Java Basic for Tester

Java IO Streams

Agenda

- *Java I/O Streams*
- *Java InputStream*
- *Java OutputStream*
- *Java FileInputStream*
- *Java FileOutputStream*
- *Java ObjectInputStream*
- *Java ObjectOutputStream*
- *Java PrintStream*

Java I/O Streams

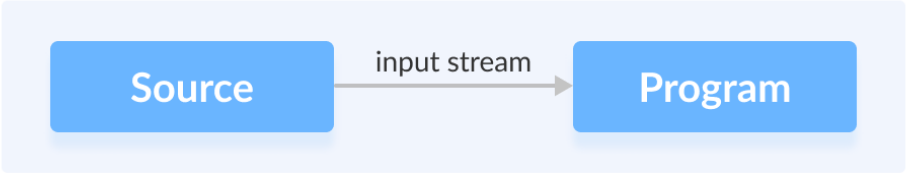
In Java, streams are the sequence of data that are read from the source and written to the destination.

An input stream is used to read data from the source. And, an output stream is used to write data to the destination.

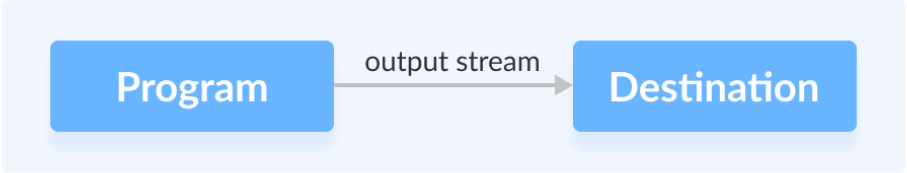
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Java I/O Streams

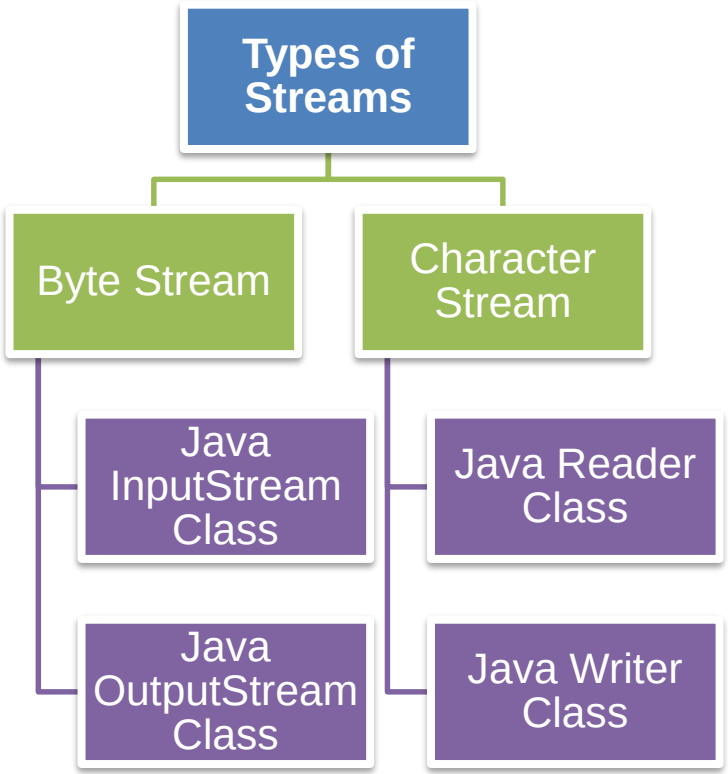
Reading data from source



Writing data to destination



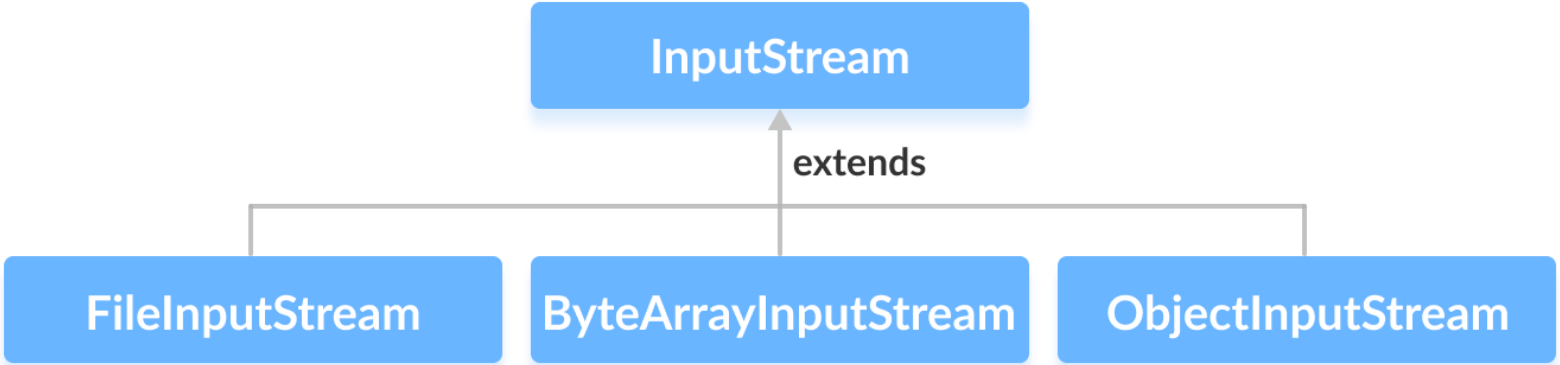
Java I/O Streams



Java InputStream

The InputStream class of the java.io package is an abstract superclass that represents an input stream of bytes.

Since InputStream is an abstract class, it is not useful by itself. However, its subclasses can be used to read data.



Java InputStream

In order to create an InputStream, we must import the java.io.InputStream package first. Once we import the package, here is how we can create the input stream.

```
InputStream object1 = new FileInputStream();
```

Here, we have created an input stream using FileInputStream. It is because InputStream is an abstract class. Hence we cannot create an object of InputStream.

Java InputStream

Methods of InputStream

read() - reads one byte of data from the input stream
read(byte[] array) - reads bytes from the stream and stores in the specified array
available() - returns the number of bytes available in the input stream
mark() - marks the position in the input stream up to which data has been read
reset() - returns the control to the point in the stream where the mark was set
markSupported() - checks if the mark() and reset() method is supported in the stream
skips() - skips and discards the specified number of bytes from the input stream
close() - closes the input stream

Java InputStream

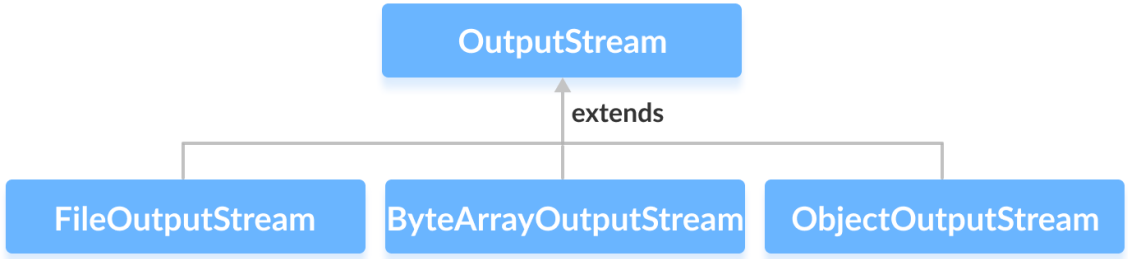
```java
package IO;
import java.io.FileInputStream;
import java.io.InputStream;
public class InputStreamDemo {
    public static void main(String args[]) {
        byte[] array = new byte[100];
        try {
            InputStream input = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            System.out.println("Available bytes in the file: " + input.available());
            // Read byte from the input stream
            input.read(array);
            System.out.println("Data read from the file: ");
            // Convert byte array into string
            String data = new String(array);
            System.out.println(data);
            // Close the input stream
            input.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  InputStreamDemo
"C:\Program Files\Java\jdk-14.0.1\bin\j
Available bytes in the file: 38
Data read from the file:
This is a line of text inside the file
```

Java OutputStream

The OutputStream class of the java.io package is an abstract superclass that represents an output stream of bytes.

Since OutputStream is an abstract class, it is not useful by itself. However, its subclasses can be used to write data.



Java OutputStream

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

```
OutputStream object = new FileOutputStream();
```

Here, we have created an object of output stream using FileOutputStream. It is because OutputStream is an abstract class, so we cannot create an object of OutputStream.

Java OutputStream

Methods of OutputStream

write() - writes the specified byte to the output stream

write(byte[] array) - writes the bytes from the specified array to the output stream

flush() - forces to write all data present in output stream to the destination

close() - closes the output stream

Methods of OutputStream

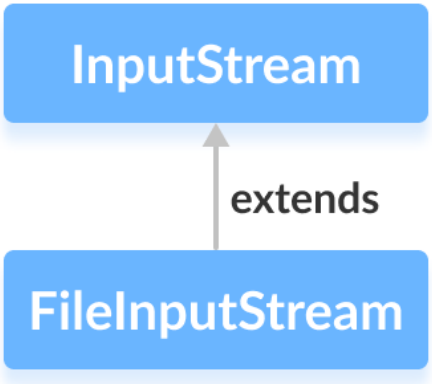
```java
package IO;
import java.io.FileOutputStream;
import java.io.OutputStream;
public class OutputStreamDemo {
    public static void main(String args[]) {
        String data = "This is a line of text inside the file.";
        try {
            OutputStream out = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\output.txt");
            // Converts the string into bytes
            byte[] dataBytes = data.getBytes();
            // Writes data to the output stream
            out.write(dataBytes);
            System.out.println("Data is written to the file.");
            // Closes the output stream
            out.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  OutputStreamDemo
"C:\Program Files\Java\jdk-14.
Data is written to the file.
```

Java FileInputStream

The FileInputStream class of the java.io package can be used to read data (in bytes) from files.

It extends the InputStream abstract class.



Java FileInputStream

Create a FileInputStream

```
FileInputStream input = new FileInputStream(stringPath);
```

```
FileInputStream input = new FileInputStream(File fileObject);
```

Java FileInputStream

read() Method

read() - reads a single byte from the file

read(byte[] array) - reads the bytes from the file and stores in the specified array

read(byte[] array, int start, int length) - reads the number of bytes equal to length from the file and stores in the specified array starting from the position start

Java FileInputStream

read() Method

```java
package IO;
import java.io.FileInputStream;
public class ReadFileInputStream {
    public static void main(String args[]) {
        try {
            FileInputStream input = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            System.out.println("Data in the file: ");
            // Reads the first byte
            int i = input.read();
            while(i != -1) {
                System.out.print((char)i);
                // Reads next byte from the file
                i = input.read();
            }
            input.close();
        }
        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  ReadFileInputStream
"C:\Program Files\Java\jdk-14.0.1\bin\ja
Data in the file:
This is a line of text inside the file
```

Java FileInputStream

available() Method: To get the number of available bytes

```java
package IO;
import java.io.FileInputStream;
public class AvailableFileInputStream {
    public static void main(String args[]) {
        try {
            // Suppose, the input.txt file contains the following text
            // This is a line of text inside the file.
            FileInputStream input = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            // Returns the number of available bytes
            System.out.println("Available bytes at the beginning: " + input.available());
            // Reads 3 bytes from the file
            input.read();
            input.read();
            input.read();
            // Returns the number of available bytes
            System.out.println("Available bytes at the end: " + input.available());
            input.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run: AvailableFileInputStream
"C:\Program Files\Java\jdk-14.0.1\bir
Available bytes at the beginning: 38
Available bytes at the end: 35
```

Java FileInputStream

skip() Method: To discard and skip the specified number of bytes

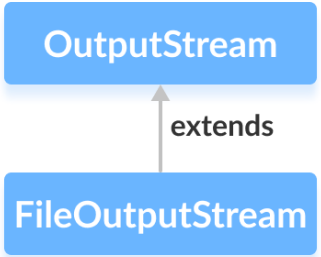
```java
package IO;
import java.io.FileInputStream;
public class SkipFileInputStream {
    public static void main(String args[]) {
        try {
            // Suppose, the input.txt file contains the following text
            // This is a line of text inside the file.
            FileInputStream input = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            // Skips the 5 bytes
            input.skip( n: 5);
            System.out.println("Input stream after skipping 5 bytes:");
            // Reads the first byte
            int i = input.read();
            while (i != -1) {
                System.out.print((char) i);
                // Reads next byte from the file
                i = input.read();
            }
            // close() method
            input.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  SkipFileInputStream
"C:\Program Files\Java\jdk-14.0.1\bir
Input stream after skipping 5 bytes:
is a line of text inside the file
```

Java FileOutputStream

The FileOutputStream class of the java.io package can be used to write data (in bytes) to the files.

It extends the OutputStream abstract class.



Java FileOutputStream

In order to create a file output stream, we must import the java.io.FileOutputStream package first. Once we import the package, here is how we can create a file output stream in Java.

1. Using the path to file

```
// Including the boolean parameter
FileOutputStream output = new FileOutputStream(String path, boolean value);

// Not including the boolean parameter
FileOutputStream output = new FileOutputStream(String path);
```

2. Using an object of the file

```
FileOutputStream output = new FileOutputStream(File fileObject);
```

Java FileOutputStream

In order to create a file output stream, we must import the java.io.FileOutputStream package first. Once we import the package, here is how we can create a file output stream in Java.

1. Using the path to file

```
// Including the boolean parameter
FileOutputStream output = new FileOutputStream(String path, boolean value);

// Not including the boolean parameter
FileOutputStream output = new FileOutputStream(String path);
```

2. Using an object of the file

```
FileOutputStream output = new FileOutputStream(File fileObject);
```

write() Method

write() - writes the single byte to the file output stream

write(byte[] array) - writes the bytes from the specified array to the output stream

write(byte[] array, int start, int length) - writes the number of bytes equal to length to the output stream from an array starting from the position start

write() Method

```java
package IO;

import java.io.FileOutputStream;

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

        String data = "This is a line of text inside the file.";

        try {
            FileOutputStream output = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\output.txt");

            byte[] array = data.getBytes();

            // Writes byte to the file
            output.write(array);

            output.close();
        }

        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

Java FileOutputStream

flush() Method: To clear the output stream

```java
package IO;

import java.io.FileOutputStream;
import java.io.IOException;

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

        FileOutputStream out = null;
        String data = "This is demo of flush method";

        try {
            out = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\output.txt");

            // Using write() method
            out.write(data.getBytes());

            // Using the flush() method
            out.flush();
            out.close();
        }
        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

Java FileOutputStream

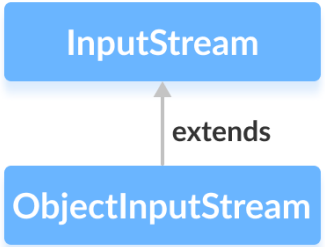
flush() Method: To clear the output stream

```java
package IO;

import java.io.FileOutputStream;
import java.io.IOException;

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

        FileOutputStream out = null;
        String data = "This is demo of flush method";

        try {
            out = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\output.txt");

            // Using write() method
            out.write(data.getBytes());

            // Using the flush() method
            out.flush();
            out.close();
        }
        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

Java ObjectInputStream

The ObjectInputStream class of the java.io package can be used to read objects that were previously written by ObjectOutputStream.

It extends the InputStream abstract class.



Java ObjectInputStream

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

```
// Creates a file input stream linked with the specified file
FileInputStream fileStream = new FileInputStream(String file);

// Creates an object input stream using the file input stream
ObjectInputStream objStream = new ObjectInputStream(fileStream);
```

read() Method

read() - reads a byte of data from the input stream

readBoolean() - reads data in boolean form

readChar() - reads data in character form

readInt() - reads data in integer form

readObject() - reads the object from the input stream

read() Method

```java
public class JavaObjectInputStream {
    public static void main(String[] args) {
        int data1 = 5;
        String data2 = "LamNS3";
        try {
            FileOutputStream file = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            ObjectOutputStream output = new ObjectOutputStream(file);
            // Writing to the file using ObjectOutputStream
            output.writeInt(data1);
            output.writeObject(data2);
            FileInputStream fileStream = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\input.txt");
            // Creating an object input stream
            ObjectInputStream objStream = new ObjectInputStream(fileStream);
            //Using the readInt() method
            System.out.println("Integer data :" + objStream.readInt());
            // Using the readObject() method
            System.out.println("String data: " + objStream.readObject());
            output.close();
            objStream.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  JavaObjectInputStream
"C:\Program Files\Java
Integer data :5
String data: LamNS3
```

read() Method

```java
public class JavaObjectInputStream1 implements Serializable {
    String name;

    public JavaObjectInputStream1(String name) {
        this.name = name;
    }
}
```

```
Run:  Main
"C:\Program File
Name: LamNS3
```

```java
class Main {
    public static void main(String[] args) {

        // Creates an object of JavaObjectInputStream1 class
        JavaObjectInputStream1 object = new JavaObjectInputStream1( name: "LamNS3");

        try {
            FileOutputStream file = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            // Creates an ObjectOutputStream
            ObjectOutputStream output = new ObjectOutputStream(file);

            // Writes objects to the output stream
            output.writeObject(object);

            FileInputStream fileStream = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            // Creates an ObjectInputStream
            ObjectInputStream input = new ObjectInputStream(fileStream);

            // Reads the objects
            JavaObjectInputStream1 newObject = (JavaObjectInputStream1) input.readObject();

            System.out.println("Name: " + newObject.name);

            output.close();
            input.close();
        } catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

read() Method

```java
public class JavaObjectInputStream1 implements Serializable {
    String name;

    public JavaObjectInputStream1(String name) {
        this.name = name;
    }
}
```

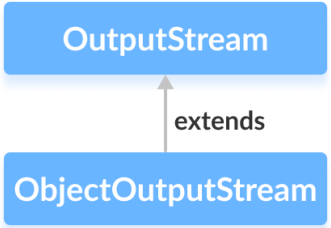
```java
class Main {
    public static void main(String[] args) {

        // Creates an object of JavaObjectInputStream1 class
        JavaObjectInputStream1 object = new JavaObjectInputStream1( name: "LamNS3");

        try {
            FileOutputStream file = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            // Creates an ObjectOutputStream
            ObjectOutputStream output = new ObjectOutputStream(file);

            // Writes objects to the output stream
            output.writeObject(object);

            FileInputStream fileStream = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            // Creates an ObjectInputStream
            ObjectInputStream input = new ObjectInputStream(fileStream);

            // Reads the objects
            JavaObjectInputStream1 newObject = (JavaObjectInputStream1) input.readObject();

            System.out.println("Name: " + newObject.name);

            output.close();
            input.close();
        } catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  Main
"C:\Program File
Name: LamNS3
```

Java ObjectOutputStream

The ObjectOutputStream class of the java.io package can be used to write objects that can be read by ObjectInputStream.

It extends the OutputStream abstract class.



Java 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);
```

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

write() Method

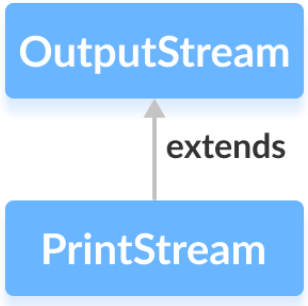
```java
public class JavaObjectOutputStream {
    public static void main(String[] args) {
        int data1 = 5;
        String data2 = "LamNS3";
        try {
            FileOutputStream file = new FileOutputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");
            // Creates an ObjectOutputStream
            ObjectOutputStream output = new ObjectOutputStream(file);
            // writes objects to output stream
            output.writeInt(data1);
            output.writeObject(data2);
            // Reads data using the ObjectInputStream
            FileInputStream fileStream = new FileInputStream( name: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");
            ObjectInputStream objStream = new ObjectInputStream(fileStream);
            System.out.println("Integer data :" + objStream.readInt());
            System.out.println("String data: " + objStream.readObject());
            output.close();
            objStream.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Run:  JavaObjectOutputStream
"C:\Program Files\Ja
Integer data :5
String data: LamNS3
```

Java PrintStream

The PrintStream class of the java.io package can be used to write output data in commonly readable form (text) instead of bytes.

It extends the abstract class OutputStream.



Java PrintStream

In order to create a PrintStream, we must import the java.io.PrintStream package first. Once we import the package here is how we can create the print stream.

1. Using other output streams

```
// Creates a FileOutputStream
FileOutputStream file = new FileOutputStream(String file);

// Creates a PrintStream
PrintStream output = new PrintStream(file, autoFlush);
```

2. Using filename

```
 // Creates a PrintStream
PrintStream output = new PrintStream(String file, boolean autoFlush);
```

Java PrintStream

print() Method

print() - prints the specified data to the output stream

println() - prints the data to the output stream along with a new line character at the end

Java PrintStream

print() method with System class

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Java PrintStream

print() method with PrintStream class

```java
package IO;

import java.io.PrintStream;

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

        String data = "This is a text inside the file.";

        try {
            PrintStream output = new PrintStream( fileName: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            output.print(data);
            output.close();
        }
        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

printf() Method

```java
package IO;

import java.io.PrintStream;

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

        try {
            PrintStream output = new PrintStream( fileName: System.getProperty("user.dir")+"\\src\\main\\resources\\file.txt");

            int age = 25;

            output.printf("I am %d years old.", age);
            output.close();
        }
        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

Thank you