

BSc (Hons) Web and Mobile Development

Student Number: S211621

Introduction to Programming Report

Module Tutor for Introduction to Programming: **Frank Carver**

Covid-19 Contact Tracer Application Using Java Language

Course Leader: **Sean Preston**

Word Count: 2092

Table of Contents

Contents

1. Introduction	3
Java History.....	3
2.Application Development	4
Creating an Eclipse Project.....	4
Interface.....	Err
or! Bookmark not defined.	10
Part 1.....	4-10
Part 2.....	4-10
3.GUI (Graphical User Interface).....	11
Part 3.....	11
4.Conclusion.....	12
5.Bibliography.....	12-13

1. Introduction

The report determines the development of a COVID-19 tracing system, which will allow university medical staff to give any student a COVID-19 test result into a university. This will be created into a COVID-19 tracing database for future records. On receiving a positive test report, the system should be able to find all other students who have been in close contact with the concerned student during their lecture sessions so they can be notified to self-isolate via email. The objective is to take the COVID-19 (also known as SARS-COV-2 or just "coronavirus") pandemic and ensure that all students are tested regularly.

The ongoing disease outbreak which began in 2019. Pneumonia of unknown cause china (WHO Organization, n.d.),

When universities decide to resume on-campus teaching, it is important to keep records of the students and staff who have been told to self-isolate. The Office for Students (OfS) is working with the Department for Education and with universities and colleges to ensure you have the information and guidance you need. As you know, this is a fast-developing situation. Our aim is to keep you, and the universities and colleges we regulate, as up to date as possible. (Office for Students, 2021)

The tracing system will be developed by Java Language, this is a programming language and javac is its compiler. (javac, 1993) Java is one of the most common, in-demand computer programming languages used today. (Veeraraghavan, 2021).

For this particular application, the software Eclipse IDE (Integrated Development Environment) has been used to write the application code. The Eclipse IDE is famous for our Java Integrated Development Environment, but we have a number id pretty cool IDE's, including out C/C++ IDE, JavaScript/TypeScript IDE, PHP IDE, and more. (Anon., 2021)

The appropriate source code produced in Java files is used to support any information which can determine the input and outputs of the program, also documented so that future developers can understand and maintain the application.

History of Java

James Gosling pioneered Java in June 1991 as a project called 'Oak.' Gosling aimed to develop a virtual machine and language with a well-known notation like C, but with more precision and simplicity than C/C++. In 1995, Java 1.0 was the first public execution. It pledged 'Write Once, Run Anywhere' on popular platforms with free runtimes. It was very safe and configurable with security that restricted network and file access. In a stable 'applet' setup, the significant web browsers soon implemented it in their standard settings.

In 1997, Sun reached out to the ISO/IEC JTC1 and then Ecma International to formalize Java, but they quickly withdrew. Java continues to be a de facto proprietary standard regulated by the Java Community Process. With the revenue generated by new vision such as the Java Enterprise Framework, Sun made several Java implementations free of charge. The critical difference is that the compiler is not present in the JRE, which differentiates between its Software Development Kit (SDK) and JRE (JRE). (Sarangam, 2021)

2.Application Development

This is where the logic of the application is centralized for this particular assignment. Based on the eclipse software system the design is created with an object-oriented perspective, which includes functions such as classes, interfaces, data types (e.g maps), input and output streams, exception handling, JUnit tests, and CSV files (these will all be explained throughout the report).

Creating an Eclipse Project

Eclipse is known to be an integrated development environment, which helps write Java program code, where a particular layout of folders in part of the assignment consists of the project folder, packages, and classes.

This is called a build path as it has a source folder and the source folder is where the main Java code will go, and where the packages are created. Packages are usually named after their domain reverse domain name., so in this case **uk.ac.uos**. The website for the university is uos.ac.uk and it is written backward.

Once the packages have been created, an interface class can be implemented – classes are normally placed inside.

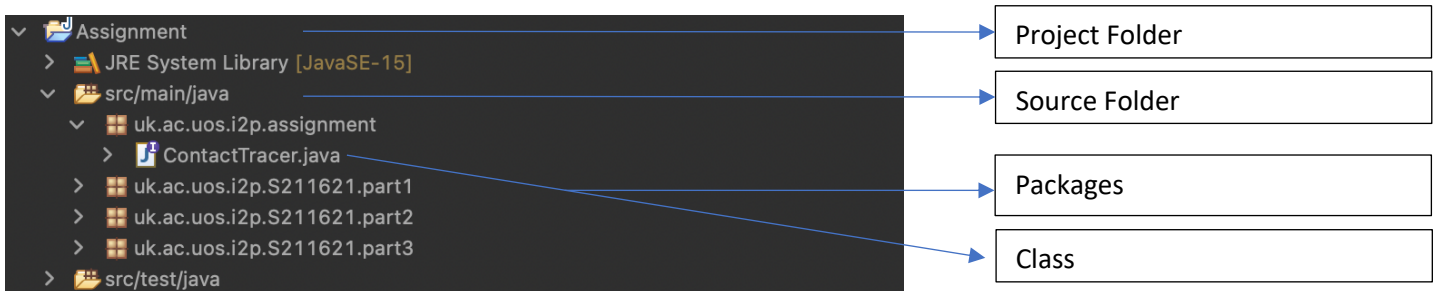


Figure 1- Shows the project hierarchy

Interface

Part 1 of the project requirements is to build a Java library.

The library will manage all the students and courses at the university and link the students to the course they are enrolled in. The object of the interface is called the Contact Tracer and a pre-written code in this class defines all the subclasses from the interface that will be used and tested later on. The diagram below shows the usage of parameters, variables, and attributes that play a major role in writing this application as each one does different things.

e.g String (with a capital S) is a type class is known as a sequence of characters that can exist inside an object class.

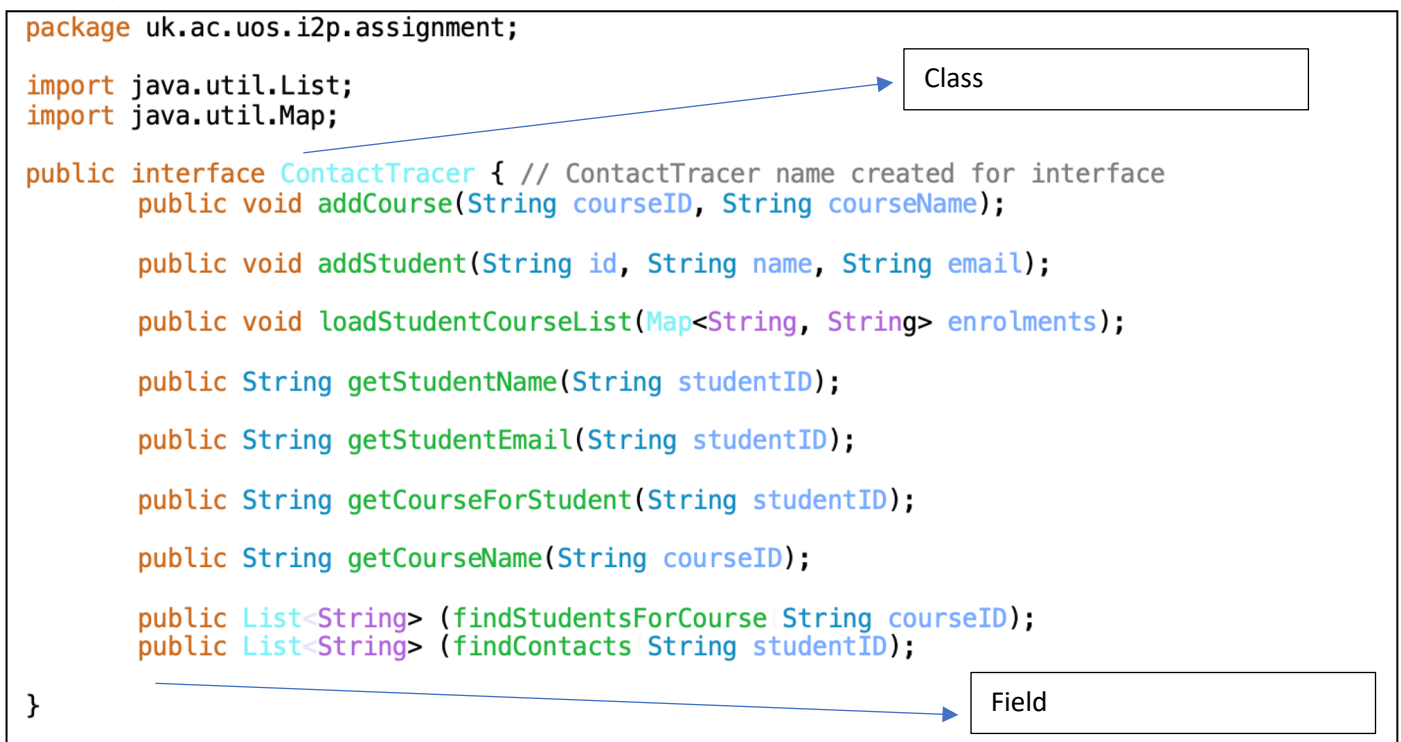


Figure 2 - Shows the pre-written Interface

The way to store code in a class is to put add in what are called fields. The fields are code written which is visually seen in a class. They're also called class and members sometimes, and people might talk about public members and private members however, they all live in the same place - *by zombies*.

The first aspect of the interface is all to do with putting data in and getting data out of the class. And the second aspect is all to do with what the data is doing.

The above methods in the pre-written interface will do the following:

addCourse and getCourseName - Input the course id and course name will add to the library

addStudent and getStudentName and getStudentEmail – two get methods have created as we require the student name and email address - Output will be student name and student email address

loadStudentCourseList – Output will be student courses

findStudentsForCourse – Output will be students enrolled in a particular course

findContacts – Output will be the contact details of the student enrolled

The Memory Contact Tracer class implements from the Contact Tracer are taken to give a certain result from the code written. Inbuilt collections such as HashMap, Map, Array, and List have been imported for the stand-alone application.

```
package uk.ac.uos.i2p.S211621.part1;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import uk.ac.uos.i2p.assignment.ContactTracer;

public class MemoryContactTracer implements ContactTracer {

    private Map<String, Course> courses = new HashMap<>();
    private Map<String, Student> students = new HashMap<>();

    private Map<String, String> enrolments = new HashMap<>();
}
```




Figure 3 - MemoryContactTracer class with Implements

MemoryContactTracer class is created, as a result, the code is much easier to read and much easier to write. This is because when implementing an interface, information taken from the ContactTracer class can determine a specific type that can be used to test the implements of the interface.

For example – HashMap is being used to store the database information, the collection of Map object is used to obtain a list that maps keys to each value as each key is linked to a specific value.

The List collection was previously used however, it made the code very complicated due to the fact a Map can store the keys and values, which allows for the value to be mapped by a key whereas, the List just creates a list of items and for this application. The Map is easier to perform a search or lookup by using the keys to gain information.

```
@Override
public void addCourse(String courseId, String courseName) {
    courses.put(courseID, new Course courseID, courseName));
}

@Override
public void addStudent(String id, String name, String email) {
    students.put(id, new Student id, name, email));
}

@Override
public void loadStudentCourseList(Map<String, String> studentEnrolments) {
    this.enrolments.putAll(studentEnrolments);
}

@Override
public String getStudentName(String studentID) {
    return students.get studentID .getName();
}

@Override
public String getStudentEmail(String studentID) {
    return students.get(studentID .getEmailId());
}
```

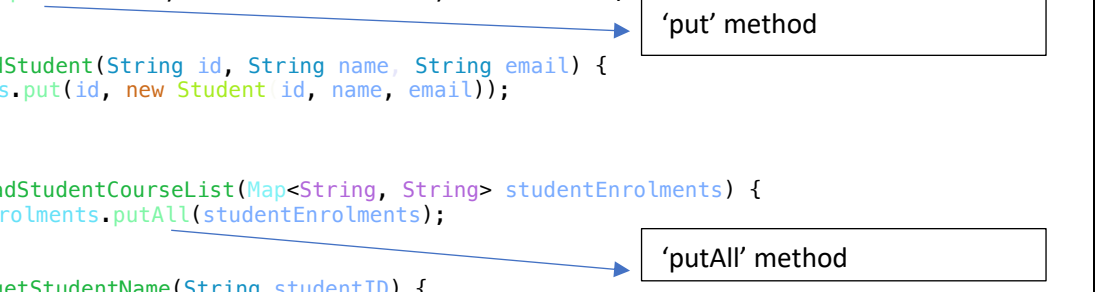




Figure 4 - Fields created using the methods of the ContactTracer Library

After creating the field, different methods are added for the application to work – certain methods and statements have to be implemented. (as shown in Figure.4)

The **'put'** method is added so that the getter method will work for the interface to retrieve information from the ContactTracer and by using the getter methods, it will display the student details.

The **'putAll'** method is added so that the Map class can take all the elements from the ContactTracer class.

The **'return'** statement is added to each of the fields so that that value can be returned (in this case so the output result can be true), this is because only one return statement can be executed at a time.

If more return statements were added then the return statement value would be false.

The aim of **'findContacts'** is to return the list of studentID, who share a class with a supplied studentID.

The studentID is to search until it returns a list of other studentID where it then will create a list, hence there is an ArrayList class.

ArrayList Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.) (Oracle, n.d.)

[Part 2](#) of the project is testing the output of the written code.

The above implements of the project have been created, and are now ready to be tested.

The tests are there to make sure that the written code does the right thing as well as sticking to the interface.

When creating the initial main source folder, a test folder is also created the same way. Inside the test folder, a ContactTracerTest package is created. It's always good to call the package "test", so that it's clear what it is, and that it doesn't accidentally overlap with any of the other packages in the project system.

All tests must be created under the same name in a Junit Test case, for the interface that requires testing.

The package has imports at the top assertions and testings, however, a method has been created. This method shows it has got an AP test (**@test**) annotation that states that it is a test.



Figure 4 - shows the implemented code for testing

The above shows the same collections that are in the MemoryContactTracer – this is because when tested the results will link up to the information which has been provided in each class. A new collection has also be imported, this is called the Assertions. To test the application, an assertion statement is required to prove that the correct assertion is executed.

In Java, to check if a code is running properly certain tests should be done and for this, a feature in the IDE called JUnit testing is used. JUnit 5 is out the door as the next-generation test framework. It is a fundamentally redesigned version of the most widely used testing library in Java. JUnit 4.0 was first released over a decade ago after the introduction of annotations in Java 5. (Eclipse, 2017)

Before testing the code, and to work properly the main testing code is written in the method.

ContactTracer and MemoryContactTracer classes are pulled into the ContactTracerTest class, to ensure the test is loading properly, this shows that the code is written is correct by creating an object. This sets the data up, and an assertion to check that the data was correctly set up. The value of the data helps by taking information from the key.

Figure 5, shows the first test which is loaded and has passed testAddCourse, and only one assertEquals line is added.

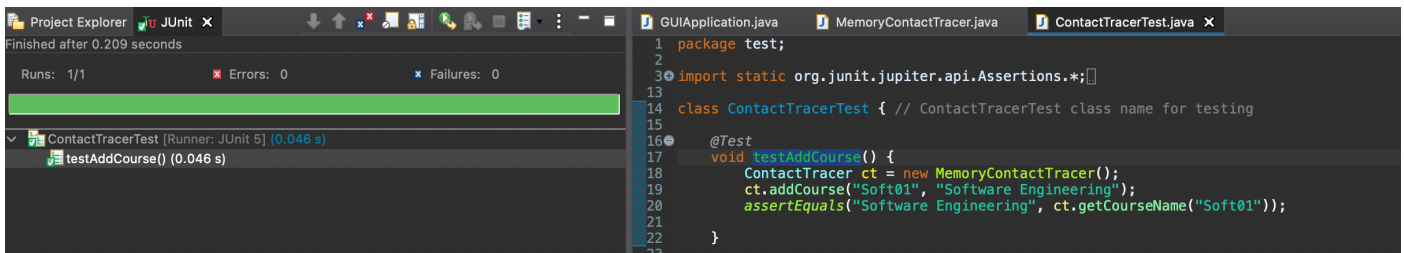


Figure 5 – shows the first test

This is because it is only searching for one-course name and course id (Software Engineer/Soft01). If the test was to fail a red line would come up instead of the green line.

Each test created is thoroughly thought out, and as shown in Figure 6, it starts to get more complicated, and more code is written to test different sections of the stand-alone application.

Figure 6 also shows that each test is getting bigger and bigger, and this is because more details have been inputted e.g course names, course id, student names, student id and student details such as email address.

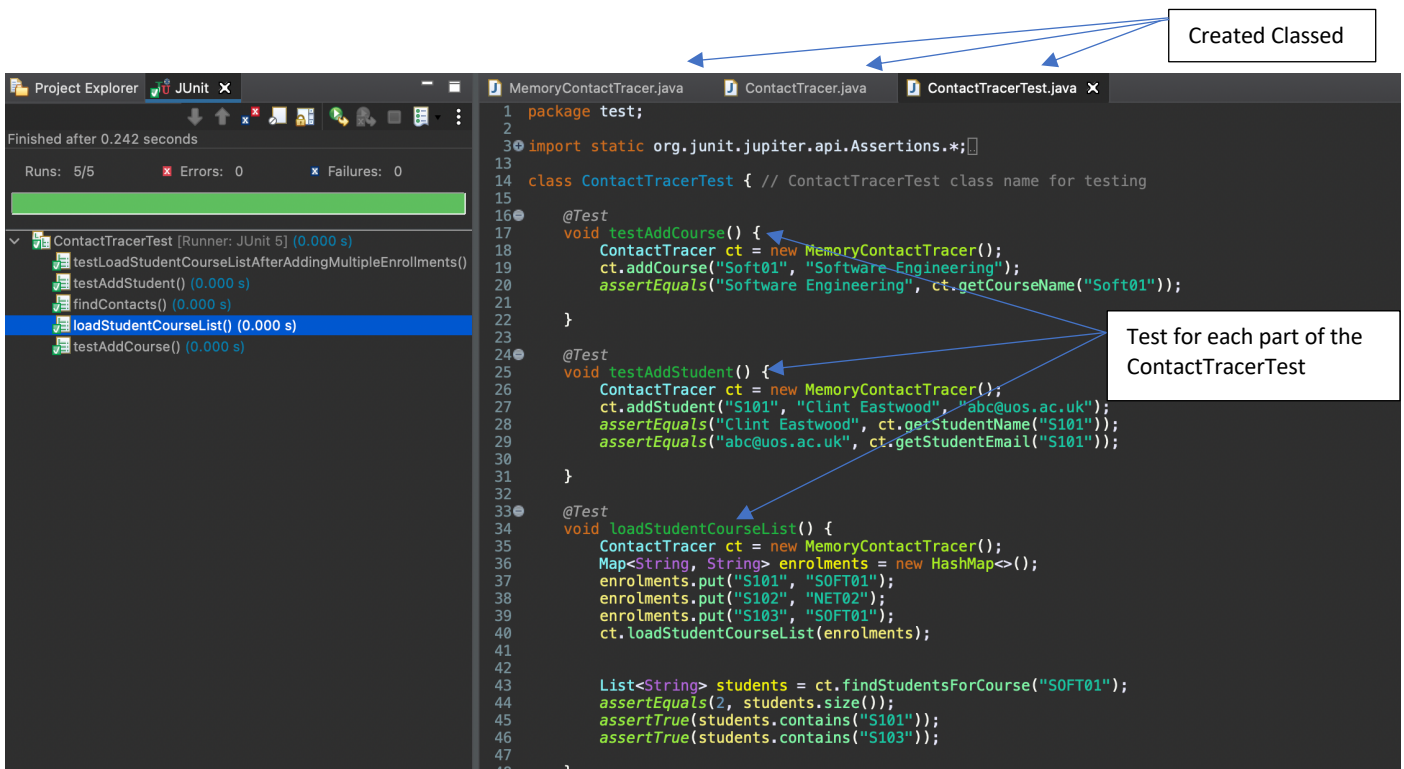


Figure 6 – shows test for each part of the ContactTracerTest

All the tests have been completed, and all successful a Command Line Interface needs to be run.

A Command Line Application from Eclipse is set it up so that it can run with different command line parameters. At present, the application is not actually doing anything and it needs to run to display output results. A print method is created to print out the arguments that are required to run the application, and this will be shown further along in the report. For this process to start another class is created, which is called the CommanLineApplication, as shown in Figure 7.

Command Line Arguments is part of a Java application that can accept any number of arguments from the command line. This allows the user to specify configuration information when the application is launched. (Oracle, n.d.)

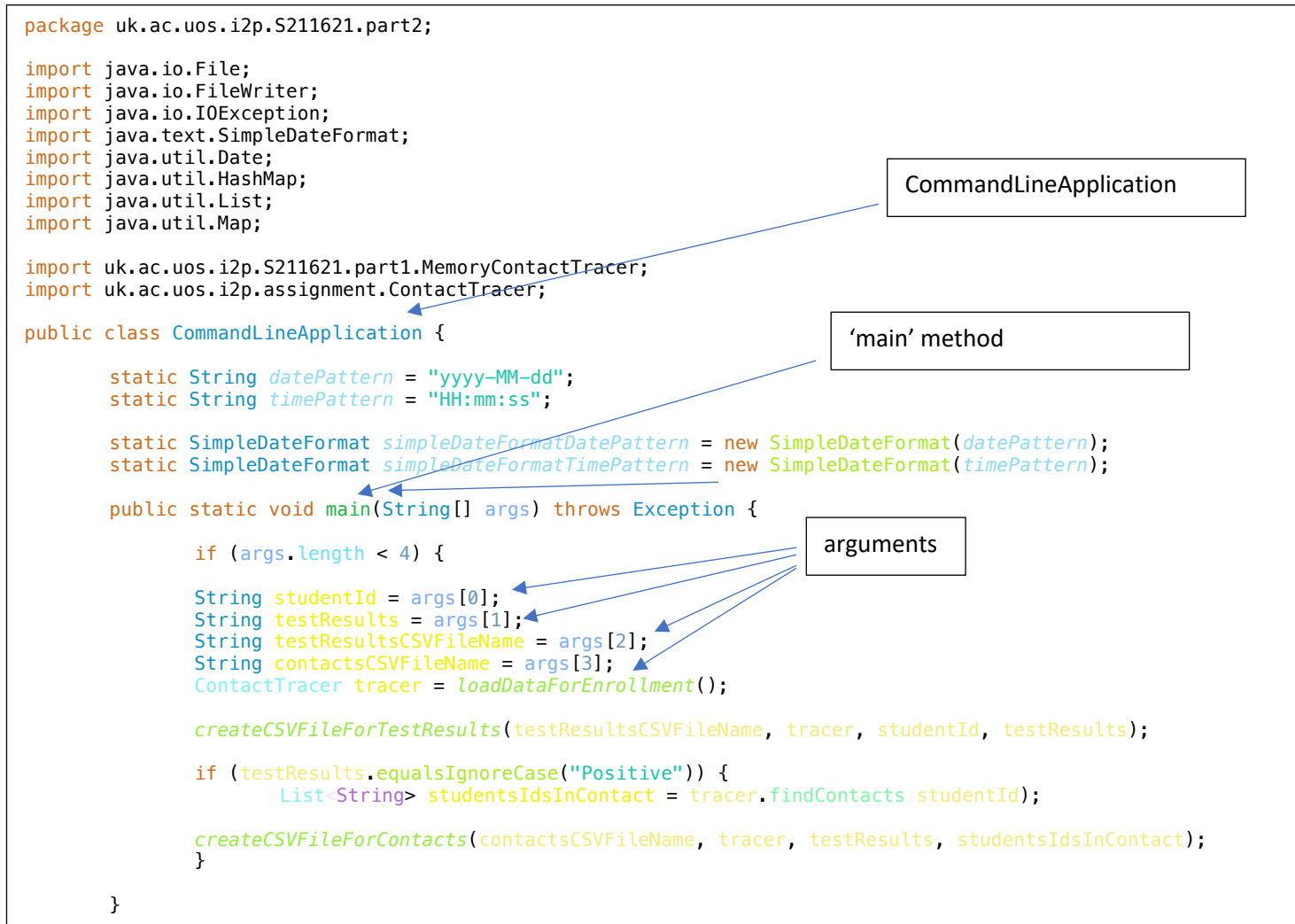


Figure 7 – shows the CL class and arguments

Inside the CommandLineApplication class, the main method passes the command line arguments. Arguments are words that are supplied after the class name as an array of Strings. In this case, the array length will still be checked. If the name is provided it is put into a variable with a CSV(comma separated values)file. This may seem complicated but, when an argument is created (in this case 4 args) those will be the outputs that will be displayed on a command line.

Figure 7 shows the necessary arguments that are required, so in this class, the following outputs will be displayed in the CSV file:

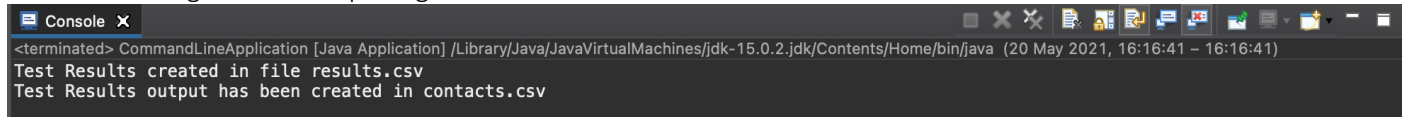
student ID
testResult

These will be placed into the CSV files called:

testResultsCSVFileName
contactsCSVFilename

The output will look like this – Clint Eastwood, S101, Positive, 2021-03-16, 17:28:55 (Figure 8)

These arguments will be shown inside a CSV file because if it is run as a java application the output on the console will show something different as per Figure 8.

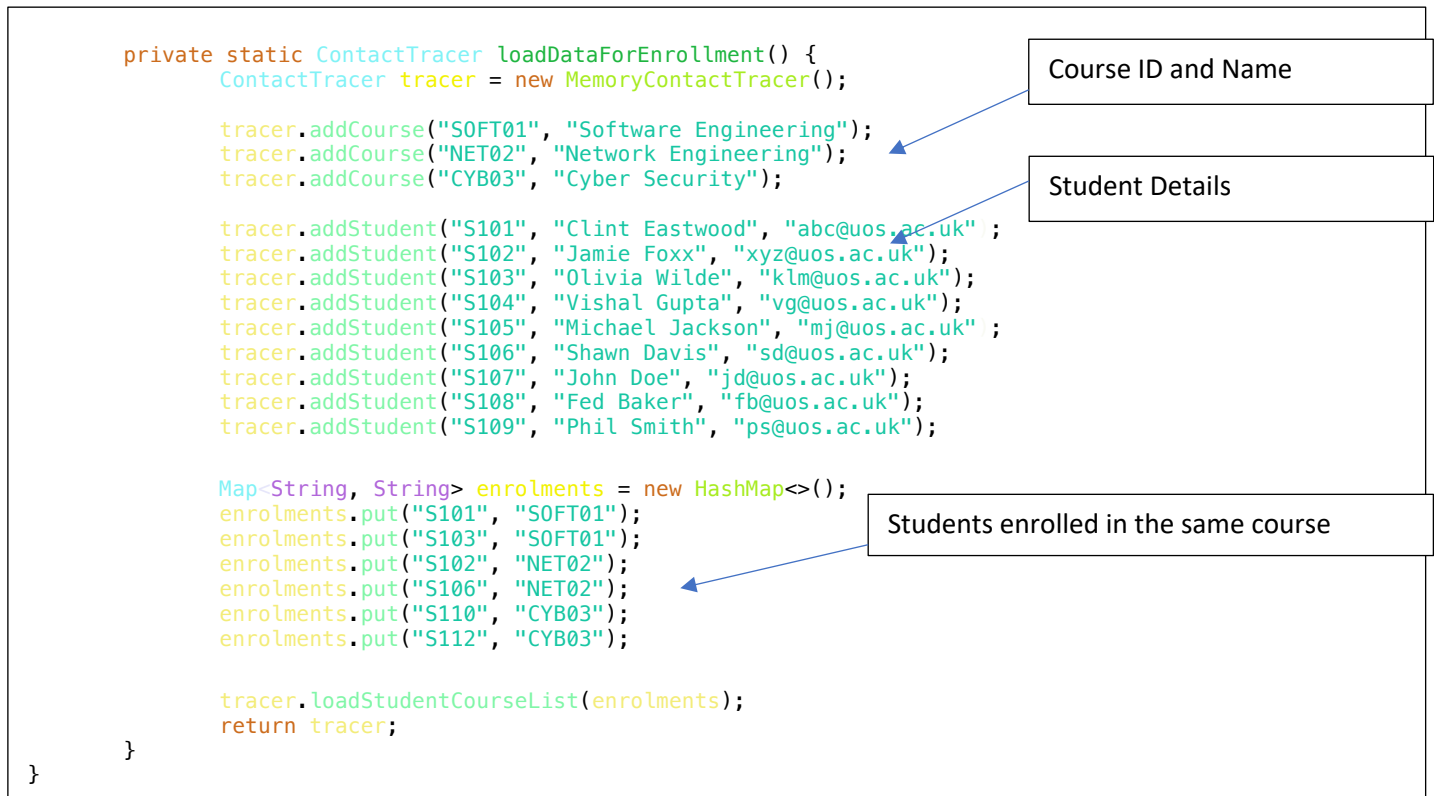


```
Console X
<terminated> CommandLineApplication [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.2.jdk/Contents/Home/bin/java (20 May 2021, 16:16:41 - 16:16:41)
Test Results created in file results.csv
Test Results output has been created in contacts.csv
```

Figure 8 – shows output of console

The difference between a command line application and a console application is that in a command line application the result can be viewed in separate CSV files where a console application, the output will just remain on the console of the screen.

The next step will be to add the actual data into the CSV files. This will be done by taking the data again from the ContactTracer and MemoryTracer, but this time the details for the output are added as shown in Figure 9.



```
private static ContactTracer loadDataForEnrollment() {
    ContactTracer tracer = new MemoryContactTracer();

    tracer.addCourse("SOFT01", "Software Engineering");
    tracer.addCourse("NET02", "Network Engineering");
    tracer.addCourse("CYB03", "Cyber Security");

    tracer.addStudent("S101", "Clint Eastwood", "abc@uos.ac.uk" ;
    tracer.addStudent("S102", "Jamie Foxx", "xyz@uos.ac.uk");
    tracer.addStudent("S103", "Olivia Wilde", "klm@uos.ac.uk");
    tracer.addStudent("S104", "Vishal Gupta", "vg@uos.ac.uk");
    tracer.addStudent("S105", "Michael Jackson", "mj@uos.ac.uk" ;
    tracer.addStudent("S106", "Shawn Davis", "sd@uos.ac.uk");
    tracer.addStudent("S107", "John Doe", "jd@uos.ac.uk");
    tracer.addStudent("S108", "Fed Baker", "fb@uos.ac.uk");
    tracer.addStudent("S109", "Phil Smith", "ps@uos.ac.uk");

    Map<String, String> enrolments = new HashMap<>();
    enrolments.put("S101", "SOFT01");
    enrolments.put("S103", "SOFT01");
    enrolments.put("S102", "NET02");
    enrolments.put("S106", "NET02");
    enrolments.put("S110", "CYB03");
    enrolments.put("S112", "CYB03");

    tracer.loadStudentCourseList(enrolments);
    return tracer;
}
```

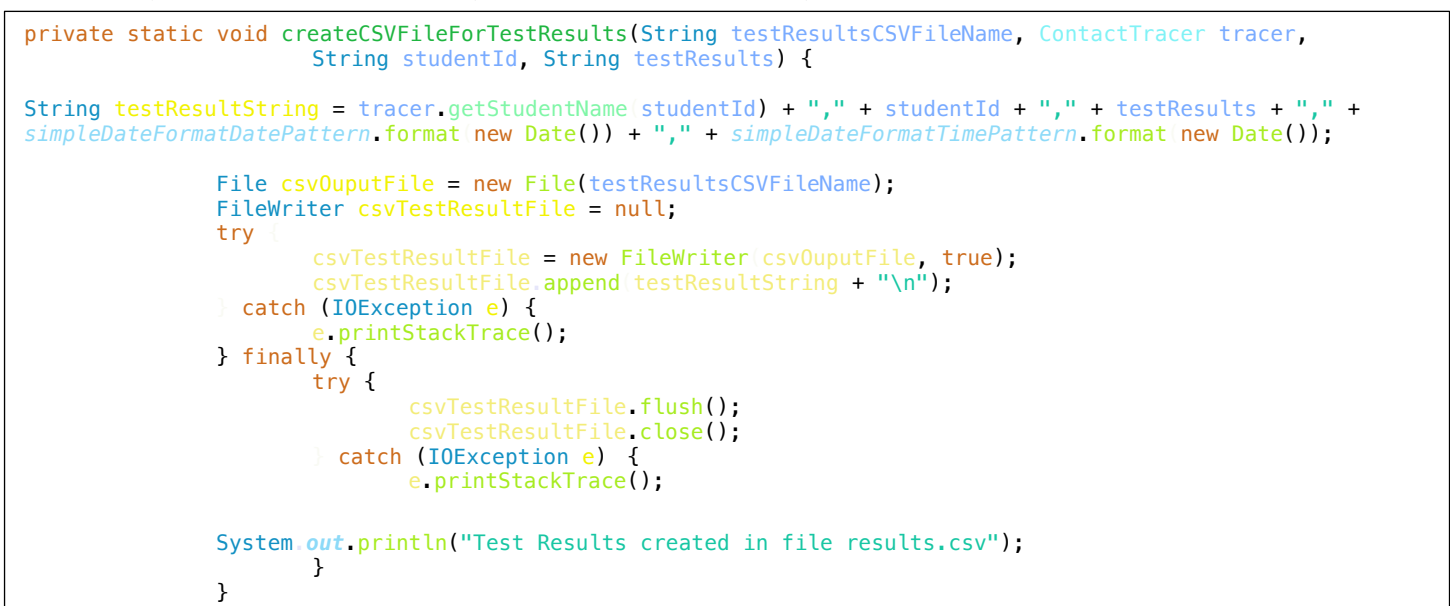
Course ID and Name

Student Details

Students enrolled in the same course

Figure 9 – shows student enrolments

Figure 9 shows different students that have enrolled in different courses, and this is the data that is required for the university to see which student test positive for covid-19 and require to isolate.



```
private static void createCSVFileForTestResults(String testResultsCSVFileName, ContactTracer tracer,
String studentId, String testResults) {

String testResultString = tracer.getStudentName studentId) + "," + studentId + "," + testResults + "," +
simpleDateFormatDatePattern.format new Date() + "," + simpleDateFormatTimePattern.format new Date();

File csvOutputFile = new File(testResultsCSVFileName);
FileWriter csvTestResultFile = null;
try {
    csvTestResultFile = new FileWriter(csvOutputFile, true);
    csvTestResultFile.append testResultString + "\n";
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        csvTestResultFile.flush();
        csvTestResultFile.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println("Test Results created in file results.csv");
}
}
```

Figure 10 – shows the CSV files

The code also throws an exception, as inside all code they have to be declared, which then needs to add the try and a catch method. A try/catch block is placed around the code that might generate an exception – this is to protect the code, *by zombies*.

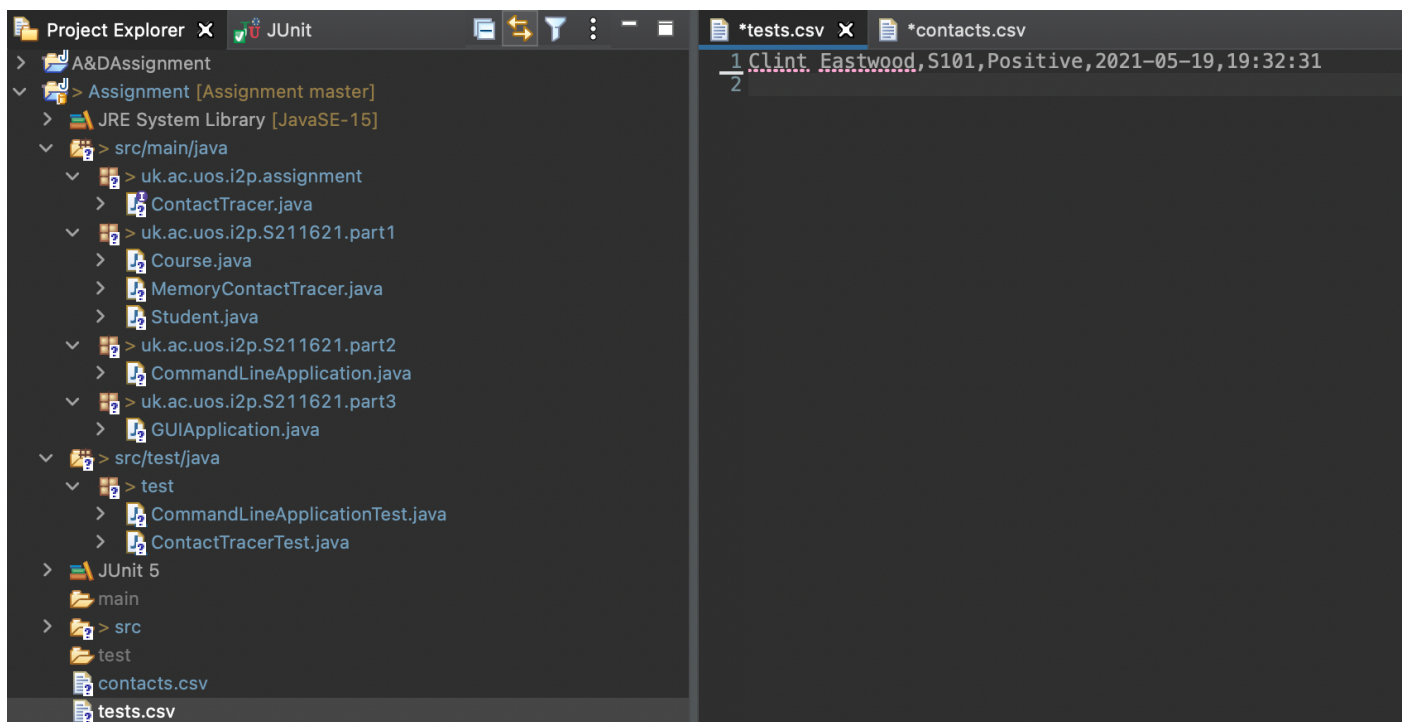


Figure 11- shows the two output files.

This test the code results for the students with positive, and two output files for each test have been created and successfully loaded (as shown in Figure 11) and have been created under the Junit 5 folder.

3. GUI – Graphical User Interface

Part 3 is to create a GUI is a graphical user interface, it is another type of interface used for the front-end user. For this application eclipse already has libraries implemented which helps to create the GUI.

```

1 package uk.ac.uos.i2p.S211621.part3;
2
3 import java.io.FileWriter;
4 import java.io.IOException;
5
6 import javax.swing.JOptionPane;
7
8 public class GUIApplication {
9
10     private static Object testResultsCSVFileName;
11     private static Object tracer;
12     private static Object studentId;
13     private static Object testResults;

```

Figure 12- shows the GUI class and Sing toolkit

Previously where classes were created, another class named GUIApplication is created. Figure 12, shows imports of Java AWT (Abstract Window Toolkit) and for the application. It has been built on the Java Swing Framework and the components follow a different model. Swing is being used in the order it can represent the data that has been created in the previous class and it controls the input of view.

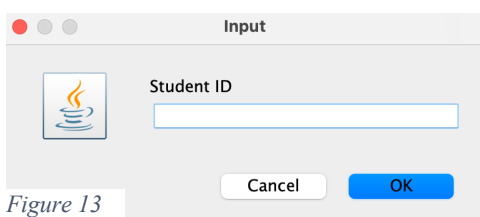


Figure 13

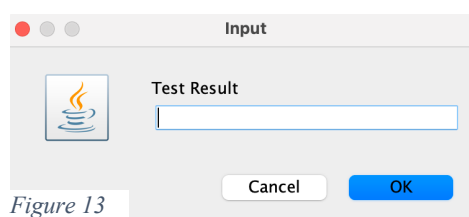


Figure 13

Figure 13 show that when the application is run inside the GUIApplication class, the front and user can input the Student ID, click 'OK, and then to Figure 14, input the Test Results click 'OK' and the results will then be submitted with no errors.

Once all this is done as all developers, try to commit the application code to GITHUB. Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world. (GITHUB, n.d.)

4. Conclusion

The project is collecting information from different students who have enrolled in various courses within the same university, course, and lecture, who have been in contact with another possible student which may have been encountered covid-19.

All the information is then transferred by the student using the Contact Tracer Application.

With the test results provided the university can see if the student may have been in contact with someone who has tested positive in the last 14 days, and if tested positive isolation is requested.

For designing the solution of the Contact Tracer Assignment a very simple approach has been taken in such a way that it works smoothly with no errors.

Bibliography

1) Anon., 2021. Eclipse Foundation. [Online]

Available at: <https://www.eclipse.org/ide/>

[Accessed 19 May 2021].

2) Eclipse, 2017. [Online]

Available at: https://www.eclipse.org/community/eclipse_newsletter/2017/october/article5.php

[Accessed 24 May 2021].

3) GITHUB, n.d. [Online]

Available at: <https://github.com/about>

[Accessed 25 May 2021].

4) javac, O., 1993. The Java Programming Language Compiler, javac. [Online]

Available at:

<https://docs.oracle.com/javase/8/docs/technotes/guides/javac/index.html#:~:text=The%20Java%20programming%20language%20compiler%2C%20javac%20%2C%20reads%20source%20files%20written,the%20Pluggable%20Annotation%20Processing%20API.>

[Accessed 19 May 2021].

5) Java, n.d. [Online]

Available at: https://java.com/en/download/help/whatis_java.html

6) Office for Students, 2021. Office for Sstudents. [Online]

Available at: <https://www.officeforstudents.org.uk/for-students/coronavirus-guide-for-students/student-guide-to-coronavirus/returning-to-university-in-2021/>

7) Oracle, 2021. [Online]

Available at: <https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html>

[Accessed 20 May 2021].

8) Oracle, n.d. [Online]

Available at: <https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html>

[Accessed 20 May 2021].

9) Oracle, n.d. Oracle. [Online]

Available at: <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

[Accessed 20 May 2021].

10) Sarangam, A., 2021. Jigsaw Academy. [Online]

Available at: <https://www.jigsawacademy.com/blogs/java/history-of-java/#:~:text=James%20Gosling%20pioneered%20Java%20in,was%20the%20first%20public%20execution.>

[Accessed 19 May 2021].

11) Veeraraghavan, S., 2021. *Simple Learn*. [Online]

Available at: <https://www.simplilearn.com/best-programming-languages-start-learning-today-article>
[Accessed 27 April 2021].

12) WHO Organization, W. H., n.d. *Pneumonia of unknown cause-china*. [Online]

Available at: <https://www.who.int/csr/don/05-january-2020-pneumonia-of-unknown-cause-china/en/>
[Accessed 19 May 2021].