## Criterion C: Development

**Techniques Used:**

1. **Abstraction**
2. **Abstract data types**
3. **2D Arrays**
4. **String manipulation**
5. **Iteration**
6. **Encapsulation**
7. **File handling**
8. **Data validation**
9. **Exception handing**
10. Automation
11. **Selection**
12. **Abstraction**

Abstraction was used in the design phase when breaking the problem down into a problem it was possible to code. I had to divide the problem into objects (student, timetable, studentlist) and methods for each process, for example checking which times pairs had in common was assigned checkpair() as part of the studentlist object.

This can be extensively seen in Criterion B.

1. **Abstract Data Types**

I required a way to store the student objects. The ArrayList data type was suitable, as demonstrated in the table below:

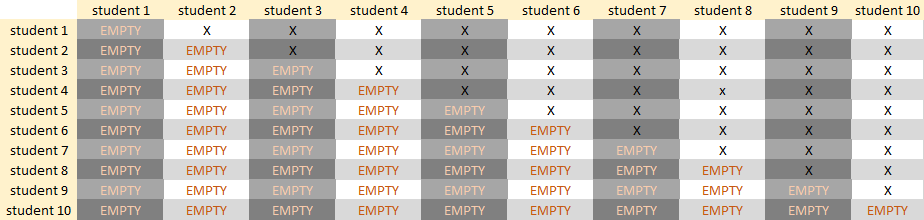
(it is compared with a standard array)

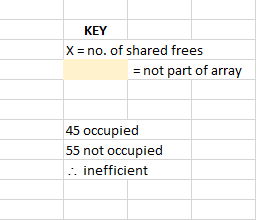
|  |  |
| --- | --- |
| **Property of ArrayList** | **Example instance where used** |
| Dynamic | The user is allowed to put up to 30 students in the program, by using a dynamic list it saves size (so if they entered 5 there wouldn’t be 25 empty slots) |
| Can contain objects | An array would not be able to contain the student objects and therefore would not fit with my OOP approach. |
| Comes with in-built methods | **boolean isEmpty()** is used at several points for data validation  **int size()** is used to find the length and saves having to count each object as it is put in the array and pass that int variable in and out of relevant methods – the one below is comparing the two. |

1. **2D Arrays**

In order to best represent the times people had in common a 2D array was used. This way I could record the interrelations between each student, so the number each had in common.

You can see the concept behind the 2D array I used below:





1. **String Manipulation**



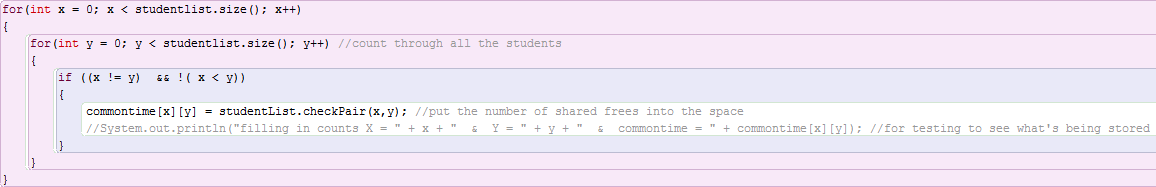
.equals() is used to compare strings – specifically the times free. Although numbers are used they are stored in a String format because there is no necessity to convert them into integers and this way if someone accidentally added a letter into the file the program would not unexpectedly terminate.

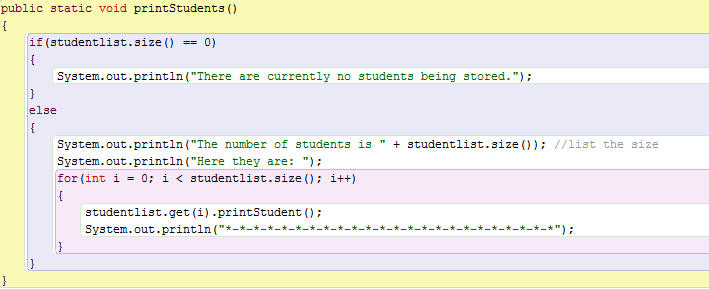


.split(",") is used to separate the String taken from each line of the .CSV file by the commas, separating out each data entity.

1. **Iteration**

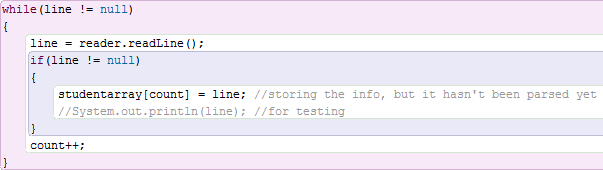
Several for() loops are used for the following purposes:

* Initializing/altering the contents of arrays
* Comparing the contents of arrays (see below)
* Printing the contents of arrays

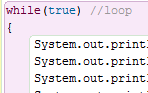


while() loops are used for the following purposes:

* while(line != null) to stop reading the .CSV file once all the lines have been read (so as soon as the line is null, it stops)



* while(true) is used to loop the main program, as it’s only supposed to terminate when someone presses “6”



1. **Encapsulation**



Although I was the only developer on the project I still used encapsulation so I couldn’t accidentally redefine one of my static variables in another method and because it’s good practice.

1. **File Handling**

The program can read from a .csv file and interpret the information, breaking it down into 1D string arrays and then putting it into an ArrayList of “student” objects. These are the exceptions I had try/catch statements to avoid:

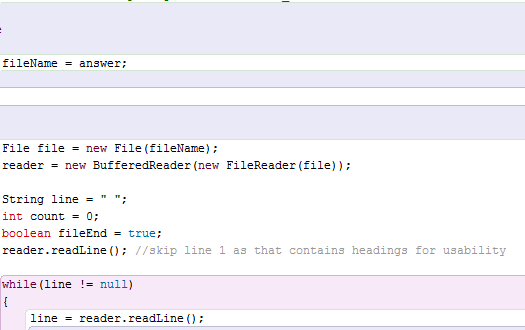




 //this one could occur at two separate unique points so it was used twice

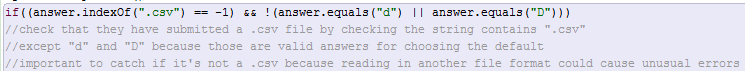


I used BufferedReader to go through the file line by line:

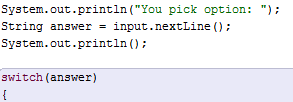


1. **Data Validation**

The file path was the main area I used data validation in. At this stage I checked that they had entered a .CSV file:



I also used a case/switch statement to catch any answers which were not an option:

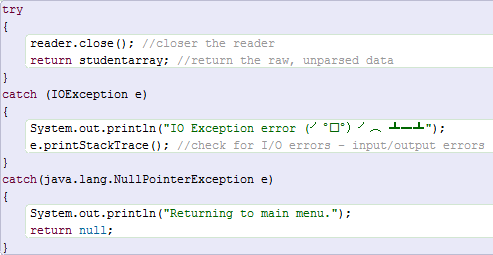


[OTHER CODE HERE]



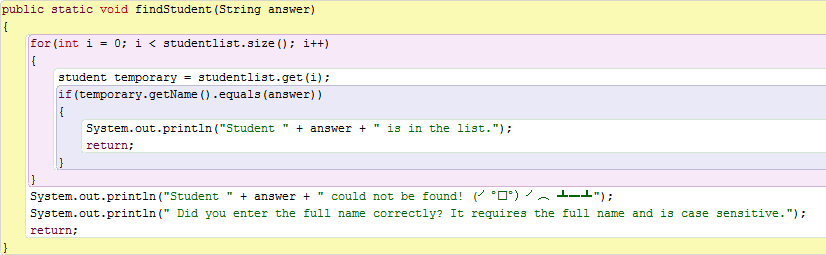
1. **Exception handing**

One point a case statement is used so the default can catch exceptions. It is also stored in a String despite asking for an int, because it doesn’t reduce functionality but allows to not need a try/catch statement. Later on when they are necessary they are used, one example being:



1. **Automation**

There is a search function which finds a student:

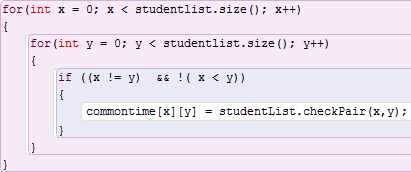


The finding pairs is also automated, the program sifts through the students to find optimal pairs (see the second part of selection for relevant screenshots).

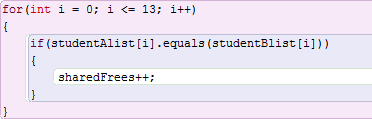
1. **Selection**

The algorithm wound fundamentally use selection because its purpose is to divide the students into pairs depending on their timetables by *selecting* the two students with the most available time in common. The outcome will be different depending on the data input.

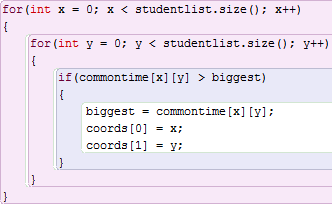
In this section the program selects the correct part of the array to analyse:



At this point it compares the two students to see how many frees they have in common:



This chooses the highest number to store as “biggest” and record which students it is:



Then print out the pair and repeat until there are no two students left to pair:

