

# Fayllar bilan ishlash

# Reja:

1. **File Handling in Java;**
2. **Directories in Java**
3. **File Operations in java**
4. **Streams**

# File Handling in Java

- File handling deganda file ichidagilarni o'qish va fayl ichiga yozish nazarda tutiladi.
- **java.io package** dagi File klassi orqali turli formatdagi fayllar bilan ishlash mumkin:

```
// Import the File class
```

```
import java.io.File
```

```
// Specify the filename
```

```
File obj = new File("filename.txt");
```

# Directories in Java

- Directory(papka)- bu fayl bo'lib ichida boshqa fayl va pakalar haqidagi ma'lumotlarni saqlaydi.
- **boolean mkdir()** – directory yaratish;
- **boolean mkdirs()** – bir nechta directory yaratish.

```
import java.io.File;
public class CreateDir {
    public static void main(String args[]) {
        String dirname = "d:/bin";
        File d = new File(dirname);
        d.mkdir();
    }
}
```

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

# Listing directories

```
import java.io.File;
public class ReadDir {
    public static void main(String[] args) {
        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();
        }
    }
}
```

# File Operations in java

- **Create/remove a File;**
- **Get File Information**
- **Write to a File**
- **Read from a File.**

# Java File methods

Method	Type	Description
canRead()	Boolean	It tests whether the file is readable or not
canWrite()	Boolean	It tests whether the file is writable or not
createNewFile()	Boolean	This method creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	It tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory



# Create File

```
// Import the File class
import java.io.File;
// Import the IOException class to handle errors
import java.io.IOException;

public class CreateFile {
    public static void main(String[] args) {
        try {
            // Creating an object of a file
            File myObj = new File("D:FileHandlingNewFilef1.txt");
            if (myObj.createNewFile()) {
                System.out.println("File created: " + myObj.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```



# Get File Information

```
import java.io.File; // Import the File class

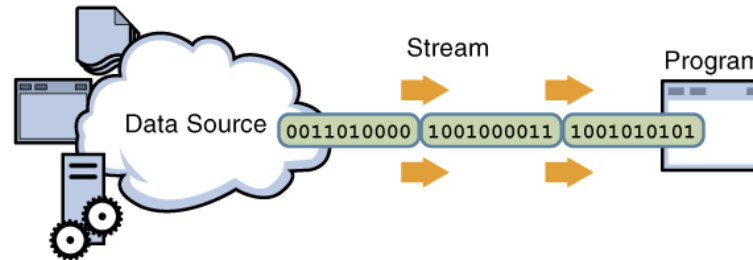
public class FileInformation {
    public static void main(String[] args) {
        // Creating an object of a file
        File myObj = new File("NewFile1.txt");
        if (myObj.exists()) {
            // Returning the file name
            System.out.println("File name: " + myObj.getName());
            // Returning the path of the file
            System.out.println("Absolute path: " + myObj.getAbsolutePath());
            // Displaying whether the file is writable
            System.out.println("Writable: " + myObj.canWrite());
            // Displaying whether the file is readable or not
            System.out.println("Readable " + myObj.canRead());
            // Returning the length of the file in bytes
            System.out.println("File size in bytes " + myObj.length());
        } else {
            System.out.println("The file does not exist.");
        }
    }
}
```

# Input and Output

## Input and Output

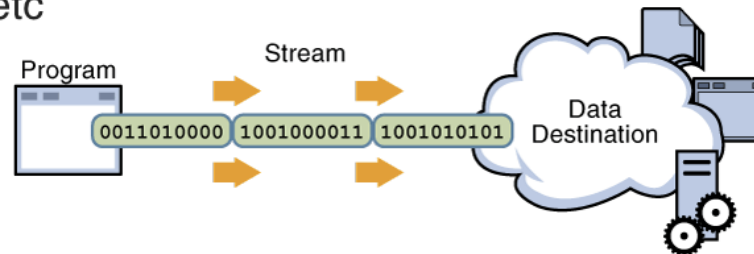
- Inputs

- command line arguments, files, network, gamepads, keyboard, mouse, temperature sensor, webcam, other processes, etc



- Outputs

- files, network, gamepad rumble, monitor, LEDs, speakers, robot motor, etc




# What is a Stream?

Java fayl ustida kiritish va chiqarish amallarini bajarish uchun **Stream(oqim)** konsepsiyasini qo'llaydi.


Javada Stream deganda quyidagi 2 xil toifadagi ma'lumotlar ketma-ketligi nazarda tutiladi:

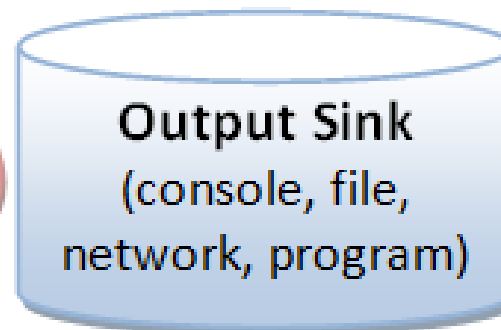
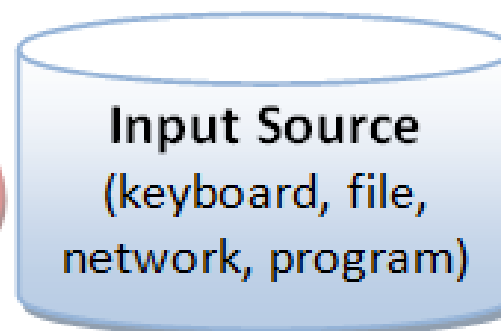
- **Byte Stream** – byte ko'rinishdagi ma'lumotlar ketma-ketligi bo'lib, faylga ma'lumotlarni byte ko'rinishda yozish yoki o'qilshda qo'llaniladi.
- **Character Stream** - char ko'rinishdagi ma'lumotlar ketma-ketligi bo'lib, faylga ma'lumotlarni char ko'rinishda yozish yoki o'qishda qo'llaniladi.

**“Character” Streams**  
(Reader/Writer)

 char  
(16-bit)

**“Byte” Streams**  
(InputStream/  
OutputStream)

 Byte  
(8-bit)



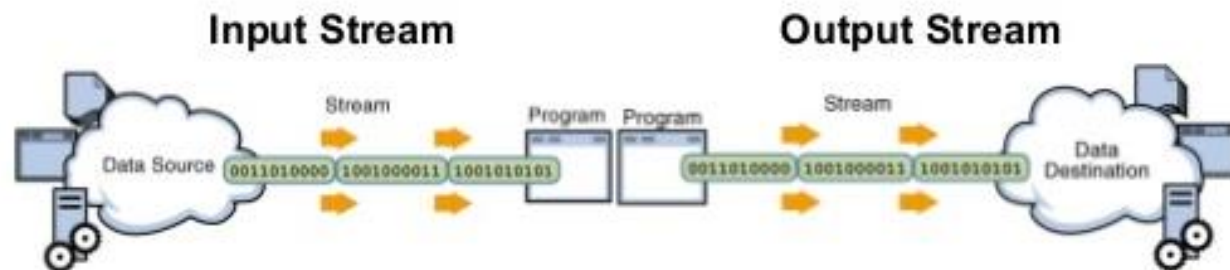
Internal Data Formats:

- Text (char): UCS-2
- int, float, double, etc.

External Data Formats:

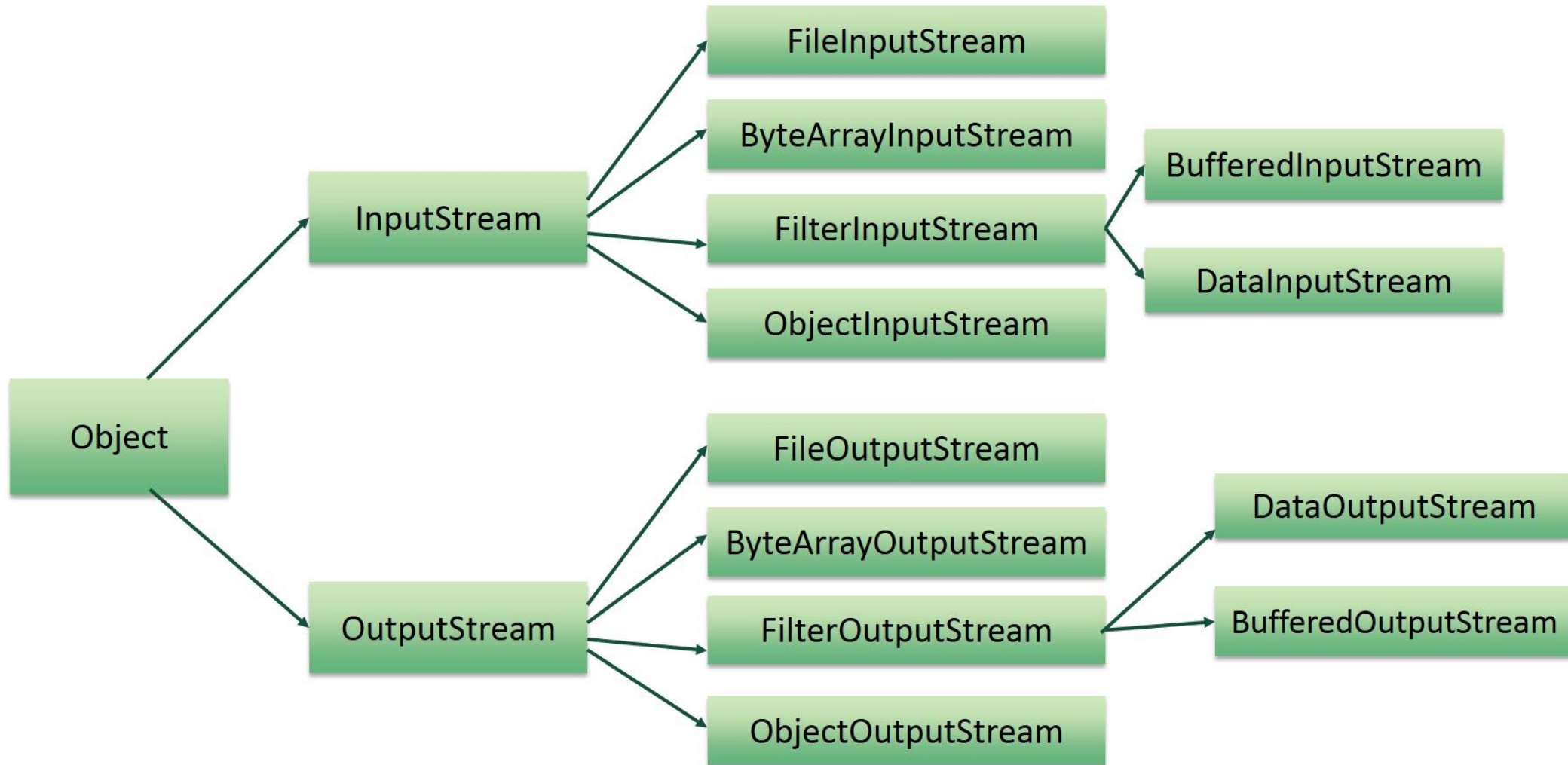
- Text in various encodings  
(US-ASCII, ISO-8859-1, UCS-2, UTF-8,  
UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

# I/O Streams

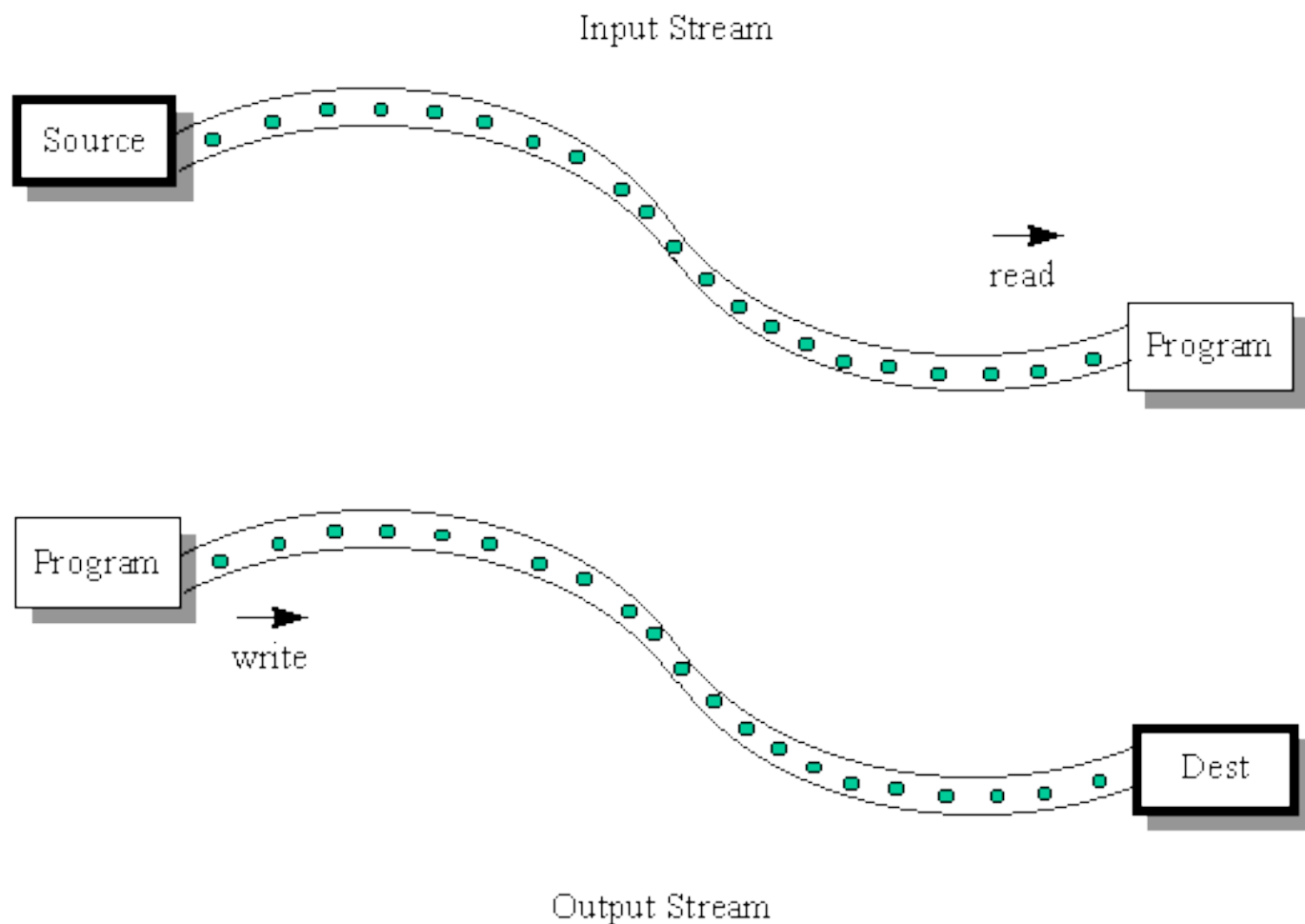




# InputStream and OutputStream



# InputStream and OutputStream

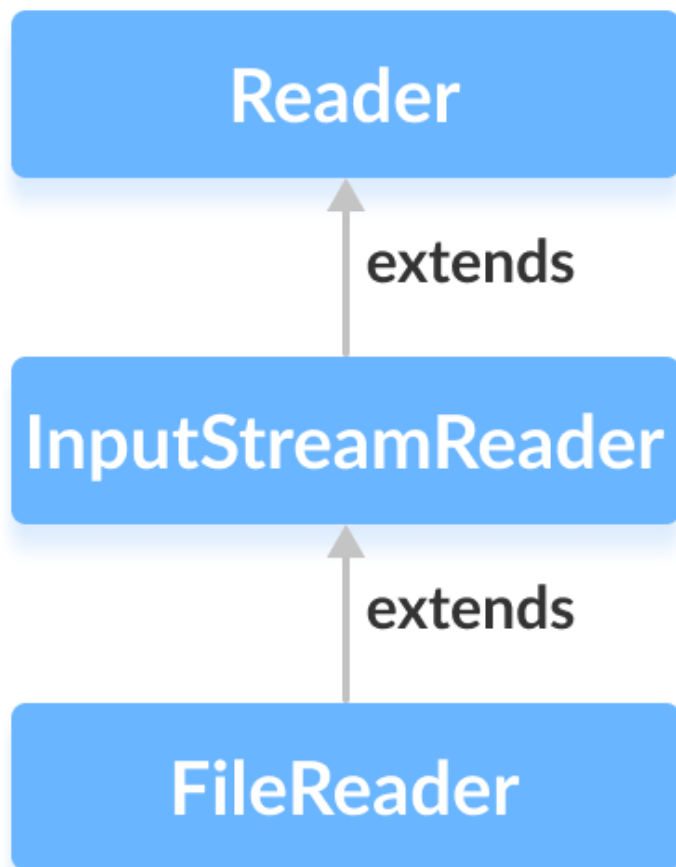




# InputStream and OutputStream

```
public static void main(String[] args) throws IOException {  
    InputStream inputStream = new FileInputStream(  
        new File("src/main/resources/sample.txt"));  
    byte[] buffer = new byte[inputStream.available()];  
    inputStream.read(buffer);  
  
    File targetFile = new File("src/main/resources/targetFile.tmp");  
    OutputStream outputStream = new FileOutputStream(targetFile);  
    outputStream.write(buffer);  
  
}
```

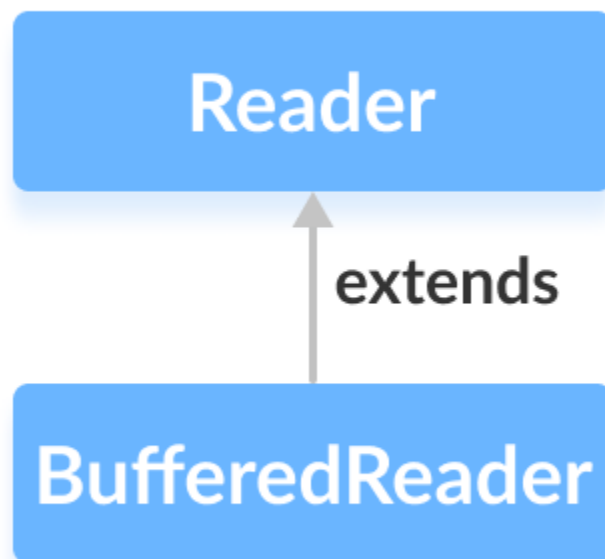
# Reader



# FileReader

```
public static void main(String[] args) {  
    // Creates an array of character  
    char[] array = new char[100];  
    try {  
        // Creates a reader using the FileReader  
        FileReader input = new FileReader("input.txt");  
  
        // Reads characters  
        input.read(array);  
        System.out.println("Data in the file: ");  
        System.out.println(array);  
  
        // Closes the reader  
        input.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

# BufferedReader



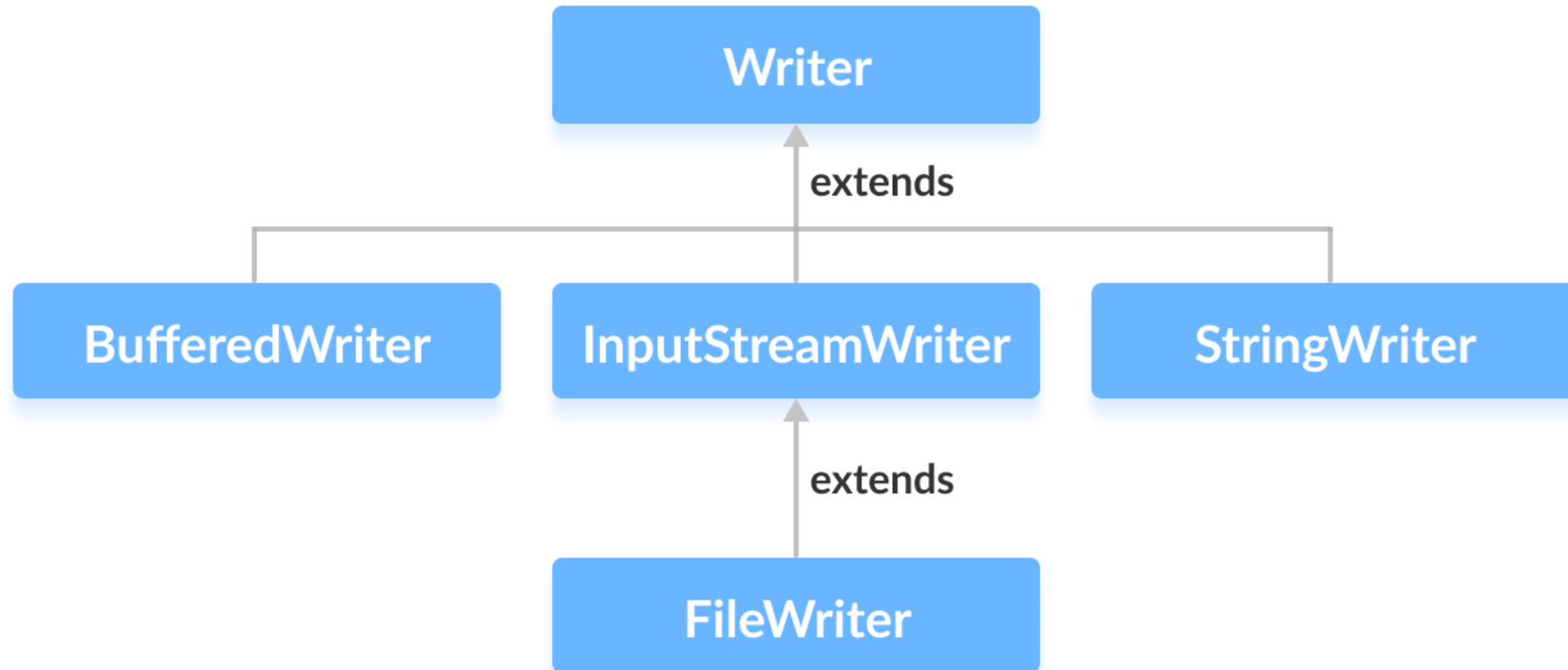
# BufferedReader

```
public static void main(String[] args) {  
    // Creates an array of character  
    char[] array = new char[100];  
    try {  
        // Creates a FileReader  
        FileReader file = new FileReader("input.txt");  
        // Creates a BufferedReader  
        BufferedReader input = new BufferedReader(file);  
        // Reads characters  
        input.read(array);  
        System.out.println("Data in the file: ");  
        System.out.println(array);  
        // Closes the reader  
        input.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

# Scanner

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner("input.txt");  
    while (scanner.hasNext()) {  
        System.out.println(scanner.nextLine());  
    }  
}
```

# Writer

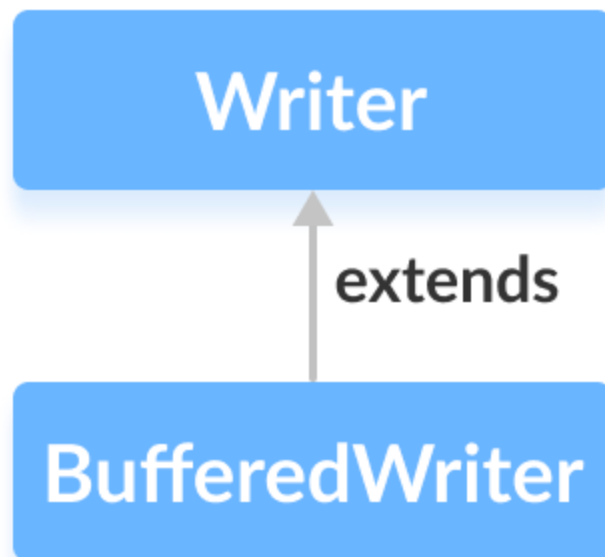




# FileWriter

```
public static void main(String args[]) {  
    String data = "This is the data in the output file";  
    try {  
        // Creates a Writer using FileWriter  
        Writer output = new FileWriter("output.txt");  
  
        // Writes string to the file  
        output.write(data);  
  
        // Closes the writer  
        output.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

# BufferedWriter

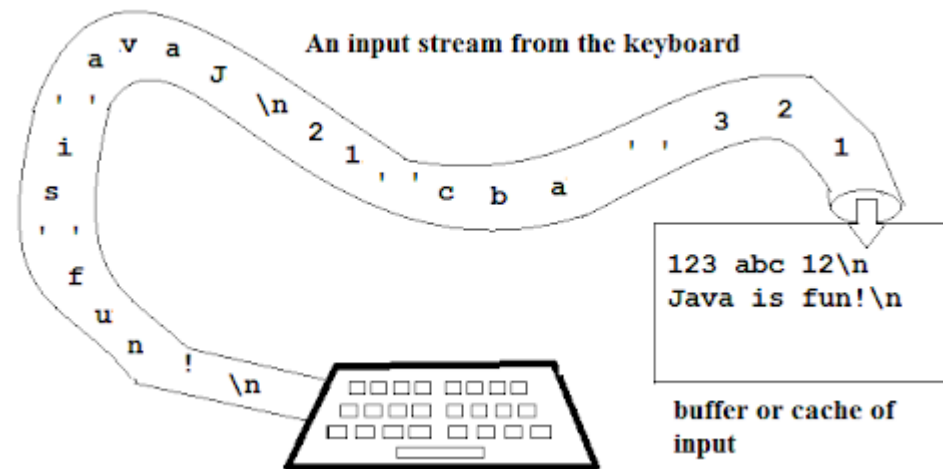


# BufferedWriter

```
public static void main(String args[]) {  
    String data = "This is the data in the output file";  
    try {  
        // Creates a FileWriter  
        FileWriter file = new FileWriter("output.txt");  
        // Creates a BufferedWriter  
        BufferedWriter output = new BufferedWriter(file);  
        // Writes the string to the file  
        output.write(data);  
        // Closes the writer  
        output.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

# Standard Streams

- Standard Input => `System.in`.
- Standard Output=>`System.out`.
- Standard Error => `System.err`.



# Standard Streams

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

**E'TIBORINGIZ UCHUN RAXMAT**