S1182: Java and Eclipse for Beginner Part II- Exercises

Ex. 1 File Printer

In this exercise you will work with a slightly more complex program that prints out the contents of the files specified on its command line.

1) Open the FilePrinter.java file which is in the FilePrinter folder.

2) The program has errors (the red icons in the left hand margin). 'Hover' the mouse over the
The error icons and observe the message 'Unhandled exception type FileNotFoundException' and 'Unhandled exceptions type IOException'

4) Now we are going to modify the program so that it handles these exceptions.

Locate the displayFile method within the source code and examine the code within it. There are three places where errors may occur:

☒ when trying to open the file:
```
fileReader = new FileReader(fileName);  // may throw a FileNotFoundException
```

☒ when trying to read from the file:
```
while ( (c = fileReader.read()) != -1) {  // may throw an IOException
```

☒ and when closing the file once we have finished with it:
```
fileReader.close();  // may throw an IOException
```

5) Surround the code which may throw exceptions with a try catch block.

6) First add a catch block for the FileNotFoundException:
```
catch ( FileNotFoundException fnfEx ) {
    // display error message here
}
```
within the body of the catch block, display an error message saying that the file could not be opened - include the name of the file in the message.

7) Now add a catch block for the IOException:
```
catch ( IOException ioEx ) {
  System.out.println( "Error reading from file." );
}
```

8) Our program should always close any file handles it has open, regardless of what happens. This is a good choice for a finally clause. Create a finally block that closes the file if it is still open:

```
finally {
  if (fileReader != null) {
    try { fileReader.close(); }
    catch (IOException ioEx) { ; } // nothing we can do now!
  }
}
```

The call to close the file has been moved from inside the try block you created earlier to the finally block. The test of fileReader ensures that we only close a file that we previously opened. Note that the call to close the file can itself throw an exception, and this example illustrates the ability to nest try-catch blocks.

9) What happens if the fileReader.close() was left in the main try catch block ?

Ex. 2 Adapting Command Line Arguments

In a previous exercise we looked at how to print out all the arguments passed to the program
In this exercise we are going to adapt this program to test if one of the many arguments passed in contain a specific value, which can be anywhere in the list.

We have just been looking at how we could use the collections classes. In this exercise we are going to use ArrayList.

1) Open the CommandLine.java file which is in the CommandLine folder.

2) Import the collection classes.
This is done by adding the following to the top of your program

```
import java.util.*;
```

3) Declare a ArrayList();
Just below the "public static void main……" add:

```
List theArguments = new ArrayList();
```

4) Remove the println statement inside the for loop, as it is no longer necessary
Delete the following:

```
System.out.println("Argument " + i + " = " + args[i]);
```

5) Inside the for loop we need to add every argument to the ArrayList
Inside the for loop, just below the for statement add:

```
theArguments.add(args[i]);
```

6) Print out a message if the arguments included the word "stop" using the contains(..) method
Just before the end of the main method add:

```
if (theArguments.contains("stop")) {
    System.out.println("it contains \"stop\"");
}
```

7) Save your program.

Now we are going to run the program with some arguments.

8) To attach arguments click the arrow on the "Run" icon  and select Run. A dialog box appears displaying the application's *runtime environment*. Ensure that CommandLine is highlighted (under the Java Application folder). If it does not exist, then select Java Application

and click New. A new runtime configuration will appear for the CommandLine application.

9) Click the Arguments tab and enter the following Albert Einstein quote into the Program arguments text area.

The important thing is not to stop questioning.

By entering information here, you are effectively appending to the Java runtime command line. This information will be passed to the application and placed in the args array.
Click Run to start running the application.

10) Check the console view and ensure that “it contains stop”.

11) Optional.
Insert code just before the end of the main method to print out the last element of theArguments.

```
System.out.println("Last element: " +
theArguments.get(theArguments.size() -1));
```

12) Optional.

Think about how you could do

```
List theArguments = new ArrayList();
if (theArguments.contains(“stop”)){
```

if you had an array rather than a ArrayList

S1182: Java from the very beginning Part II - Sample Solutions

Ex. 1 File Printer

```
import java.io.*; // include Java's IO library

/**
 * FilePrinter application - displays the contents of the file or files passed
on
 * the command line to standard out.
 */
class FilePrinter {

  /**
   * Main entry point - display the contents of each file
   */
  public static void main(String[] args) {

    for (int f=0; f < args.length; f++) {
      displayFile(args[f]);
    }

  } // end of main method


  /**
   * Pretty print the contents of a file to the screen and handle any
   * exceptions
   */
  private static void displayFile(String fileName) {

    System.out.println(fileName + ":");
    System.out.println();

    FileReader fileReader = null;
    // declare outside of the scope of the try block
    try {
      fileReader = new FileReader(fileName);
      int c;
      while ( (c = fileReader.read()) != -1) {
        System.out.print((char)c);
      }
    }
    catch (FileNotFoundException notFoundEx) {
      System.out.println("Could not open " + fileName);
    }
    catch (IOException ioEx) {
      System.out.println("Error reading from " + fileName);
```

```
    }
    finally {
     if (fileReader != null) {
      try { fileReader.close(); }
      catch (IOException ioEx) { ; } // nothing we can do now!
     }
    }

  } // end of displayFile method


} // end of class
```

Ex. 2 Using ArrayList

```
import java.util.*;
import java.io.*;

/**
 * A Java applications to check if the arguments contain a specific value
 */
public class CommandLine {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		List theArguments = new ArrayList();
		// is equivalent to
		// List <Object> theArguments = new ArrayList <Object> ();
		// Create an ArrayList of String
		// List <String> theArguments = new ArrayList <String> ();

		if (args.length > 0) {
			for (int i=0;i<args.length;i++) {
				// Java 5 uses generics. Unless the content type
is specified there will be a warning
				theArguments.add(args[i]);
			}
		}
		else {
			System.out.println ("You didn't provide any
arguments");
		}
```

```
            // Iterate through the collection
            for (Iterator it=theArguments.iterator(); it.hasNext(); ) {
            Object element = it.next();
                  System.out.println (element);
            }

            if (theArguments.contains ("stop")) {
                  System.out.println("it contains \"stop\"");
            }


            try {
               System.out.println("Last element: " +
theArguments.get(theArguments.size()-1));
            } catch (java.lang.ArrayIndexOutOfBoundsException e) {
                  System.out.println("Accessing array with: " +
(theArguments.size()-1));
                  e.printStackTrace();
            }
      }
}
```