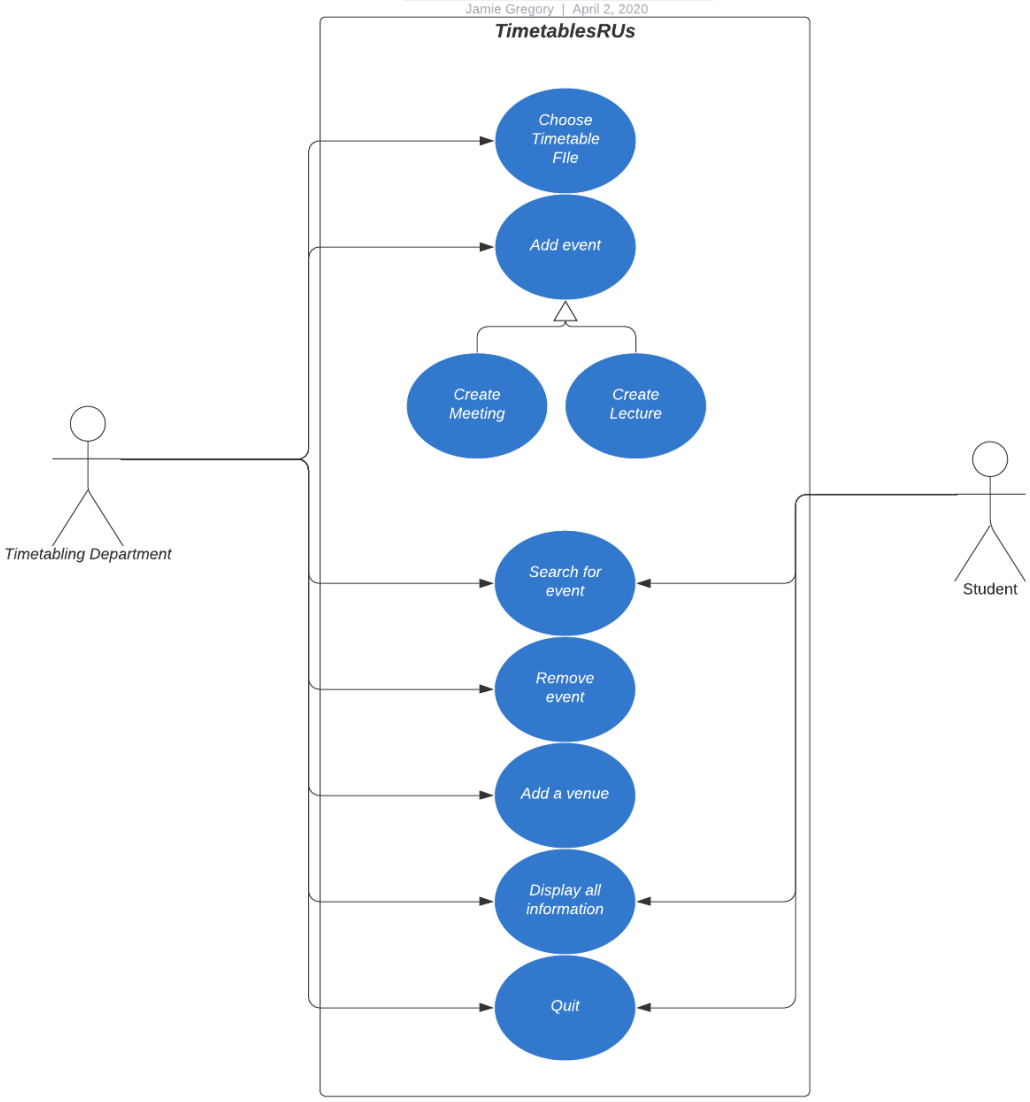
Mini Assignment 3 - TimetablesRUs

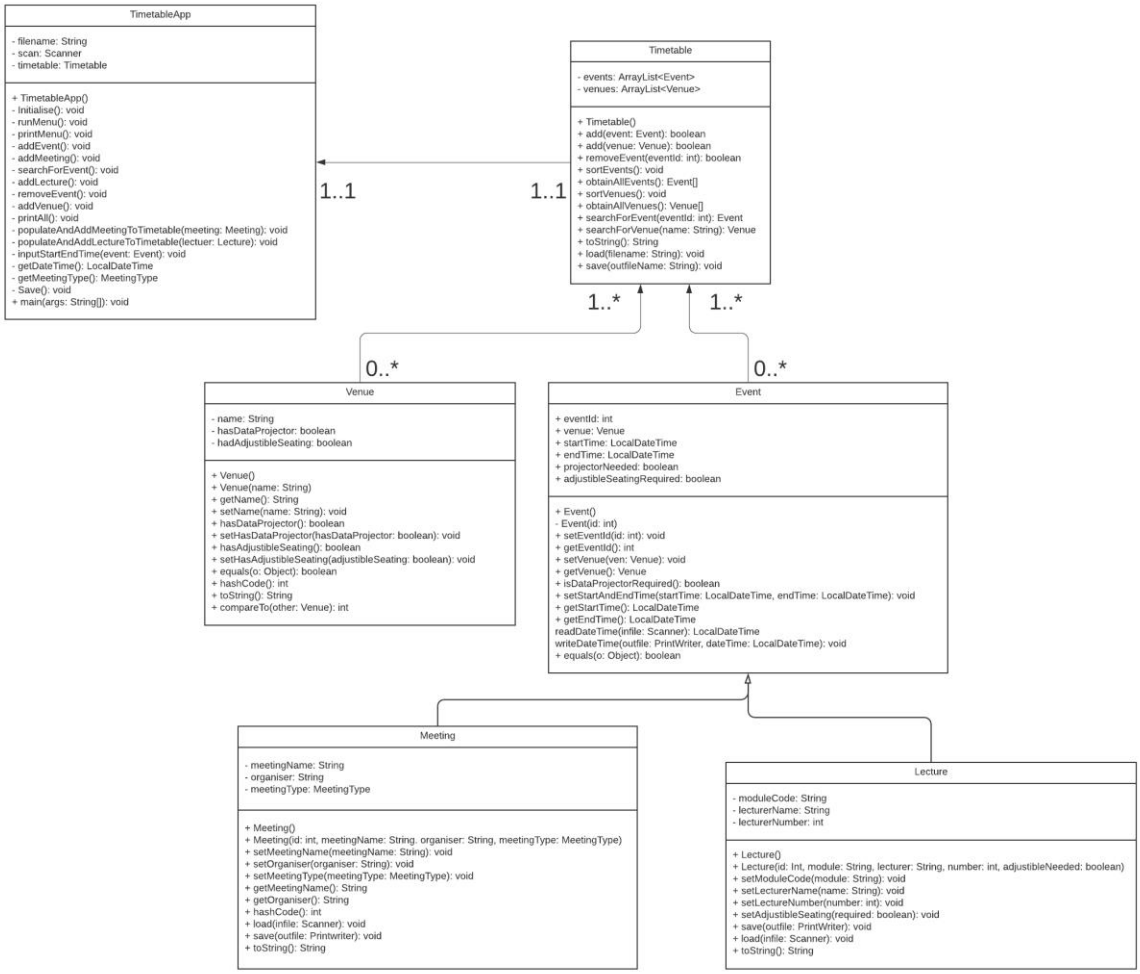
JAMIE GREGORY
JAG73@ABER.AC.UK

02/04/2020

Timetable Use Case

Jamie Gregory | April 2, 2020




TimetableApp
- filename: String
- scan: Scanner
- timetable: Timetable
+ TimetableApp()
- Initialise(): void
- runMenu(): void
- printMenu(): void
- addEvent(): void
- addMeeting(): void
- searchForEvent(): void
- addLecture(): void
- removeEvent(): void
- addVenue(): void
- printAll(): void
- populateAndAddMeetingToTimetable(meeting: Meeting): void
- populateAndAddLectureToTimetable(lectuer: Lecture): void
- inputStartEndTime(event: Event): void
- getDateTime(): LocalDateTime
- getMeetingType(): MeetingType
- Save(): void
+ main(args: String[]): void
1..1
1..1
Timetable
- events: ArrayList<Event>
- venues: ArrayList<Venue>
+ Timetable()
+ add(event: Event): boolean
+ add(venue: Venue): boolean
+ removeEvent(eventId: int): boolean
+ sortEvents(): void
+ obtainAllEvents(): Event[]
+ sortVenues(): void
+ obtainAllVenues(): Venue[]
+ searchForEvent(eventId: int): Event
+ searchForVenue(name: String): Venue
+ toString(): String
+ load(filename: String): void
+ save(outfileName: String): void
1..*
1..*
0..*
0..*
Venue
- name: String
- hasDataProjector: boolean
- hadAdjustibleSeating: boolean
+ Venue()
+ Venue(name: String)
+ getName(): String
+ setName(name: String): void
+ hasDataProjector(): boolean
+ setHasDataProjector(hasDataProjector: boolean): void
+ hasAdjustibleSeating(): boolean
+ setHasAdjustibleSeating(adjustibleSeating: boolean): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
+ compareTo(other: Venue): int
Event
+ eventId: int
+ venue: Venue
+ startTime: LocalDateTime
+ endTime: LocalDateTime
+ projectorNeeded: boolean
+ adjustibleSeatingRequired: boolean
+ Event()
- Event(id: int)
+ setEventId(id: int): void
+ getEventId(): int
+ setVenue(ven: Venue): void
+ getVenue(): Venue
+ isDataProjectorRequired(): boolean
+ setStartAndEndTime(startTime: LocalDateTime, endTime: LocalDateTime): void
+ getStartTime(): LocalDateTime
+ getEndTime(): LocalDateTime
readDateTime(infile: Scanner): LocalDateTime
writeDateTime(outfile: PrintWriter, dateTime: LocalDateTime): void
+ equals(o: Object): boolean
Meeting
- meetingName: String
- organiser: String
- meetingType: MeetingType
+ Meeting()
+ Meeting(id: int, meetingName: String. organiser: String, meetingType: MeetingType)
+ setMeetingName(meetingName: String): void
+ setOrganiser(organiser: String): void
+ setMeetingType(meetingType: MeetingType): void
+ getMeetingName(): String
+ getOrganiser(): String
+ hashCode(): int
+ load(infile: Scanner): void
+ save(outfile: Printwriter): void
+ toString(): String
Lecture
- moduleCode: String
- lecturerName: String
- lecturerNumber: int
+ Lecture()
+ Lecture(id: Int, module: String, lecturer: String, number: int, adjustibleNeeded: boolean)
+ setModuleCode(module: String): void
+ setLecturerName(name: String): void
+ setLectureNumber(number: int): void
+ setAdjustibleSeating(required: boolean): void
+ save(outfile: PrintWriter): void
+ load(infile: Scanner): void
+ toString(): String

Solving the assignment

When I first looked at the assignment brief, I was under the assumption that it was going to be an easy one. After working on it, it quickly got more complex than I anticipated. I spent some time breaking it down on paper to help me get an idea of what was wanted from the program. I wrote down what classes I needed, what information they would hold and the functions I would need to manipulate them. Once I had broken it down enough, I set to work on realizing my ideas.

Firstly, handing the incorrect input of the user. I used try:catch to catch the InputMismatchException. This works fine and does correctly catch the exception, but it isn't exactly elegant. I tried to isolate the try:catch to a few lines of code, but the scope issues prevented this from working.

Another difficulty I had was implementing the sorting for events. I had very little experience with sorting before this assignment, so implementing a method that would sort using two fields was new to me. Due to my lack of experience, I had to spend some time researching how I could actually do this, but I did figure out an implementation that works, though I'm sure it's not the cleanest solution.

I do believe that I completed all code that was required for this assignment, though I didn't add much/any flair, which I do regret.

Testing this program was fairly simple, due to the scale of the program. I manually tested each option in the menu, making sure to try both correct and incorrect inputs to try and break it. The code handles illegal input now, so that wasn't an issue. I also spent a lot of time testing different text files with different contents to ensure that my implementation was working correctly.

I personally believe that my work could get me around 55-65% for this assignment. My documentation is clear and shows the areas in which I struggled with the problems that arose. My implementation does everything that it is asked. The indentation is correct, I made sure to use the correct access specifiers. Step 4 has been completed; inheritance has been used, equality is indeed determined by ID, I have implemented methods to sort the slots/venues and the printAll method always sorts them in the desired way. The TimetableApp class has been modified appropriately and the data file was slightly modified to allow for loading and saving of the new information required.

```
Please enter the filename of timetable information: C:\Users\jamie\Desktop\Mini-Assignment-3\TimetablesRUs\tt.txt
```

The program firsts asks for the filename of the timetable information

```
Using file C:\Users\jamie\Desktop\Mini-Assignment-3\TimetablesRUs\tt.txt
1 -  add a new event
2 -  search for a booked event
3 -  remove a timetable event
4 -  add a venue
5 -  display everything
q -  Quit
What would you like to do:
```

The user is then presented with a list of actions.

```
Meeting or Lecture?(M/L)
M
Enter meeting name:
Student Meeting
Enter organiser name:
Jamie
Meeting type, enter the number (1 - staff meeting
 2 - learning and teaching meeting
 3 - subject panel meeting
 4 - other kind of meeting
4
Meeting type selected: OTHER
Enter the unique timetable meeting identifier: (unique number)
4
Enter start time for timetable event
On one line (numbers): year month day hour minutes
2020 04 02 15 30
The date/time you entered was: 2020-04-02T15:30
Enter end time for timetable event
On one line (numbers): year month day hour minutes
2020 04 02 15 45
The date/time you entered was: 2020-04-02T15:45
Is a data projector required?(Y/N)
Y
Enter venue name
B23
```

The user can create an event with the required information. They can also remove said event by providing the event id.

```
Enter the venue name
A12
Does it have a data projector?(Y/N)
Y
Does it have adjustible seating?(Y/N)
Y
```

A venue can be made, with both data projectors and adjustable seating.

```
Venues in timetable system are:
Venue{name='B22', hasDataProjector=true, hasAdjustibleSeating=false}
Venue{name='B23', hasDataProjector=true, hasAdjustibleSeating=true}
Venue{name='Room101', hasDataProjector=false, hasAdjustibleSeating=false}

events in timetable are:
Meeting{meetingName='staff 1', organiser='cwl', meetingType=STAFF, meetingId=1, requiresDataProjector=true, venue=Venue{name='B22', hasDataProjector=true, hasAdjus
Meeting{meetingName='Meeting to discuss first year languages', organiser='afc', meetingType=OTHER, meetingId=2, requiresDataProjector=false, venue=Venue{name='B22'
```

The user can also display both the venues and events.