On this module I have been taught object-oriented and functional programming paradigms, by using three different distinct languages, such as Kotlin, Scala and Clojure to complete a timetable clash detection system. Different programming languages have its own set of strengths and weaknesses.

Kotlin is a free and open source pragmatic programming language for the JVM (Java Virtual Machine) and the Android platform. It blends object-oriented and functional programming elements while emphasising interoperability, safety, clarity, and tools support.

This one biggest strength of Kotlin is that it can complies with existing Java code, for example, when doing a coursework, if I don't have a clue how to code in the Kotlin, then I can code in Java first and then the IntelliJ will complier the Java code to Kotlin code automatically, this means that two programming languages can be used conveniently at the same time in Kotlin, this is very helpful and make things easier. This is because Kotlin is positioned as a 100% Java-interoperable programming language, it is consistent with Java and all related tools and frameworks, which makes it possible to switch to Kotlin step by step. The other strength is less bug, this is because Kotlin offer a much clearer and more compact codebase that makes the code in production more stable and consistent, bugs will be detected during the compile time, this means that user can fix errors before runtime. In additional, Kotlin has combines functional and procedural programming, this means that if procedural programming has no advantage, there might have a difficulty where the functional approach can be more effective, therefore, Kotlin combines the best of the functional as well as procedural programming, this will increase the teams' productivity.

However, Kotlin also have weakness. The one biggest weakness of Kotlin during the coursework is there are limited resources to learning Kotlin. It increases the difficulty to find the resources, tutorial video, and other information over the internet.

Clojure developed by Rich Hickey, it's free, open source and a dynamic and functional dialect of the Lisp programming language on the Java platform. There are many ways to run the code in Clojure. For example, to print "Hello World" in Clojure, I can print it from terminal such as "clj-e (println "Hello World")", or I can create "hello.clj" file and run this file as script or using "src/hello.clj" with main function and other ways. The disadvantage of Clojure is that when the code has the error during the run time, Clojure will print a lots of error message in the terminal, this might be a simple error, such as typed something wrong in somewhere, got null value by looking for the wrong key in a map or trying to take the head of an empty list of something else.

Scala stand for Scalable Language, it is a multi-paradigm programming language, it includes features of functional programming and object-oriented programming. Its source code is compiled into bytecode and executed by JVM (Java Virtual Machine).

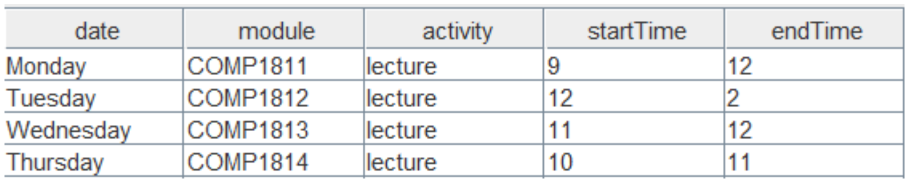
The one strength of Scala is easy to pick up, this means that Scala coming from an object-oriented background with Java or another such language, Scala's syntax appears familiar. This makes easier to pick up than other language. Also, it is much more concise than Java. For example, in Java, it would need nine lines of code to perform an operation, but in Scala, it would only need about three lines. This aids productivity. In additional, when I code in Scala, I can import the Scala class in the Java without any changes, this is making things easier. For example, during the coursework, I've created a clash detection class in scala, then I want the scala class can run in the Java GUI (Graphical user interface), all I need is to import the Scala class in the Java and that works.

Similar to the Kotlin, Scala has a limited community presence, which means that is difficult to find the resources (such as tutorial video) to learning Scala over the internet. When I am doing the coursework and get stuck in some points, and I need to spend lots of time to find the resources. This makes time consuming.

The GUI of the timetable detection system isn't that gorgeous. We faced a lot of problems during the implementation. The one problem is that I write the code in a new Java file and try to convert to the Kotlin, it is successfully converted to Kotlin code, but when I want to integrate the Kotlin code into the Java GUI file, it keeps displaying error, and I need to spend times to keep adjust the Kotlin code in the Java GUI file and make it able to call the function in Kotlin successfully call.

During the coursework, we first create a whole Java GUI it has all the needed buttons. In the first version of the Java GUI is only had two buttons in the home page which is "Set up and Manage" button, "Clash Detection" button. When user click to the "Set up And Manage" button it will jump to the other page with four buttons, which is "Set up Module", "Set up Programme", "Manage Module" and "Manage Programme" button. The set up button, will allow user to add the module information and programme information, whereas the manage button will allow user to delete the added module information and programme information. In the "Clash Detection" button, it will allow user to enter the programme name to check the clash information in the certain programme. All these GUI features are working successfully but it won't save any information because we haven't included any function in the GUI file. This is the initial prototype of the timetable detection system.

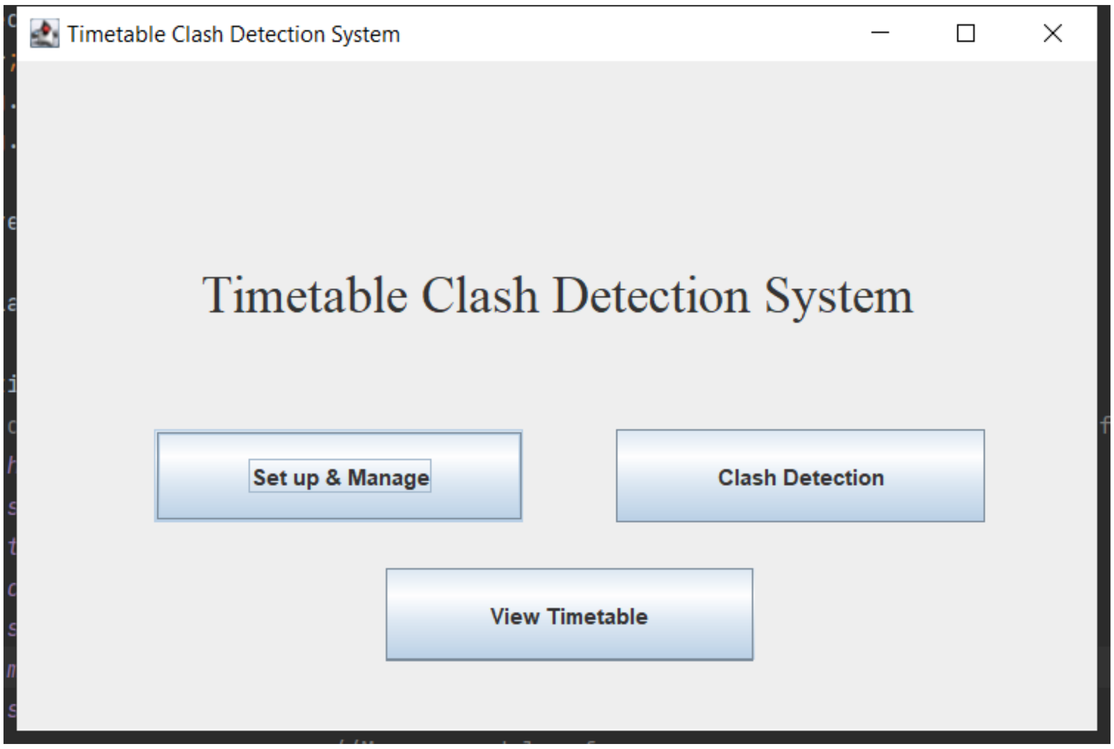
After that, we have updated the GUI feature by adding "View Timetable" button in the home page. This button will allow user to check the whole timetable information. In this feature, we sort the timetable information from Monday to Friday and sort the stating time incrementally from 9 AM to 21PM. But this timetable information can only display in the terminal. What we want is when click the "View Timetable" button there will be a dialog window pop up and bring the sorted timetable information. To achieve this, I first to sort the timetable information in the Java file and try to use "DeafultListModel" to make the timetable information to able to display in the GUI window. It was successfully, but when we try to put the worked Java file into the "interface.java" file (main GUI file) to implemented with the "View Timetable" button, and we must adjust the worked Java file and make it work in the main GUI file. Also, we want to make "View Timetable" button as the timetable table (showed in below figure). We have tried our best to implement this but wasn't successful, then we decide to keep the "View Timetable" button display timetable information as sorted list. In the future this feature can be improve by switching timetable list to the certain timetable format. The other feature such as to check the clash in the timetable. In the first, the timetable clash information can only display in the terminal, and we try to make it in the dialog window, for example when click to the "Clash Detection" button, the clash information will pop up with a little window. And we have successfully did this feature.



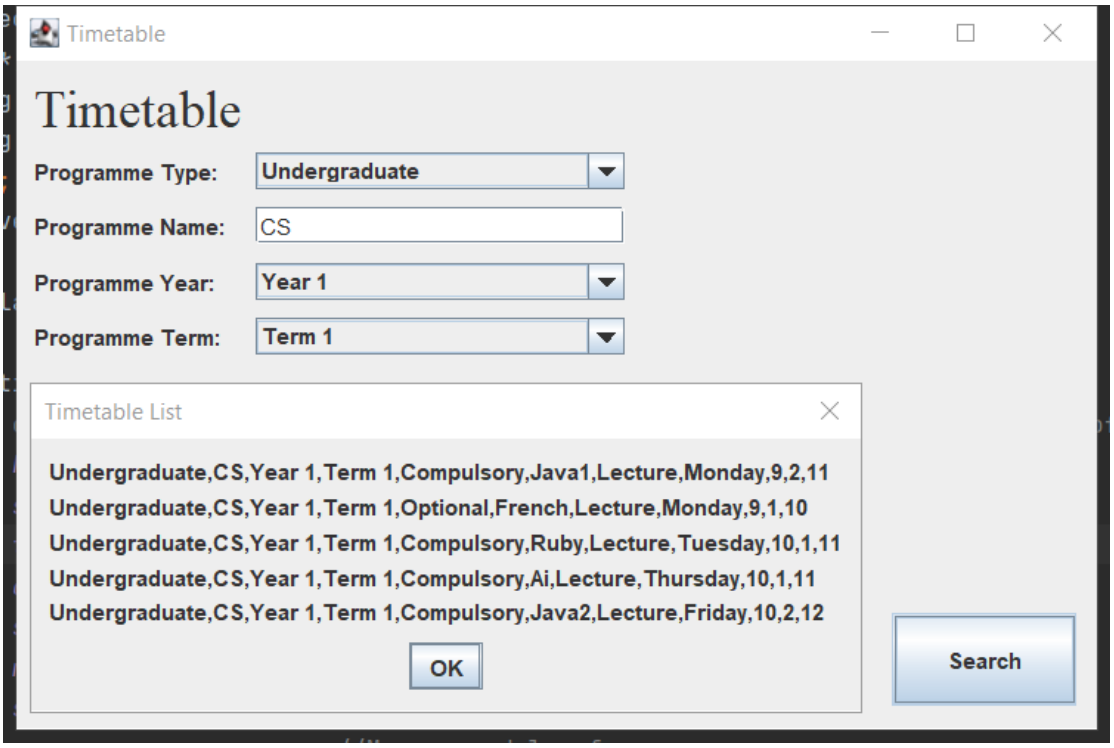
date	module	activity	startTime	endTime
Monday	COMP1811	lecture	9	12
Tuesday	COMP1812	lecture	12	2
Wednesday	COMP1813	lecture	11	12
Thursday	COMP1814	lecture	10	11

The initial timetable clash detection system only has the features I stated above. That is a big step forward to complete the whole system. We have checked the coursework specification again to make sure that we have include all the mentioned feature in the timetable clash detection system. And we found that we have missed some features which didn't include in our system. Which are 1) "postgraduate programme runs over one year", in our system postgraduate programme is set the same year as the undergraduate programme, 2) "module can be compulsory or optional", in our system we didn't clarify the compulsory and optional module, 3) "study four modules each term", 4) "Activities must not start before 9:00 or finish after 21:00", we have implemented the not start before 9:00 but missed finish after 21:00. To complete those features, we have added the if condition to the system, when postgraduate programme is more than one year, more than four compulsory modules added to the system, finished after 21:00. Then the little windows will pop up with an error message such as "Activity can't finish after 21:00".

Overall, our system is work successfully and meet all features in the coursework specification. My team are working hard and working together to help each other to solve the problems we have faced during the implementation. In this coursework I have learnt how to integrate different languages together to complete the whole project. Furthermore, the system's user interface can by improved.


Timetable Clash Detection System
Timetable Clash Detection System
Set up & Manage
Clash Detection
View Timetable

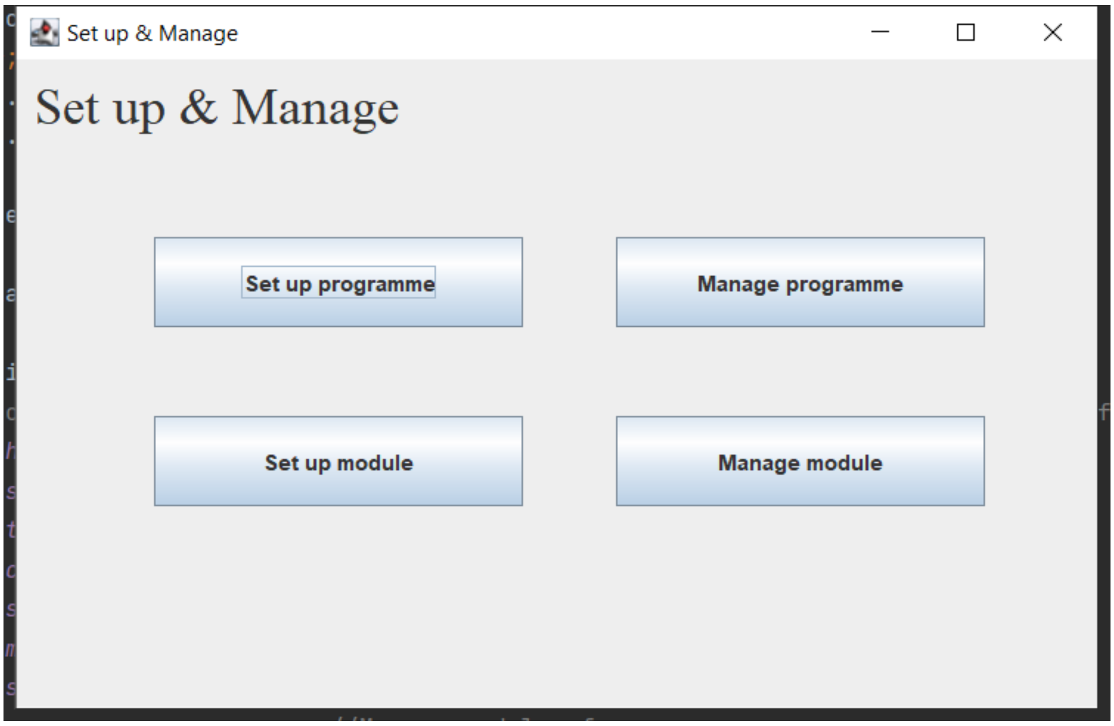
This is the home page; the user can use the buttons to choose what they wanted to do



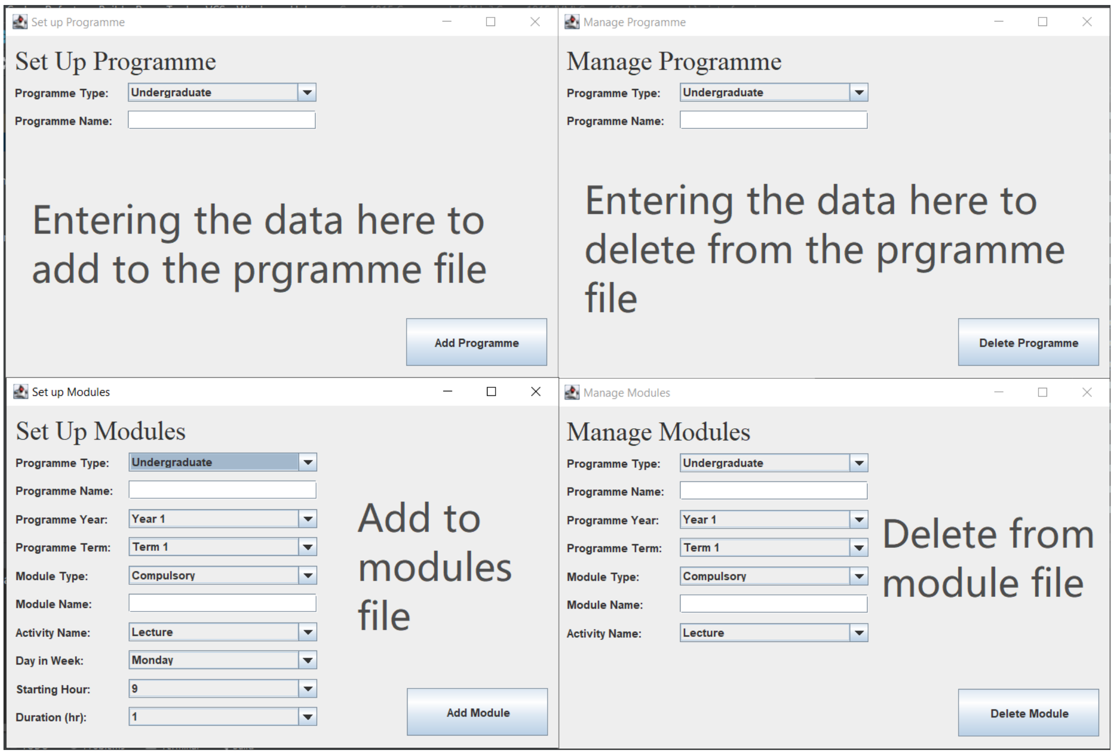
Programme Type: Undergraduate
Programme Name: CS
Programme Year: Year 1
Programme Term: Term 1

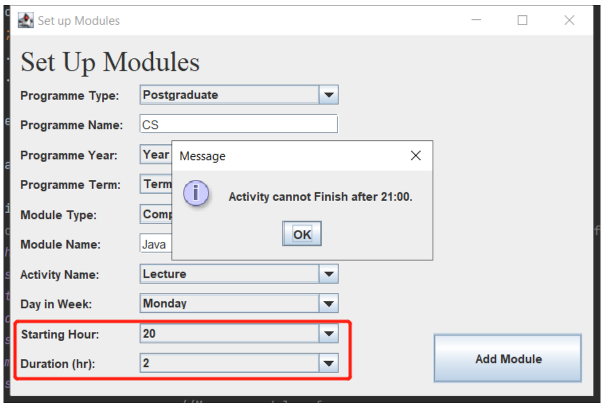
```
Undergraduate,CS,Year 1,Term 1,Compulsory,Java1,Lecture,Monday,9,2,11
Undergraduate,CS,Year 1,Term 1,Optional,French,Lecture,Monday,9,1,10
Undergraduate,CS,Year 1,Term 1,Compulsory,Ruby,Lecture,Tuesday,10,1,11
Undergraduate,CS,Year 1,Term 1,Compulsory,Ai,Lecture,Thursday,10,1,11
Undergraduate,CS,Year 1,Term 1,Compulsory,Java2,Lecture,Friday,10,2,12
```

This is the timetable page; the user can use the drop-down list and text field to select/enter the information of the timetable they wanted to see. And then they can use the search button to get a popup window that contains timetable list in order.



This is the set up and mange page, the user can use the buttons to complete the action they wanted to do





Set up Modules

Set Up Modules

Programme Type: Postgraduate

Programme Name: CS

Programme Year: Year

Programme Term: Term

Module Type: Comp

Module Name: Java

Activity Name: Lecture

Day in Week: Monday

Starting Hour: 20

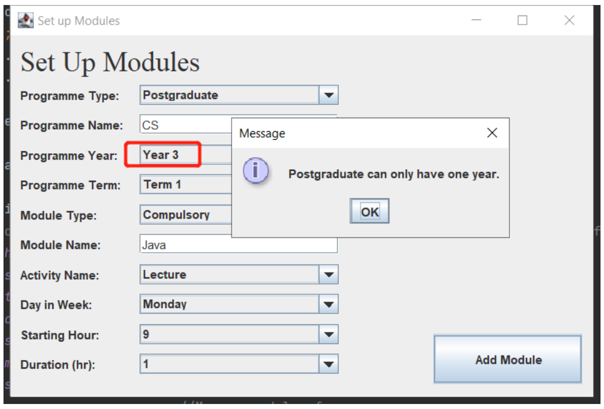
Duration (hr): 2

Message

Activity cannot Finish after 21:00.

OK

Add Module



Set up Modules

Set Up Modules

Programme Type: Postgraduate

Programme Name: CS

Programme Year: Year 3

Programme Term: Term 1

Module Type: Compulsory

Module Name: Java

Activity Name: Lecture

Day in Week: Monday

Starting Hour: 9

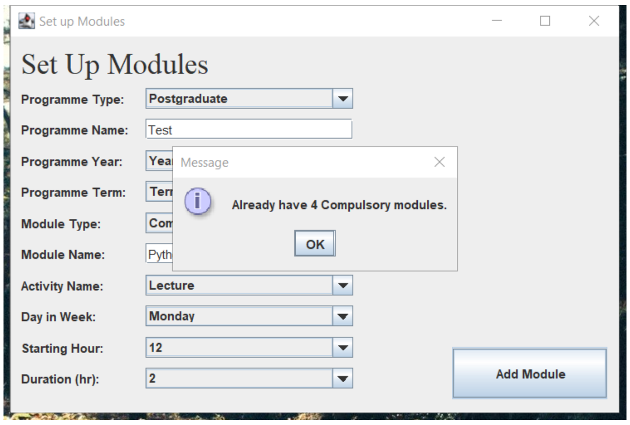
Duration (hr): 1

Message

Postgraduate can only have one year.

OK

Add Module



Set up Modules

Set Up Modules

Programme Type: Postgraduate

Programme Name: Test

Programme Year: Yea

Programme Term: Ter

Module Type: Com

Module Name: Pyth

Activity Name: Lecture

Day in Week: Monday

Starting Hour: 12

Duration (hr): 2

Message

Already have 4 Compulsory modules.

OK

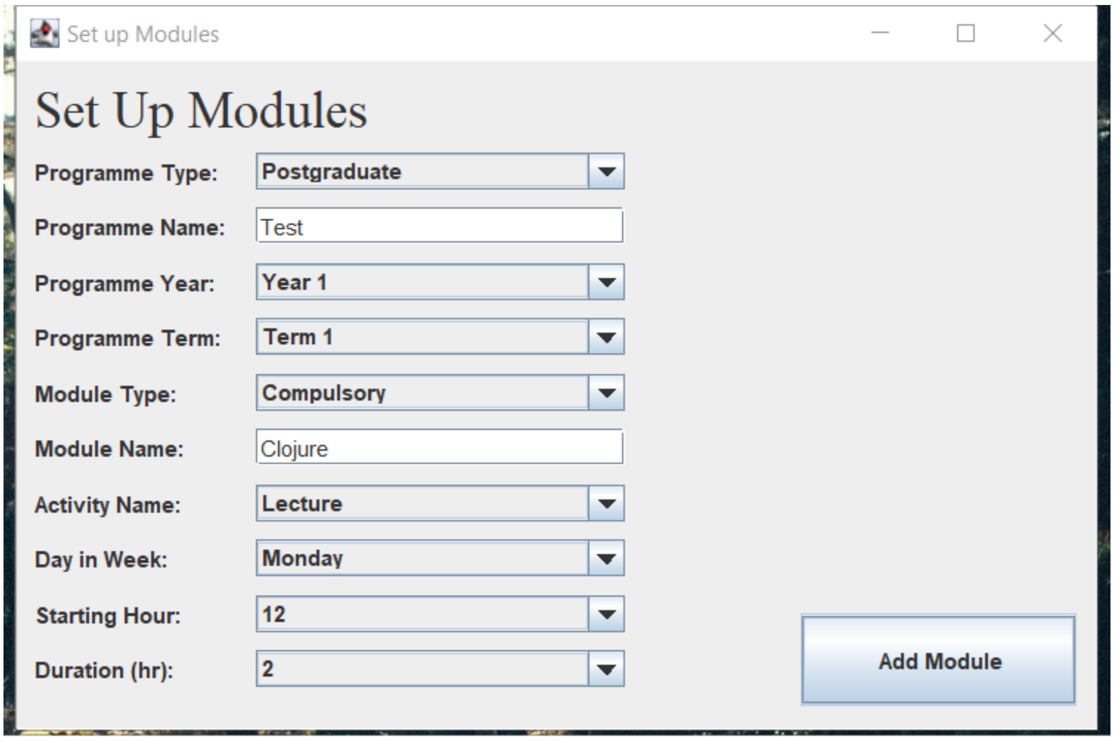
Add Module

The system will flag out with pop up window if the data does not satisfy the requirement:

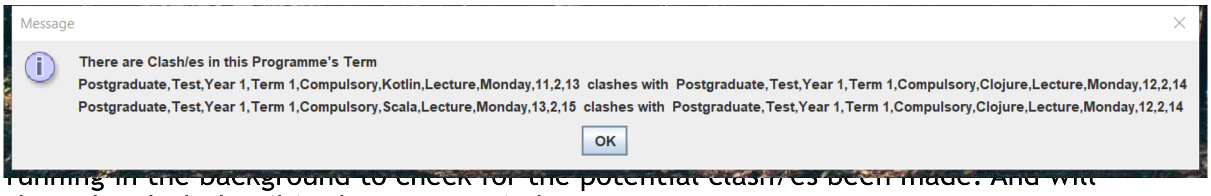
-Activity cannot finish after 9

-Postgraduate only have one year of study

-Student will study 4 modules

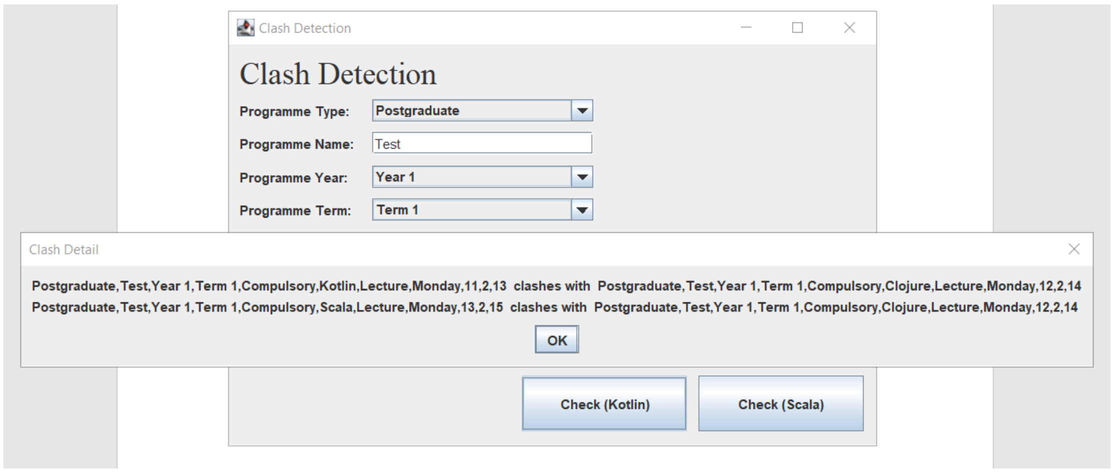


Programme Type: Postgraduate
Programme Name: Test
Programme Year: Year 1
Programme Term: Term 1
Module Type: Compulsory
Module Name: Clojure
Activity Name: Lecture
Day in Week: Monday
Starting Hour: 12
Duration (hr): 2



There are Clash/es in this Programme's Term
Postgraduate,Test,Year 1,Term 1,Compulsory,Kotlin,Lecture,Monday,11,2,13 clashes with Postgraduate,Test,Year 1,Term 1,Compulsory,Clojure,Lecture,Monday,12,2,14
Postgraduate,Test,Year 1,Term 1,Compulsory,Scala,Lecture,Monday,13,2,15 clashes with Postgraduate,Test,Year 1,Term 1,Compulsory,Clojure,Lecture,Monday,12,2,14

running in the background to check for the potential clash/es been made. And will show the clash detail in the pop-up window



Programme Type: Postgraduate
Programme Name: Test
Programme Year: Year 1
Programme Term: Term 1

Postgraduate,Test,Year 1,Term 1,Compulsory,Kotlin,Lecture,Monday,11,2,13 clashes with Postgraduate,Test,Year 1,Term 1,Compulsory,Clojure,Lecture,Monday,12,2,14
Postgraduate,Test,Year 1,Term 1,Compulsory,Scala,Lecture,Monday,13,2,15 clashes with Postgraduate,Test,Year 1,Term 1,Compulsory,Clojure,Lecture,Monday,12,2,14

And lastly, the user can enter the detail of the information here and use the button to run the clash check either with the kotlin or the Scala function. And the detail of the clash will be displayed on the pop-up window

In this project, we use three languages: Java, Kotlin and Scala. The Scala file is called "Clash Detection" this file is used to check timetable clash. In the "Clash Detection" file, we first create a class, 2) define the "val" variables, 3) define a function to get data from the "module.csv" file, 4) use for loop to compare one activity with all other activities in the list to check clashes, and 5) set clash conditions and stores clash information to the list when clash condition is true.

The Kotlin file is called “kotlin. kt”. This file contains the Object-Oriented Design, clash detection, persistence, and lambda in Kotlin. In this file, we create two classes. The first class is a data class called “Programme”; this class is used to set up the programmes of study, such as programme type and programme name. In this class, we define a function called “writeToFile”, this function is used to write the programme data to the “programm.csv” file. The other function is called “deleteLine”, which is used to delete the unwanted data. To achieve this, we first read the “programme.csv” file and search the matching lines by using “filter” and use for loop to remove the unwanted data, then use another for loop to update the new data to the “programme.csv” file.

The second class is an open class called “Module”, in this class, we define all needed variables.

Define a function called “writeToFile”, in this function, we first search through the timetable to check the number of compulsory modules and use if/else statement to set the programme type (postgraduate), year of study, maximum of 4 compulsory modules and set an end time which is before 21:00, then write data into “module.csv”.

The second function is called “deleteLine”; this is similar to the “deleteLine” in the programme class. The third function is called “searchFile”, in this function, we used if and for loop to search the clashes on the compulsory module's activities and add the matching activities to the timetable list. The fourth function is called “ovlp”; this function compares the activity with all other activities in the list to check clashes. We used two for loops to compare the date, starting time and end time in the timetable list, set clash conditions and stores clash information to the list when the clash condition is true. The last function in this class is called “sort”, and this is used to sort the timetable list incrementally; we used “sortWith” to sort the timetable list, for-loop to update the timetable list with a new sorted list, create companion object to convert days in the weekdays into numbers (such as “Monday” - 1, “Tuesday” - 2, etc..) and select time information to sorting.

The last file is written in Java called “interface.java”. This file contains all the GUI features, such as the “set up programme” button, “set up module” button, “clash detection” button and more. We create a class called “Clash_Detection” and implement it with “ActionListener” in this file.

If the user clicks the “Check (Kotlin)” button, this will retrieve strings of information from the GUI component, display the clash information by calling “check.ovlp(check.getTimetableList())”.

Then stores clash information and show it in the dialog window. The “Check (Scala)” is similar to the “Check (Kotlin)” and import the Scala file to Java.

The “Search” button calls the “searchFile” function from Kotlin. It stores timetable information into the string and shows it in the dialog window. The “Add Programme” button will call the “writeToFile” function from Kotlin. The “Delete Programme” button will call the “deleteLine” function from Kotlin. The “Add Module” button will call the “writeToFile” function from Kotlin, and if certain information did not pass the check, the error message would be raised. For example, if more than four compulsory modules are added, activities finish after 21:00. The “Delete Module” button will call the “deleteLine” function from Kotlin.

Ref

https://www.netguru.com/blog/kotlin-pros-and-cons

https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html