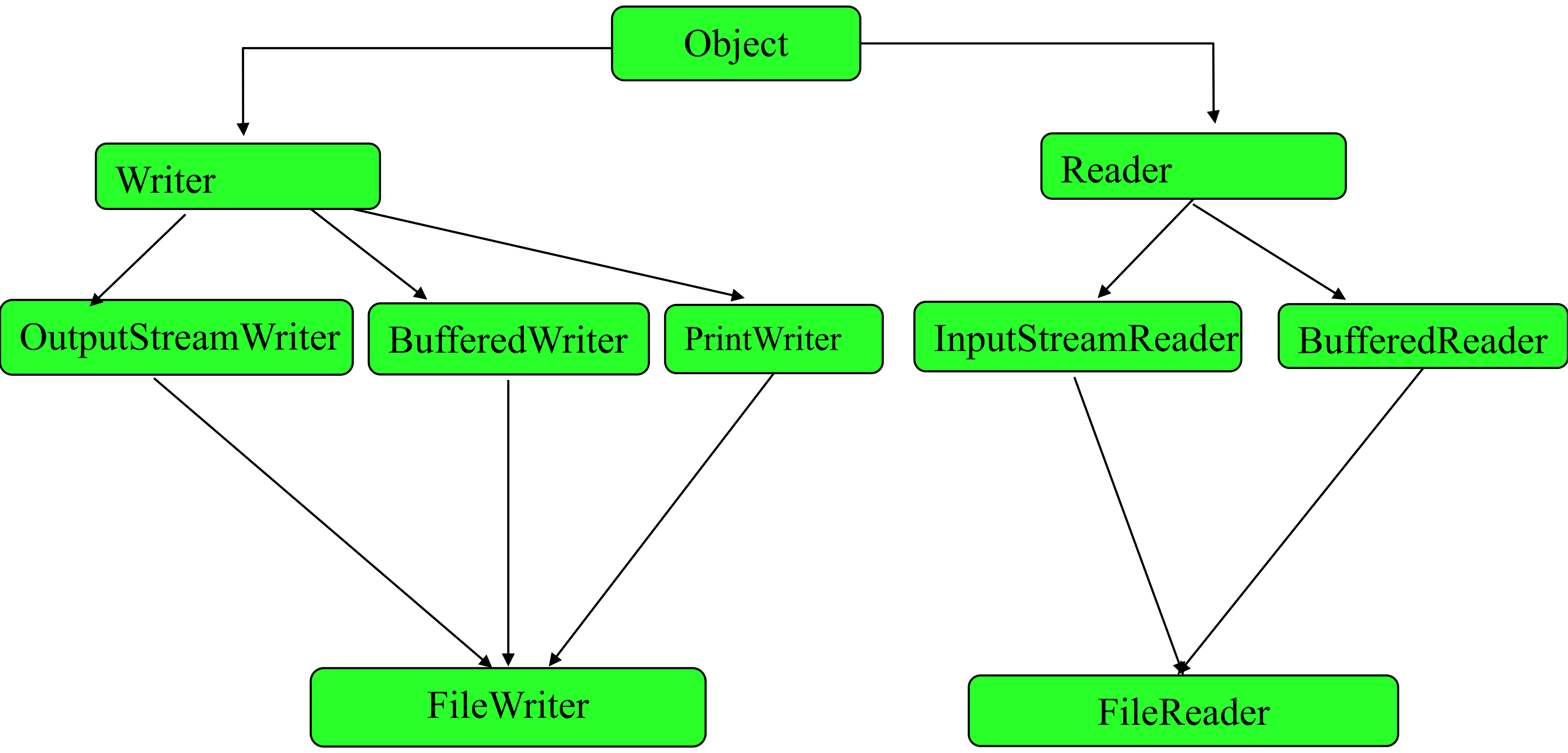
Conclusions

- Conclusion 1:
  - The most enhanced writer to write character data to the file is PrintWriter where is the most enhanced reader to read character data from the file is BufferedReader.

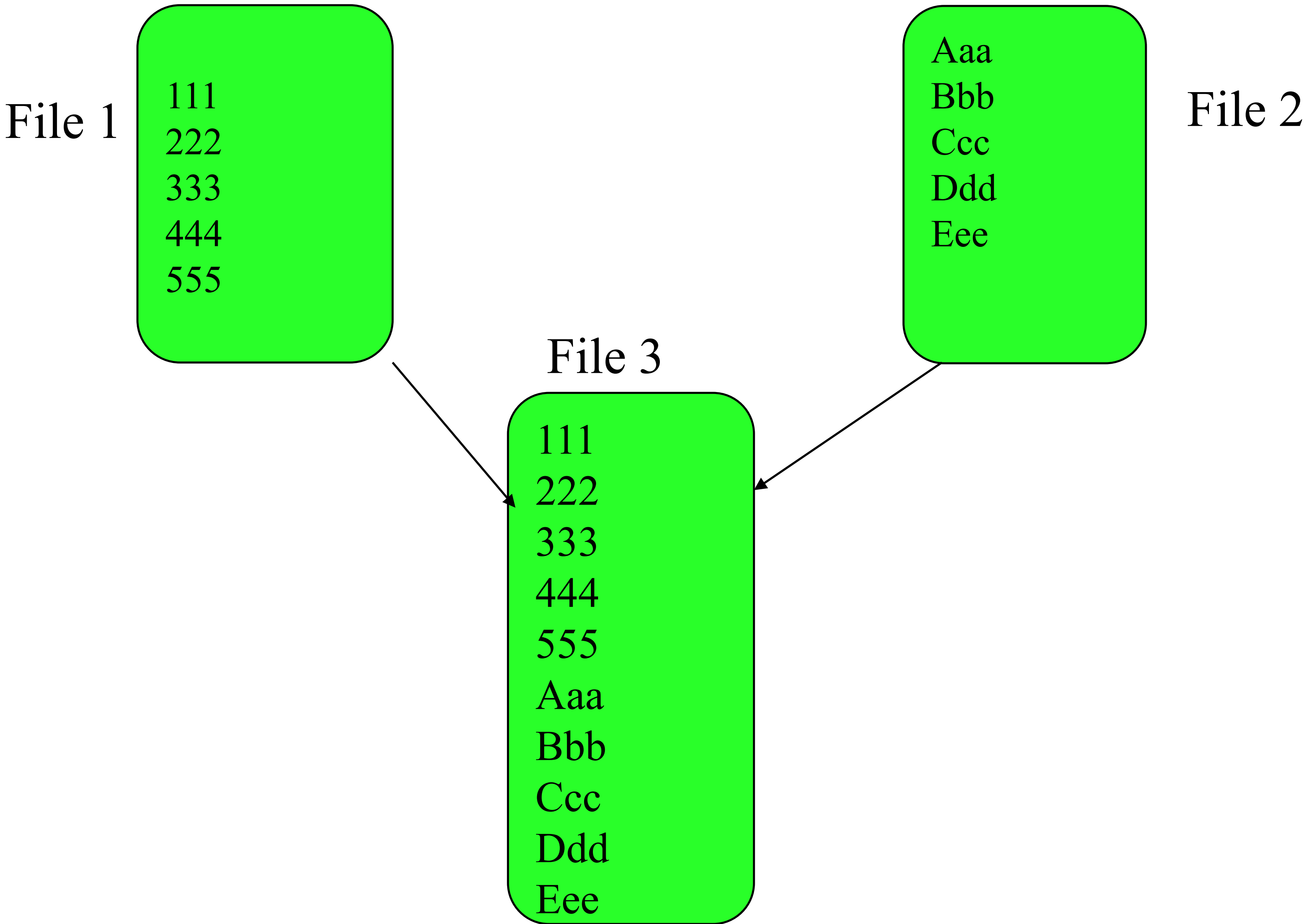
- Conclusion 2:
  - In general we can user Reader and Writer to handle character data (text data) whereas we can use streams to handle binary data.

We can use FileInputStream to read binary data from the file and we can use FileOutputStream to write binary data to the file (like images, video files, audio files, etc).

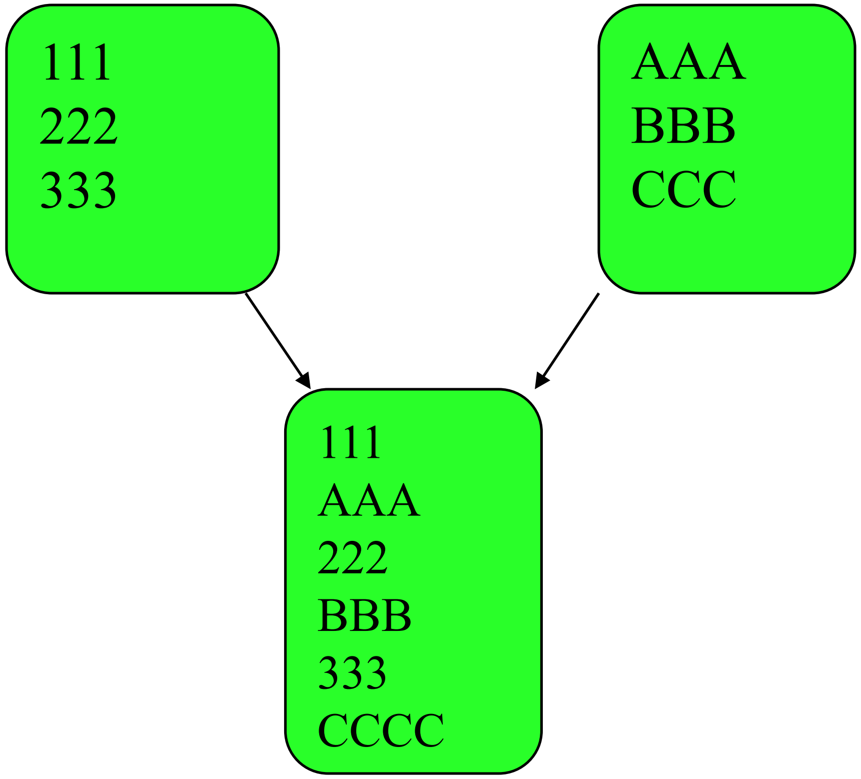
Type of steams



How you can merge two files in one?



- Write a program to perform file merge operation where merging should be done line by line alternatively from each file.


111
222
333
AAA
BBB
CCC
111
AAA
222
BBB
333
CCCC