**Further Object- Oriented Programming**

U08026: Week 4

Assertions and Exceptions

David Lightfoot

Department of Computing and Communication Technologies

# Agenda

## In these exercises, you will

## use assertions to deal with *pre-* and *postconditions*.

## use *exceptions* to deal with unexpected behaviour.

# Part 1: Assertions

# Exercise 1

## Study this program and try to work out what *what* does.

package assertions;

/\*\*

\*

\* @author David Lightfoot 2013-10-09

\*/

public class Assertions {

public static double what(String[] s) { *// Note: might be erroneous!*

*// precondition assertions go here*

int sum = 0;

double m;

for (int i = 0; i != s.length; i++) {

sum += s[i].length();

}

m = sum / s.length;

*// postcondition assertions go here*

return m;

}

public static void main(String[] args) {

String[] sample1 = {"David", "David", "Bedour", "Chris", "Paul"};

String[] sample2 = {"", "", "", "", ""};

String[] sample3 = null;

String[] sample4 = {"David", "David", "Bedour", "Chris", "Paul"};

sample4[3] = null;

String[] sample5 = {};

String[] sample6 = {"David", "David", "Bedour", "Chris", "Younas"};

String[] sample7 = {"David", "David", "Bedour", "Chris", "Younas",

"Sharon", "Nigel"};

// use this to check that assertion checking is switched on

assert(false): "my test false assertion worked"; // 0

// System.out.println(what(sample1)); // 1

// System.out.println(what(sample2)); // 2

// System.out.println(what(sample3)); // 3

// System.out.println(what(sample4)); // 4

// System.out.println(what(sample5)); // 5

// System.out.println(what(sample6)); // 6

// System.out.println(what(sample7)); // 7

}

}

# Exercise 2

## Copy the program into a new Java project and run it.

## Nothing happens. That’s right! Assertion checking is off by default in *NetBeans*.

# Exercise 3

## Enable assertion checking for the project (see this week’s *Powerpoint* slides). Note that this setting will persist after this *NetBeans* session.

## Run the program again.

## You should see something like:

run:

Exception in thread "main" java.lang.AssertionError: my test false assertion worked

at assertions.Assertions.main(Assertions.java:36)

Java Result: 1

BUILD SUCCESSFUL (total time: 0 seconds)

## Note that *my* message “my test false” has been displayed.

# Note on terminology

## In the following exercises, when I say

## **reveal line *n***, I mean remove comment markers // from line *n*

## and **hide** other numbered lines, by adding comment markers //.

# Exercise 4

## Reveal line **1** (don’t forget to hide line 0).

## Run the program again.

## You should see something like:

run:

5.0

BUILD SUCCESSFUL (total time: 0 seconds)

# Exercise 5

## Reveal line **2** (don’t forget to hide line 1).

## Run the program again.

## Still OK? Is it the right answer?

# Exercise 6

## Reveal line **3** (don’t forget to hide line 2).

## Run the program again.

## Whoops! What happened?

## What had we expected to be true? This is a precondition of *what*.

## Add an assertion (*assert* statement) to *what* to assert this precondition.

## Include a message in the assertion which makes it clear that the assertion was created by you (make it *fanciful*; this to make it clear that the message was not generated by the Java system itself).

## Run the program again.

# Exercise 7

## Reveal line **4** (don’t forget to hide line 3).

## Run the program again.

## Whoops! What happened now?

## What had we expected to be true? This is another precondition of *what*.

## Add program statements to apply multiple assertions to *what* to assert this precondition.

## Run the program again.

# Exercise 8

## Reveal line **5** (don’t forget to hide line 4).

## Run the program again.

## Whoops! What happened this time?

## What had we expected to be true? This is yet another precondition of *what*.

## Add program statements to apply a further assertion to *what* to assert this precondition.

# Exercise 9

## It is a *postcondition* of *what* that *m* multiplied by *s.length* should be equal to *sum*.

## Reveal line **6** (don’t forget to hide line 5).

## Run the program again.

## Is everything OK?

## Add an assertion just before the *return* statement to assert this postcondition.

## Run the program again.

## What happened?

## What is wrong with *what*?

## Correct the error in *what* and run again.

## The postcondition assertion should now be true and no message will be emitted.

# Exercise 10

## Reveal line **7** (don’t forget to hide line 6).

## Run the program again.

## All OK?

## **For submission – Part 1 of 2:**

## Copy the text of your final version of *what* with all its *assert* statements and the *correction* into a *Word* file ready for submission.

# Part 2: Exceptions

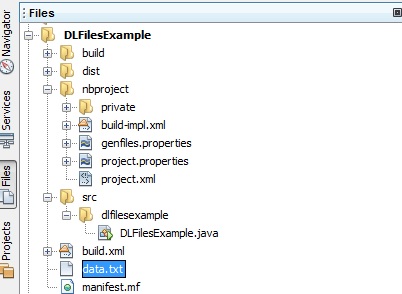
# Exercise 11

## When trying to open a file for reading, it is possible that a file by that name does not exist, or is in the wrong place, or is not available for access. We must try to open a file stream and catch any exceptional circumstances.

## Make a project and copy in the program text below.

## You will find that the program produces an error message as the fact that there is no file called data.txt is caught and reported.

## Use *NetBeans* to create and empty file; name it *data.txt* and save it under the project, as in:



package dlfilesexample;

/\*\*

\*

\* @author DLightfoot

\*/

import java.io.\*;

class DLFilesExample {

public static FileInputStream openStream(String fileName)

throws FileNotFoundException {

FileInputStream fis = new FileInputStream(fileName);

System.out.println("File input stream created");

return fis;

}

public static void main(String args[]) {

FileInputStream fileInputStream = null;

String fileName = "data.txt"; // 0

// String fileName = "foo.bar"; // 1

// String fileName = null; // 2

System.out.println("Starting with file name = " + fileName);

// attempt to get file input stream

try {

fileInputStream = openStream(fileName);

}

catch (FileNotFoundException ex){

System.out.println("Whoops: FileNotFoundException caught: no such file");

}

}

}

## Run the program again. You should get the message:

Starting with file name = data.txt

File input stream created

## Reveal line **1** (don’t forget to hide line 0).

## Reveal line 1.

## Run the program again. You should get the message:

Starting with file name = foo.bar

Whoops: FileNotFoundException caught: no such file

# Exercise 12

## Reveal line **2** (don’t forget to hide line 1).

## Run the program again. You should see something like this:

Starting with file name = null

Exception in thread "main" java.lang.NullPointerException

at java.io.FileInputStream.<init>(FileInputStream.java:134)

at java.io.FileInputStream.<init>(FileInputStream.java:97)

at dlfilesexample.DLFilesExample.openStream(DLFilesExample.java:14)

at dlfilesexample.DLFilesExample.main(DLFilesExample.java:31)

## Java Result: 1

## Add a further *catch* clause to the main program to catch any exception (of class *Exception*) and give a suitable message.

Run the program again. You should the see the result from catching the null-pointer exception.

## **For submission – Part 2 of 2:**

Copy the text of your main program into the *Word* file you used for Exercise 10.

**Also:** Zip the contents of the files you used for the *assertions* part and those for the *exceptions* part separately and submit along with the single *Word* file.