CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

PrintStream

Overview

1. Introduction
2. PrintStream class declaration
3. PrintStream constructors
4. PrintStream methods
5. PrintStream Examples

Introduction

- **PrintStream in Java** is an output stream that provides various methods to print representations of various data values conveniently.

- For example, `System.out` is a `PrintStream` that is the most commonly used output stream in Java programs. It is an instance of `PrintStream` class. It represents a standard output stream and is generally used to write text output to the console.

- `System.err` is also an instance of `PrintStream` class. It represents a standard error stream and displays error messages similar to `System.out`.

- Since `PrintStream` is a byte output stream, it converts all the characters into bytes using the specified encoding or charset in the `PrintStream()` constructor invocation, or platform's default character encoding if not specified.

Introduction

Note:

a. `System` is a `final class` stored in `java.lang` package. It contains three i/o objects: namely `System.in`, `System.out`, and `System.err`.

b. `in`, `out`, and `err` are `static variables` (class variables. The variable in is of type `InputStream` and the other two are of type `PrintStream`.

c. `System` class is responsible for performing standard input/output operation.

Functionality added to Output Stream

`PrintStream` is not limited to a console. It is a filter output stream that can also be connected to any other output stream: *`FileOutputStream`, `ByteArrayOutputStream`, `BufferedOutputStream`*, etc.

It adds the following functionality to another underlying output stream:

1. `PrintStream` provides methods to print any data values, primitive, or object in an appropriate format for printing.

2. Its methods never throw an `IOException` while writing data to an underlying output stream. If a method call throws an exception named `IOException`, it sets an internal flag, rather than directly throwing the exception to the caller. The flag can be checked by using its `checkError()` method. This method returns true if an `IOException` occurs during method execution.

3. `PrintStream` adds an auto-flush capability to an underlying output stream to flush the contents written to it automatically. If we specify the auto flush flag to true in its constructor, it will flush its contents.

PrintStream class declaration

`PrintStream` is a concrete subclass of `FilterOutputStream` that extends `OutputStream` superclass. It implements `Closeable, Flushable, Appendable,` and `AutoCloseable` interfaces.

The general syntax to declare `PrintStream` class in Java is as follows:

```
public class PrintStream
       extends FilterOutputStream
       implements Appendable, Closeable
```

`PrintStream` class was added in Java 1.0 version. It is present in `java.io.PrintStream` package. `LogStream` extends `PrintStream` class.

The `PrintStream` class implements an `Appendable` interface in Java 5.0 that makes it suitable for use with a `java.util.Formatter`.

PrintStream Constructors

`PrintStream` class defines several different types of constructors in Java that are as follows:

1. PrintStream(File file):
This constructor creates a new print stream object with the specified file object as a parameter without automatic line flushing.

2. PrintStream(File file, String cset):
This constructor creates a new print stream with the specified file object and charset, without automatic line flushing.

3. PrintStream(File file, Charset charset):
This constructor creates a new print stream with the specified file object and charset object without automatic line flushing.

PrintStream Constructors

`PrintStream` class defines several different types of constructors in Java that are as follows:

4. PrintStream(OutputStream os):
This constructor creates a new print stream without automatic flushing.

5. PrintStream(OutputStream os, boolean autoFlush):
It creates a new print stream with a specified output stream os and autoFlush. If autoFlush is true, flushing automatically takes place by the output buffer.

6. PrintStream(OutputStream out, boolean autoFlush, String encoding):
It creates a new print stream with an additional parameter that is the name of supported character encoding.

7. PrintStream(OutputStream os, boolean autoFlush, Charset charset):
This constructor creates a new print stream with the specified OutputStream os, automatic line flushing, and charset.

PrintStream Constructors

`PrintStream` class defines several different types of constructors in Java that are as follows:

8. PrintStream(String fileName):
This constructor creates a new print stream with the specified file name without automatic line flushing.

9. PrintStream(String fileName, String cset):
This constructor constructs a new print stream, without automatic line flushing, with the specified file name and charset.

10. PrintStream(String fileName, Charset charset):
This constructor constructs a new print stream, without automatic line flushing, with the specified file name and charset object.

PrintStream Constructors

Note:

a) The parameter `os` defines an output stream to which values and objects will be printed.

b) Constructors can throw `FileNotFoundException` if the file is not found and `UnsupportedEncodingException`

PrintStream Methods

In addition to methods inherited from `FilterOutputStream` class, `PrintStream` class also defines some useful methods. These methods are as follows:

PrintStream Methods

Method	Description
PrintStream append(char c)	This method appends the specified character to this output stream and returns stream.
PrintStream append(CharSequence cs)	This method appends the specified character sequence to this output stream and returns stream.
PrintStream append(CharSequence cs, int begin, int end)	This method appends a subsequence of the specified character sequence to this output stream and returns stream. The subsequence starts from the index begin and extends to the character at index end-1.
boolean checkError()	This method flushes the stream and checks its internal error state. It returns true if and only if this stream has encountered an IOException.
protected void clearError()	This method clears the internal error state of this stream.

PrintStream Methods

Method	Description
void close()	It closes the stream.
void flush()	This method flushes the stream.
PrintStream printf(String format, Object… args)	This method flushes the stream and checks its internal error state. It returns true if and only if this stream has encountered an IOException.
void println()	This method terminates the current line by writing the line separator string.
void println(boolean x)	It prints a boolean and then ends the line.
void println(char x)	This method prints the specified character x and then terminates the line.

PrintStream Methods

Method	Description
void println(char[ ] x)	This method prints an array of characters and then terminates the line.
void println(double x)	This method prints the specified double value and then terminates the line.
void println(float x)	This method prints the specified float value and then terminates the line.
void println(int x)	This method prints the specified integer value and then terminates the current line.
void println(long x)	This method prints a long value and then terminates the line.
void println(Object x)	This method prints an Object and then terminates the current line.
void println(String x)	This method prints a String value and then terminates the line.

PrintStream Methods

Method	Description
protected void setError()	This method sets the error state of the stream to true.
void write(byte[ ] b, int n, int m)	It writes m bytes from the specified byte array starting at nth byte to this stream.
void write(int b)	This method writes the specified byte to the underlying output stream.

Example 1: Display data with println()

1. Let’s take an example program where we will display some data using `println()` method of `PrintStream` class. Look at the source code.

Example 1: Display data with println()

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Date;
public class PrintStreamTester1 {
   public static void main(String[] args) throws IOException {
       FileOutputStream fos = new FileOutputStream("./src/outfile.txt");
       PrintStream ps = new PrintStream(fos);

       // Printing char value.
       ps.println('A');

       // Printing int value.
       ps.println(2000);

       // Printing string value.
       ps.println("I love Java Programming");

       // Printing current date.
       ps.println(new Date());

       ps.close();
       fos.close();
   }
}
```

Example 1: Display data with println()

Output:

```
A
2000
I love Java Programming
Mon Jul 11 07:37:37 CDT 2022
```

`outfile.txt file` contains the above output data

Example 2: Display data with print()

2. PrintStream class in Java also supports print() method for all types, including Object. The main difference between print() and println() methods are

- The print() method prints specified data to the output stream.
- The println() method prints specified data to the output stream along with a new line character at the end.

Let’s understand with the help of an example program.

Example 2: Display data with print()

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
public class PrintStreamTester2 {
   public static void main(String[] args) throws IOException {
       FileOutputStream fos = new FileOutputStream("./src/outfile.txt");
       PrintStream ps = new PrintStream(fos);

       // Printing char value.
       ps.print('S');

       // Printing int value.
       ps.print(5500);

       // Printing string value.
       ps.println("I love Java Programming");

       // Printing char value.
       ps.println('S');

       ps.close();
       fos.close();
   }
}
```

Example 2: Display data with print()

Output:

```
1  S5500I love Java Programming
2  S
```

`outfile.txt file` contains the above output data

As you can observe in the output, print() method does not provide a new line character at the end.

Example 3: Display data with printf()

3. Let’s take an example program where we will print some data using printf() method of PrintStream class. This method was added in PrintStream with the release of JDK 5.

Example 3: Display data with printf()

```
public class PrintStreamTester3 {
   public static void main(String[] args) {
       System.out.println("Displaying some numeric values " + "in different formats.\n");

       System.out.printf("Displaying integer values in various formats: ");
       System.out.printf("%d %(d %+d %05d\n", 5, -5, 5, 5);

       System.out.println();

       System.out.printf("Default floating-point format: %f\n", 9999999.99);
       System.out.printf("Floating-point with commas: %,f\n", 9999999.99);

       System.out.printf("Negative floating-point default: %,f\n", -9999999.99);
       System.out.printf("Negative floating-point option: %,(f\n", -9999999.99);
   }
}
```

Example 3: Display data with printf()

Output:

```
Displaying some numeric values in different formats.

Displaying integer values in various formats: 5 (5) +5 00005

Default floating-point format: 9999999.990000
Floating-point with commas: 9,999,999.990000
Negative floating-point default: -9,999,999.990000
Negative floating-point option: (9,999,999.990000)
```

As you can see in the output, `printf()` method can be used in the place of `println()` method for writing to the console whenever the formatted output is required. The `printf()` method can be called on `System.out`. It uses the `Formatter` class to format data.

`PrintStream` class in Java also defines the `format( )` method that works exactly like `printf( )` method.

END