COMPSCI 105

Principles of Computer Science

Command line arguments
Writing to files

Contents

Command line arguments

Starting a Java application from the command prompt:

```
C:\>java MyApplication
```

Using Command-line arguments

```
C:\>java MyApplication Hello World
```

In general

```
C:\>java ApplicationName [Parameter List]
```


main() method

Applications always start with the main method
- Passed an array of Strings
- The command line arguments

```
C:\>java MyApplication Hello World
```

```
public class MyApplication {
	public static void main(String[] args) {
		String first = args[0];
		String second = args[1];

		System.out.println(first);
		System.out.println(second);
	}
}
```

```
Hello
World
```

Exercise

Write a program called Ex1 which prints any command line parameters to standard output. For example, the following output should be produced if three parameters are specified on the command line:

```
C:\exercise>java Ex1 hello there world
hello there world
```


Exercise

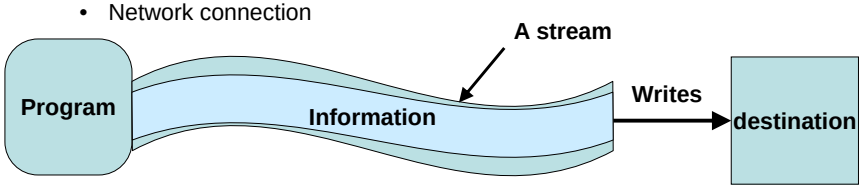
Write a class called "Display" that expects a single command line parameter to be given. It print out the contents of a text file with the given name.

```
C:\>java Display Test.txt
This is a small file
Of only three lines
Used to test some Java code
C:\>_
```


Output Stream

Stream of output data

- Console
- File
- Network connection



Algorithm

- open a stream
- while there is more data to write
  - write the next piece of data
- close the stream

FileWriter class

FileWriter

- Write a single character to a text file
- Write an entire String to a text file.

Translates characters

- From internal Java char
- To character encoding used by the file system
  - ASCII
  - UTF-8

Creating a FileWriter

```
FileWriter( File file ) throws FileNotFoundException

FileWriter( String fileName ) throws FileNotFoundException
```

FileWriter - `write()`

`public void write( int c ) throws IOException`

- Write a single character.

Parameters

- `c - int` specifying a character to be written

Throws

- `IOException` - If an I/O error occurs

FileWriter - `write()`

`public void write( String str ) throws IOException`

- Write a String

Parameters:

- `str - String` to be written

Throws:

- `IOException` - If an I/O error occurs

FileWriter - `close()`

`public void close() throws IOException`

- Close the stream, flushing it first.
- Once a stream has been closed, calls to write() or flush() will cause an IOException to be thrown.
- Closing a previously-closed stream has no effect.

Throws:

- `IOException` - If an I/O error occurs

FileWriter example

```
private void fileWriterEx01() {
      FileWriter fw;
      try {
            fw = new FileWriter( "output.txt" );
            fw.write( 72 ); //Code for 'H'
            fw.write( 'e' );
            fw.write( "llo \nout there!" );
            fw.close();
      }  catch( IOException e ) {
            System.out.println( e );
      }
}
```

BufferedWriter class

BufferedWriter

- Accessing the disk can be slow
- Writing one character at a time is slow
- Storing the characters in a buffer and writing a lot at the same time is much more efficient

Creating a BufferedWriter

```
BufferedWriter( Writer stream )
```

BufferedWriter methods

- Same as FileWriter
- `public void newLine()`
  - Write a line separator.

BufferedWriter example

```
private void bufferedWriterExample() {
      BufferedWriter bw;
      FileWriter fw;
      try {
            fw = new FileWriter( "output.txt" );
            bw = new BufferedWriter( fw );
            bw.write( 'H' );
            bw.write( 'e' );
            bw.write( "llo \nthere!" );
            bw.newLine();
            bw.write( "How are you today?" );
            bw.close();
       }  catch( IOException e ) {
            System.out.println( e );
       }
}
```


BufferedWriter example

```
private void bufferedWriterExample() {
      try {
            BufferedWriter bw = new BufferedWriter( new
                  FileWriter( "output.txt" ) );
            bw.write( 'H' );
            bw.write( 'e' );
            bw.write( "llo \nthere!" );
            bw.newLine();
            bw.write( "How are you today?" );
            bw.close();
       }  catch( IOException e ) {
            System.out.println( e );
       }
}
```


PrintWriter class

PrintWriter

- Similar to other Writer classes
- Additional methods for writing text

Creating a PrintWriter

```
PrintWriter( Writer stream )

PrintWriter( File f ) throws FileNotFoundException

PrintWriter( String filename ) throws FileNotFoundException
```

PrintWriter methods

`public void print ( String s )`

- outputs an entire String to a text file

`public void println( String s )`

- outputs an entire String to a text file, and then ends the line with a newline character

`public void println( any object type )`

- outputs a text representation of an object by calling the toString() method of the object type, and then ends the line with a newline character

`void println( any primitive type )`

- outputs a text representation of any primitive type (by converting it to a String) and then ends the line with a new line character

PrintWriter example

Using a PrintWriter to print output

```
private void printWriterExample() {
      boolean b = true;
      double d = 4.56;
      int num = 245;
      try {
            PrintWriter out = new PrintWriter(
                  new FileWriter( "output.txt" ));
            out.println( "PrintWriter!"  );
            out.print( b + " " );
            out.print( d );
            out.println(  " " + num + " Cool huh?"  );
            out.close();
      }  catch( IOException e ) {
            System.out.println(  e  );
      }
}
```


Input / Output Example

```
private void inputOutputExample() {
      try {
            BufferedReader in = new BufferedReader (new
                  FileReader("input.txt" ) );
            PrintWriter out = new PrintWriter( new
                  FileWriter( "output.txt" ) );
            String line;
            while ( (line = in.readLine()) != null ) {
                  out.println( line );
            }
            in.close();
            out.close();
      }  catch( IOException e ) {
            System.out.println(  e  );
      }
}
```


Exercise

Write a program called Ex3 which accepts an input filename and an output filename and creates the output file to be the same as the input file, with line numbers appended to the start of each line.

Call the program:

```
C:\exercise>java Ex3 in.txt out.txt
```

Input/Output Exercise

Input file (in.txt)

```
"There are two ways of constructing a software design;
one way is to make it so simple that there are obviously
no deficiencies, and the other way is to make it so
complicated that there are no obvious deficiencies. The
first method is far more difficult." - C. A. R. Hoare
```

Output file (out.txt)

```
1 : "There are two ways of constructing a software design;
2 : one way is to make it so simple that there are obviously
3 : no deficiencies, and the other way is to make it so
4 : complicated that there are no obvious deficiencies. The
5 : first method is far more difficult." - C. A. R. Hoare
```