Lab 1A – Exploratory Testing

# Lab Intro

This lab will introduce you to the concept of **Exploratory Testing**. You will download the supplied zip file on Moodle and import the zip folder into the Integrated Development Environment we will be using for this unit (IDE).

**Key Term – Exploratory Testing**Testing software by freely exploring it, like an adventure, instead of following a set script. Testers use their intuition and creativity to find problems.



## Learning Outcomes

By the end of this session, you will be able to:

* Understand the fundamental principles of exploratory testing
* Explore the benefits and limitations of exploratory testing in real-world scenarios

## Resources

* **Lab1A-Apps.zip** – available on Moodle

## Lab Setup

* Create a new directory for this unit (ideally on your W: drive) and give it a sensible name (e.g. “STQA”), and a sub-folder called “Week 1”.
* Download and extract **Lab1A-Apps.zip** to your newly created Week 1 subfolder
* Open **IntelliJ** and select “Open”and select the Lab1A-Apps folder (See Appendix A for help if you could not attend the lab session).
* You should then see the Project pane on the left-hand side with four packages in the **src** folder – each of these packages map to an exercise in this lab sheet.

# Exercise 1 – SimpleCalculator.java

The first exercise is a deceptively simple calculator application. Run the **SimpleCalculator.java** class in the ex1 package and you should see the GUI presented in Figure 1.

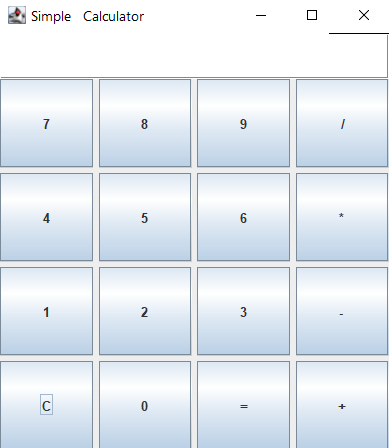


Figure - SimpleCalculator.java Program

Have at it – look for, and briefly list below – every defect you believe you have found with this program. **Note**: Although the list below contains 10 numbered items, that does not mean there are exactly 10 defects in the program – there could be more or less.

|  |
| --- |
| 1. No error handling. 2. UI inst fully 3. Code duplication 4. Operate precedence handling. for example, BIDMAS 7\*5+3 5. Doesn’t operate negative number, right? 6. Doesn’t the old result or display then on the screen 7. = straight away generates an exception 8. You can’t do anything with 9. Too many spaces with tittle of the application |

# Exercise 2 – NoteApp.java

The second program is a very simple note-taking application. Run NoteApp.java and test away.

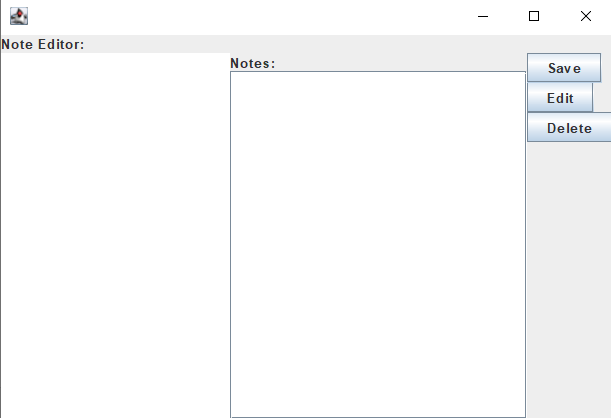
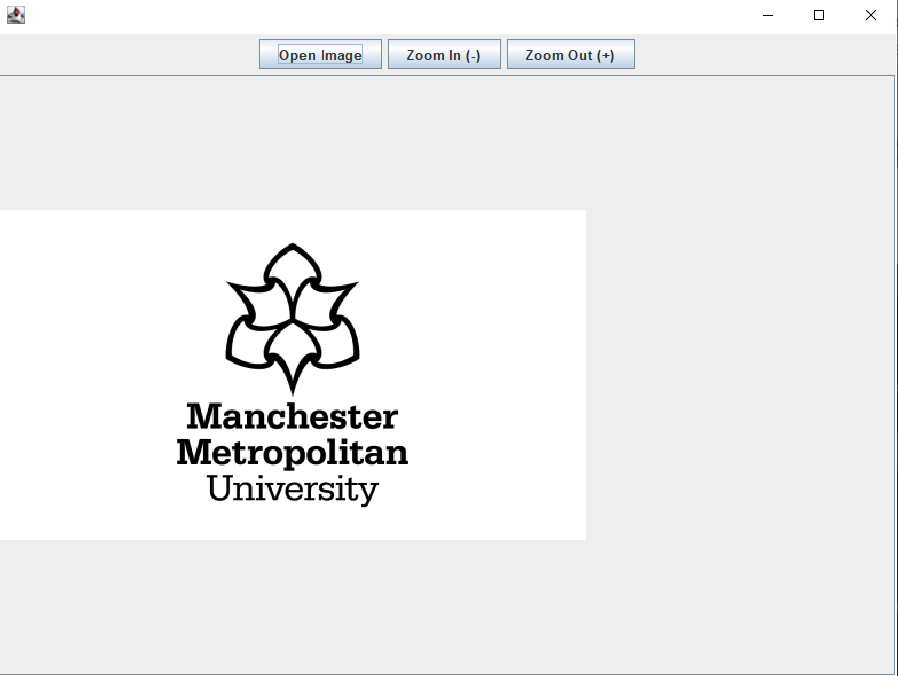


Figure 2 NoteApp.java Program

|  |
| --- |
| 1. UI isn’t user friendly 2. Edit button doestn actually do any thing 3. Not application tittle 4. Adding note it’s a bit difficult when you already have few note |

# Exercise 3 – ImageViewer.java

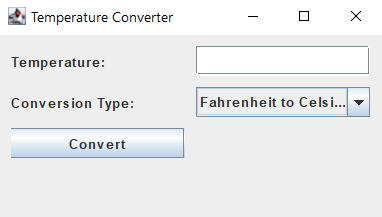
The third program allows a user to select an image from the file system to view it and zoom in/out. Test the application and document the bugs you find below. You should source your own images to test this application.



|  |
| --- |
| 1. After zooming out and gives me exception 2. After an image and you change a new image, its doesn’t return to the normal size just stay zoomed in 3. Not application name |

# Exercise 4 – TemperatureConverter.java

Open TemperatureConverter.java and conduct testing on the program. You may need to look at the code here to perform an analysis of the logic involved. Don’t worry if you are not a Java programmer, the code is very intuitive in terms of where the calculations are being performed.



|  |
| --- |
|  |

# Lab Summary

We have explored some of the simplest applications possible in this lab to get a feel that even deceptively simple programs can be littered with defects or things that could easily lead to something being labelled as a defect in the future. In the next lab, we will associate defects with actual use cases of the applications so that the defects discovered are clearly aligned with the expected functionality of the applications we are testing.

# Appendix A – IntelliJ Project Setup

