Software testing: assignment 2

Contents

[Testing approach: 2](#_Toc10063472)

[TASK 1: 4](#_Toc10063473)

[Leap year issue: 4](#_Toc10063474)

[Case sensitivity issue: 5](#_Toc10063475)

[Input checking and ambiguous errors: 6](#_Toc10063476)

[Bookings associated with certain days of the month: 7](#_Toc10063477)

[A vacation can be booked on the 32nd of the month 8](#_Toc10063478)

[TASK 2) Fault testing 9](#_Toc10063479)

[TASK 3) TEST CASES OF STRUCTURAL TESTING OF CLASSES 11](#_Toc10063480)

[ROOM: 12](#_Toc10063481)

[Planner: 13](#_Toc10063482)

[Person: 14](#_Toc10063483)

[Organisation 15](#_Toc10063484)

[Calender: 16](#_Toc10063485)

[Test plan: 17](#_Toc10063486)

[Defect report: 22](#_Toc10063487)

[Test case Results: 23](#_Toc10063488)

[Conclusions: 24](#_Toc10063489)

[MINUTES OF MEETING 26](#_Toc10063490)

[Meeting 1 26](#_Toc10063491)

[Meeting 2 27](#_Toc10063492)

[Meeting 3 28](#_Toc10063493)

[APPENDIX 1 (Black-Box Test): 29](#_Toc10063494)

[Appendix 2 (Integrated Tests): 30](#_Toc10063495)

# Testing approach:

The given case study is provided with 7 class files and 6 high level functions namely for the planner activity. To start the testing process, we have created an Eclipse Project and added the class files provided under a package called planner. Then we have created the Test folder and made the test classes for implementation.

Main Classes

1. Schedule Meeting
2. Schedule Vacation
3. Person Vacation
4. Person Availability
5. Peron Agenda
6. Room Availability
7. Room Agenda

All functions have void methods

We create a File Input stream to generate the cases using a text file. The text file has the following data, startDay.

The Black-box test is run through an automated test which implements a file input stream inside test folder under BlackBoxTest.java. The rationale of automating Blackbox is that the program already work there we can stream the inputs and check the console for the results. Text data files1. ScheduleMeeting.txt2. ScheduleVacation.txt

We believe its appropriate to assume the following:

* The software life span will have a maximum lifespan of approximately 10 to 30 years
* The user has some idea of what they are entering the program
* The program is used in a professional manner
* Reasonable values will be entered

The team shouldn’t waste time running testcases for the maximum & minimum possible value e.g. year 1600 & 3000 . The team should instead prioritise testing the most common value and continue down that path and at the end test an outlier values if required. If the outlier is bound to occur less then 5% of the time it can be ignored.

|  |  |
| --- | --- |
| Test case | Probability Set |
| 1 | 0.8 |
| 2 | 0.8 |
| 3 | 0.8 |
| 4 | 0.8 |
| 5 | 0.14 |
| 6 | 0.14 |
| 7 | 0.06 |

When choosing values for the test case:

* If the module we are testing utilises a boundary e.g. if(x < =5 && x>=1) We should use boundary value analysis to test the program. The yellow values we test to ensure there is no flaw with the boundary. The other values can be chosen randomly to test the system but it is not necessary.

|  |  |  |
| --- | --- | --- |
| Invalid (min -1) | Valid | Invalid (max+1) |
| 0 | 1, 2, 3, 4, 5 | 6 |

* If the module we are testing does not utilise a boundary e.g. if(x ==”Y”) we should use equivalence partitioning where the value is either true or false where we test the valid inputs and some invalid inputs

|  |  |
| --- | --- |
| Invalid | Valid |
| “N”, “qwerty” ,5, 4.1, “q”, “y” | “Y” |

We also must define what an acceptable error message is; The error message must contain enough information to alert the customer what mistake they have made, in the case it is not an error due to the customer it should let them know of this.

Good user error message; “ERROR, user entered unexpected value, Value: x”

Good user error message; “ERROR, user entered unexpected datatype, datatype: x.datatype, expected int”

Good system error message; “UNEXPECTED ERROR, please restart application or contact us at x”

Bad error message: “ERROR”

Bad error message: “ERROR 190xuc”

# TASK 1:

Refer to Appendix 1 to see the test case

## Leap year issue:

A leap year effects this program as it adds an extra day to the year, therefore February the 29th is a valid date in the event of a leap year. A leap year happens approximately once every four years.

To test this issue, we will have to try book the 29th of February on leap years and years that aren’t leap years.

|  |
| --- |
| **Leap Year** |
| 2020 |
| 2024 |
| 2028 |
| 2032 |
| 2044 |
| 2060 |
| 2172 |
| 1804 |
| 1972 |

*Green = Test*

*Yellow = Consider testing*

*Red = only test as an extreme outlier or don’t bother testing*

While there are many options that can be tested for this application. It makes sense to test dates that will are most likely to be used in the application.

|  |  |
| --- | --- |
| **Date** | **Probability of use** |
| 2019-2035 | 0.8 |
| 2036-2060 | 0.1999 |
| 2061-2100 | 0.001 |
| 2101 onwards | 0.0001 |
| 2000-2018 | 0.0001 |
| Before 2000 | 0.0000000000000000001 |

Since the program is a booking system it is highly unlikely that someone would enter a date in the past as it would serve no purpose, it is also quite unlikely that this software will be used for more than 20 years. Due to these reasons its best to test leap years that are most likely to occur.

For this testing many green leap years should be tested, and a few yellow leap years should be tested.

|  |
| --- |
| 2020 |
| 2024 |
| 2028 |
| 2032 |
| 2044 |
| 2060 |

## Case sensitivity issue:

If the user enters an input not matching the checker it is not valid which can be frustrating for the end user.

E.g.

A != a

Yes != YES || yes || yEs || yES

Let’s say we have the user input their full name John Smith

|  |  |
| --- | --- |
| **Input** | **Probability** |
| John Smith | 0.2 |
| john smith | 0.2 |
| John smith | 0.15 |
| john Smith | 0.15 |
| JOHN SMITH | 0.2 |
| jONh SmiTH | 0.05 |
| JOnh SMITH | 0.05 |

If we only accommodate for 1 answer e.g. input = “john smith” we are only accounting for what will be input 1/5 of the time, frustrating a first-time user.

Therefore all results should be tested to ensure a favourable user experience

## Input checking and ambiguous errors:

To test this we will use a variety of invalid inputs and see what the program returns as an error message we should test:

* Names
* Time
* Date
* Month
* Room
* Attendee
* Option

|  |  |
| --- | --- |
| Option | Test |
| NAME | !ALPHABETIC |
| TIME | !Valid TIME |
| TIME | START !before END |
| DATE | !Valid DATE |
| MONTH | !Valid Month |
| ROOM | !Valid ROOM |
| ATTENDEE | !ALPHABETIC |
| OPTION | !NUMERIC |
| OPTION | !EXIST |

Refer to testing approach for information on a good error message

e.g.

ERROR option is not numeric

ERROR option does not exist

ERROR date is not valid

## Bookings associated with certain days of the month:

In any year there will be a set amount of days in each month, in a leap year February will have 29 days instead of 28

|  |  |
| --- | --- |
| **Month** | **Days** |
| January | 31 |
| February | 28 |
| March | 31 |
| April | 30 |
| May | 31 |
| June | 30 |
| July | 31 |
| August | 31 |
| September | 30 |
| October | 31 |
| November | 30 |
| December | 31 |

We should test the boundary for these months to ensure that they cant book an invalid date. To do this we test the last day of that month (x) and the invalid date (x+1)

|  |  |
| --- | --- |
| **Month** | **Values to test** |
| January | 31, 32 |
| February | 28, 29 |
| February (LEAP YEAR) | 29, 30 |
| March | 31, 32 |
| April | 30, 31 |
| May | 31, 32 |
| June | 30, 31 |
| July | 31,32 |
| August | 31,32 |
| September | 30, 31 |
| October | 31, 32 |
| November | 30, 31 |
| December | 31, 32 |

## A vacation can be booked on the 32nd of the month

No month has 32 days, considering we already tested the boundaries of each month, we should know whether the boundary is valid so, we should test valid day at the boundary and invalid days of each month.

Basing it on the chart of days in each month above

|  |  |
| --- | --- |
| Months to test | Values to test |
| Jan, March, May, July, August, October & December | 31 (V), 32 (NV), 34(NV), |
| April, June, September, November | 30(V), 31(NV), 32(NV) |
| February | 28(V), 29(NV), 32(NV) |
| February (Leap year) | 29(V), 30(NV), 32(NV) |

Issues booking event between 11-11:59pm:

The issue seems to be the booking not considering the day rolling over, a boundary test should be performed and then checked whether the time rolls over

e.g. 10:59, 11:00, 11:59(day) , 12:00(day +1)

Issues with bookings associated with 30th of November

A boundary test should be performed this is already covered in the certain days of the month & the vacation can be booked on the 32nd of the month.

# TASK 2) Fault testing

In this section we are trying to crash the program, we go about this by inputting faulty user input when input is prompted as this is the dynamic aspect of the program.

E.g. enter a valid date:

-> 100/29/4000

-> ERROR || CRASH

Rationale

* Identify classes and methods that throw exceptions
* Target the specific method to force the program to throw exception
* Test for variable data types
* This helps in identifying methods which need automated testing

Fault testing is the process of identifying errors by trying to break the program. These errors can then be fixed, the objective of this is to provide the program with flexibility and an increased level of stability. In the case that these incorrect inputs don’t crash the program or throw an error message and instead continue executing the code, The code must be changed to register that input as an error.

NOTE: refer to appendix 2 to view test cases

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name | Method Name | Exception Thrown | Details |
| Planner | none |  |  |
| Calendar | checkTimes() | ConflictsException |  |
|  | isBusy() | ConflictsException |  |
|  | addMeeting() | ConflictsException | isBusy() and addMeeting() methods internally call the checkTimes method which throws the exception. |
| Meeting | none |  |  |
| Organization | checkRoom(String ID) | Exception |  |
|  | checkPerson(String name) | Exception |  |
| Person | addMeeting() | ConflictsException |  |
|  | isBusy() | ConflictsException | isBusy() method internally calls the isBusy() method of the Calendar class |
| Room | addMeeting() | ConflictsException |  |
|  | isBusy() | ConflictsExxception |  |
|  |  |  |  |
|  |  |  |  |

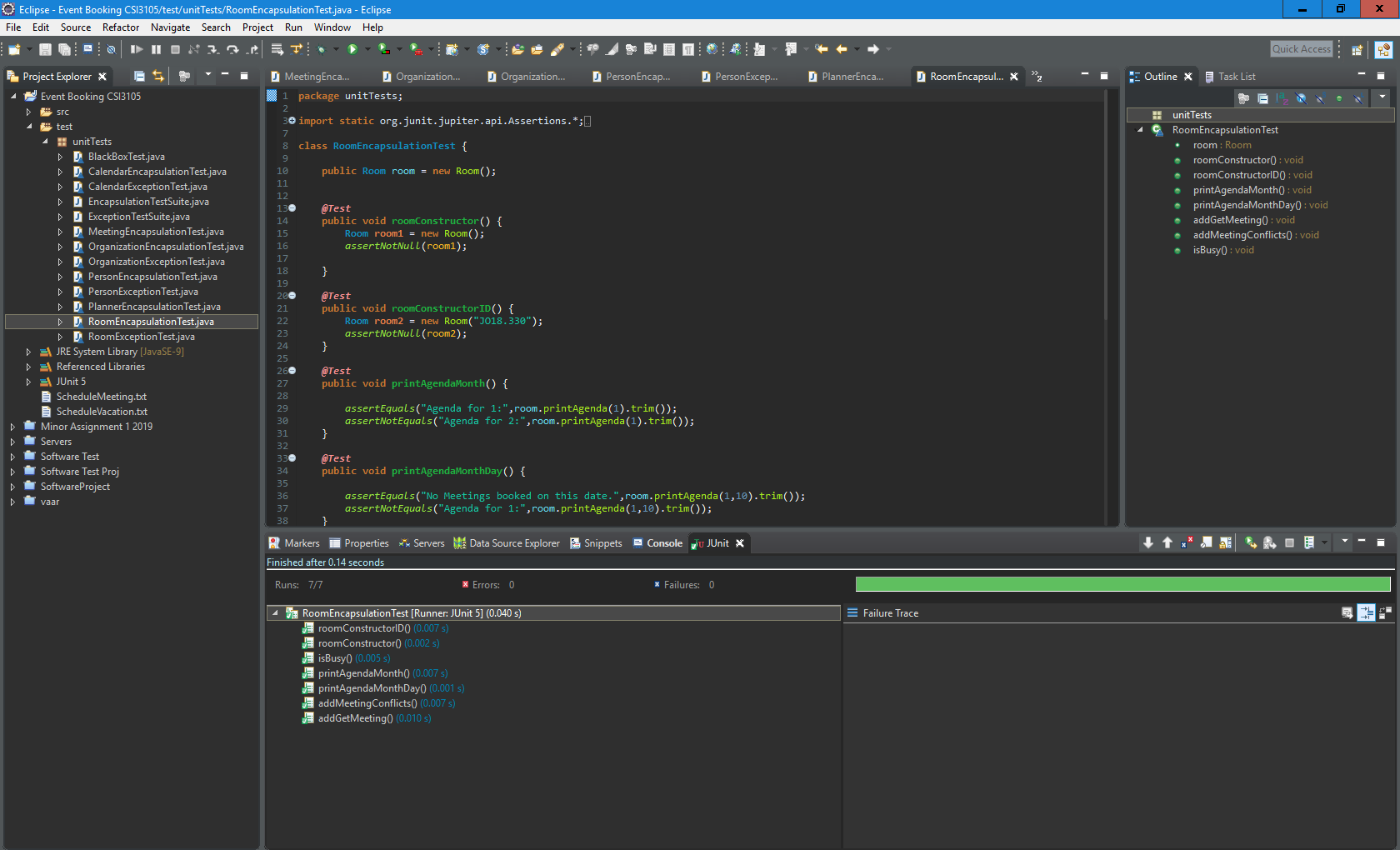
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| **A-01** | **Calendar.java**  Leap year Feb29 book event | **Day does not exist**  **Cannot add data to program** | **Error message”Day does not exist”**  **Return** | **The program need to upgrade and able to add leap year feb29 book event** | **Leap year feb29 common problem** |
| **A-02** | **Calendar.java**  **checkTimes()**  Issues booking between 11:00pm–11:59pm | **Abe to add data between 11:00-11:59** | **Error message”illegal hour”** | **The program need to upgrade and able to select time between 11:00 – 11:59** | **Problem that need to fix ASAP** |
| **A-03** | **Calendar.java**  Booking associated with 30th November | **Able to add data with booking associated with 30th November** | **Error message” day does not exist”** | **The program need to upgrade and able to add data from 30th November** | **Issues from the assignment** |
| **A-04** | Calendar.java  Vacation booked start date can be 32nd | **Unable to add data of vacation booked start date 32nd** | **Error message” day does not exist”** | **The program need to upgrade and able to add data from 32nd** | **Extract work from assignment** |
| **A-05** | Main menu() intput”1”, enter a month with string”abcde” | **Unable move to next step** | **Error message”please enter a number from 1 – 12”** | **Integer Fault testing** | **This section only allow integer** |
| **A-06** | Main menu() intput”3”,enter a starting hour”23:20” | **Should able to book vacation dates** | **Error message” please enter a number from 0-6”** | **Unable to book the vacation dates** | **Special hour booking test** |
| **A-07** | Meeting.java();Schedule a meeting;  Rooms select input rooms is not in the list example: MT18.111 | Unable move to next step, message display error:  “must select the room that is provide in the list” | Error message display :” requested room does not exist” | As an expected result with error message display | Error testing |
| **A-08** | Meeting.java();Schedule a meeting;  Starting hour of meeting input.  Example”50” | Unable move to next step, message display error:  Must be 0-24 digital number | Error message display: “Illegal hour.” | As an expected result with error message display | Error testing |
| **A-09** | Enter the vacation day that is out of range  Example “38” | Error message will display on the screen | Day does not exist show in the program | Day of range 1-30 | Day must between 1 -30 |
| **A-10** | String value input test for month enter.  Example “HTT” | Error message display | Error message display please enter number from1-31. | Fault testing method | Month method not allow string |

# TASK 3) TEST CASES OF STRUCTURAL TESTING OF CLASSES

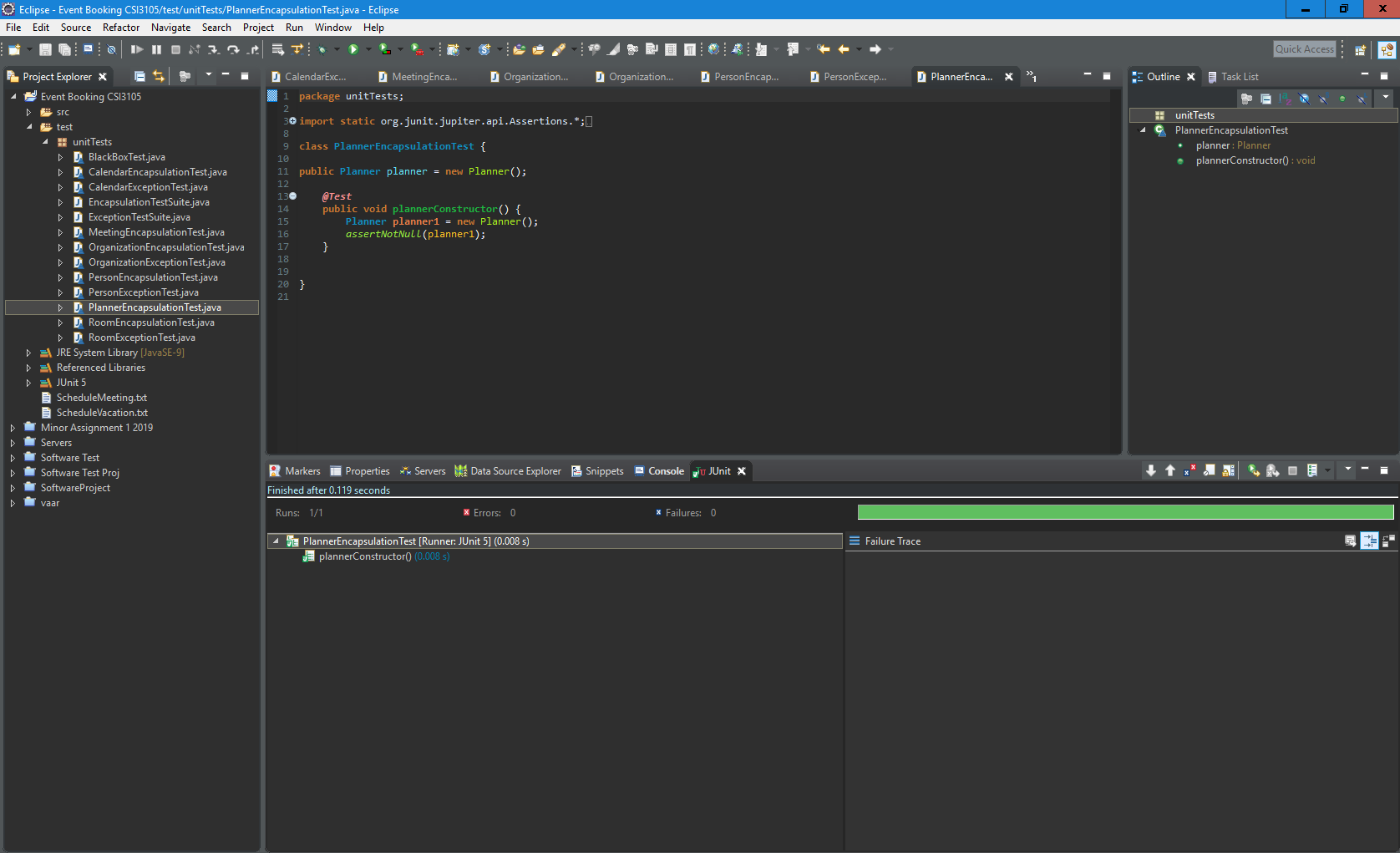
We will create test case to determine whether the class reaches certain milestones such as calling another function, printing something or booking a meeting. in the case of this program the main functions are separated into different files, so we must test the integration between these files to ensure the calls work correctly. It is very important that bookings are made correctly as well as other important processes.

Encapsulation tests must be done for each file of the application

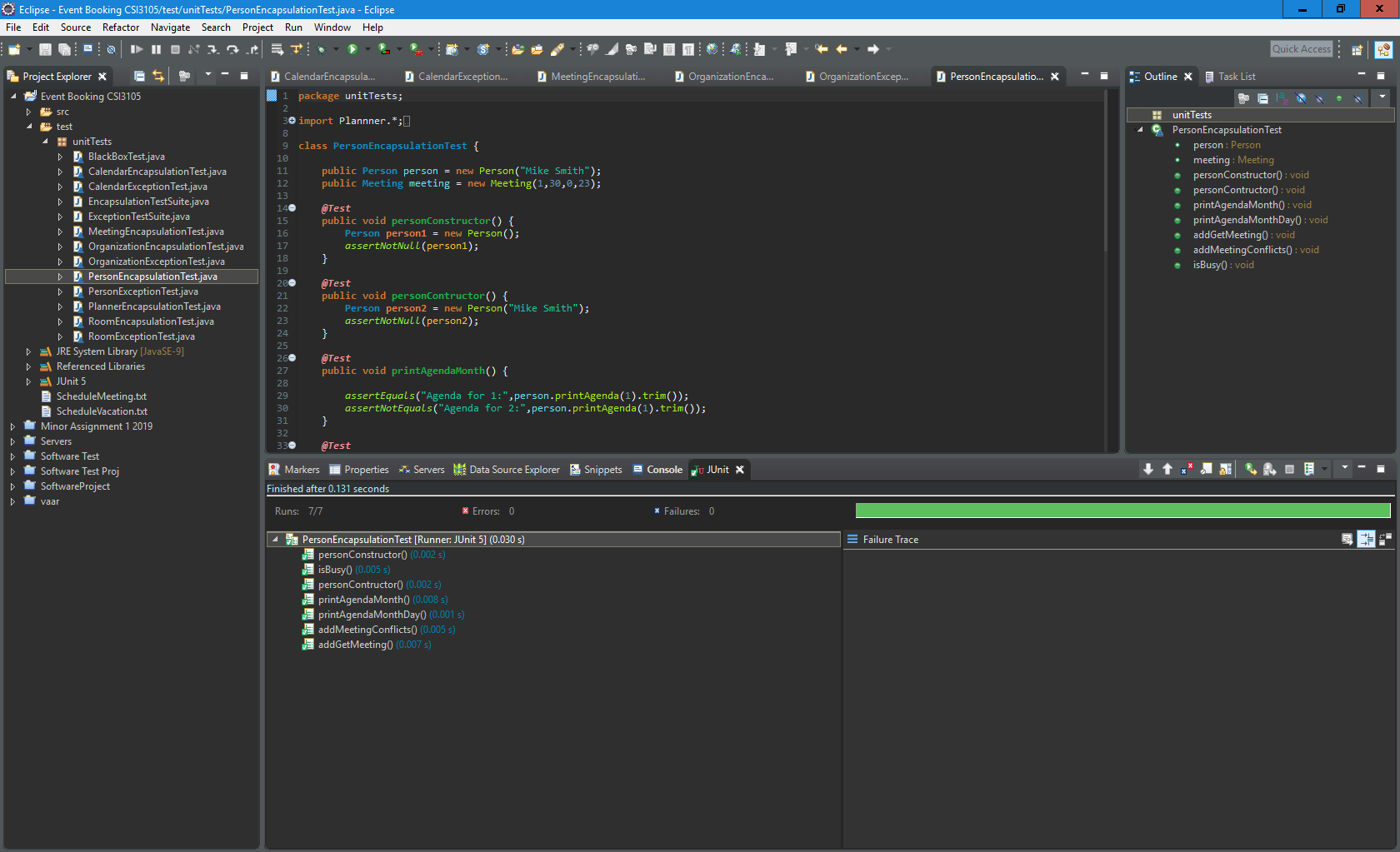
### ROOM:



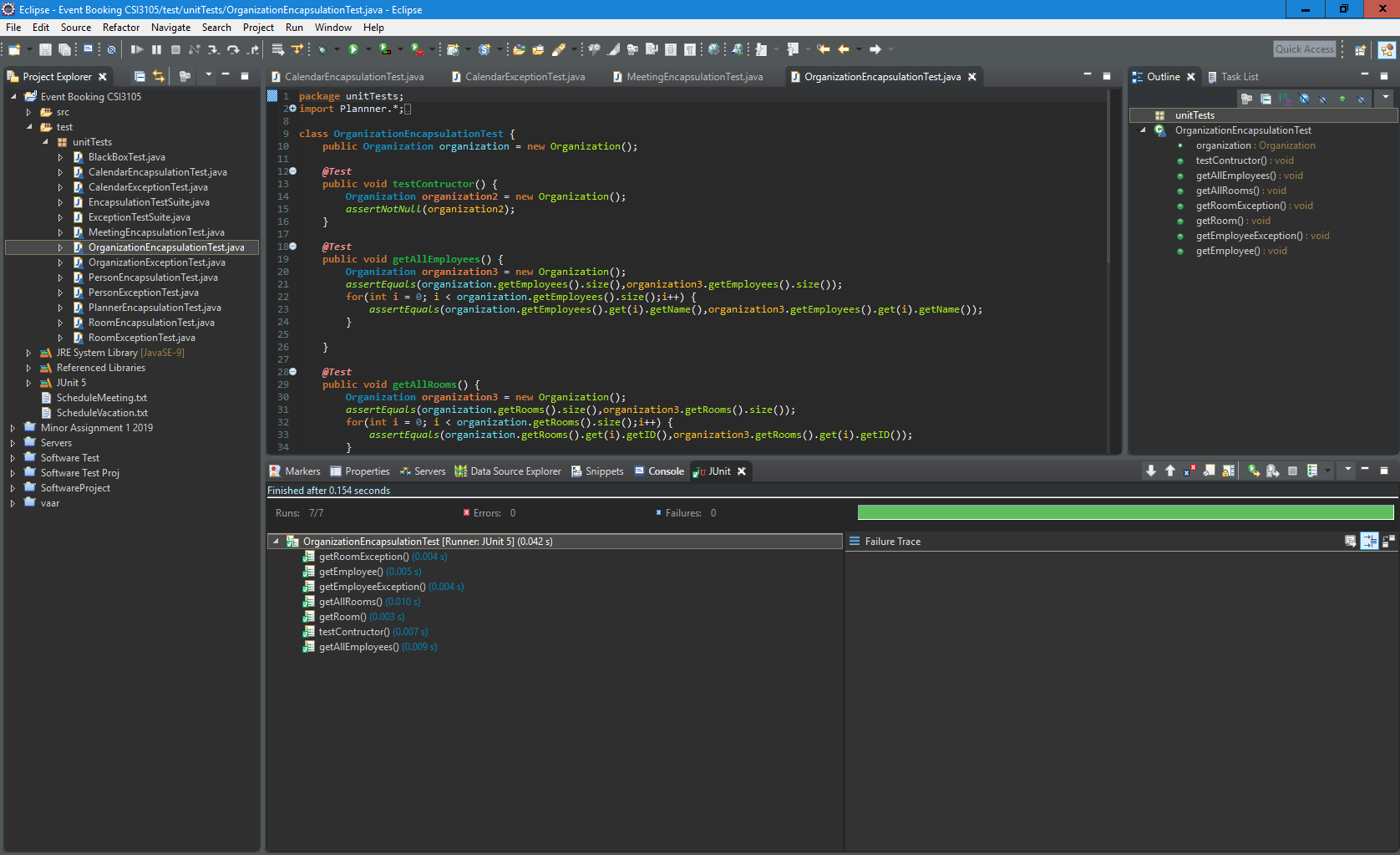
### Planner:

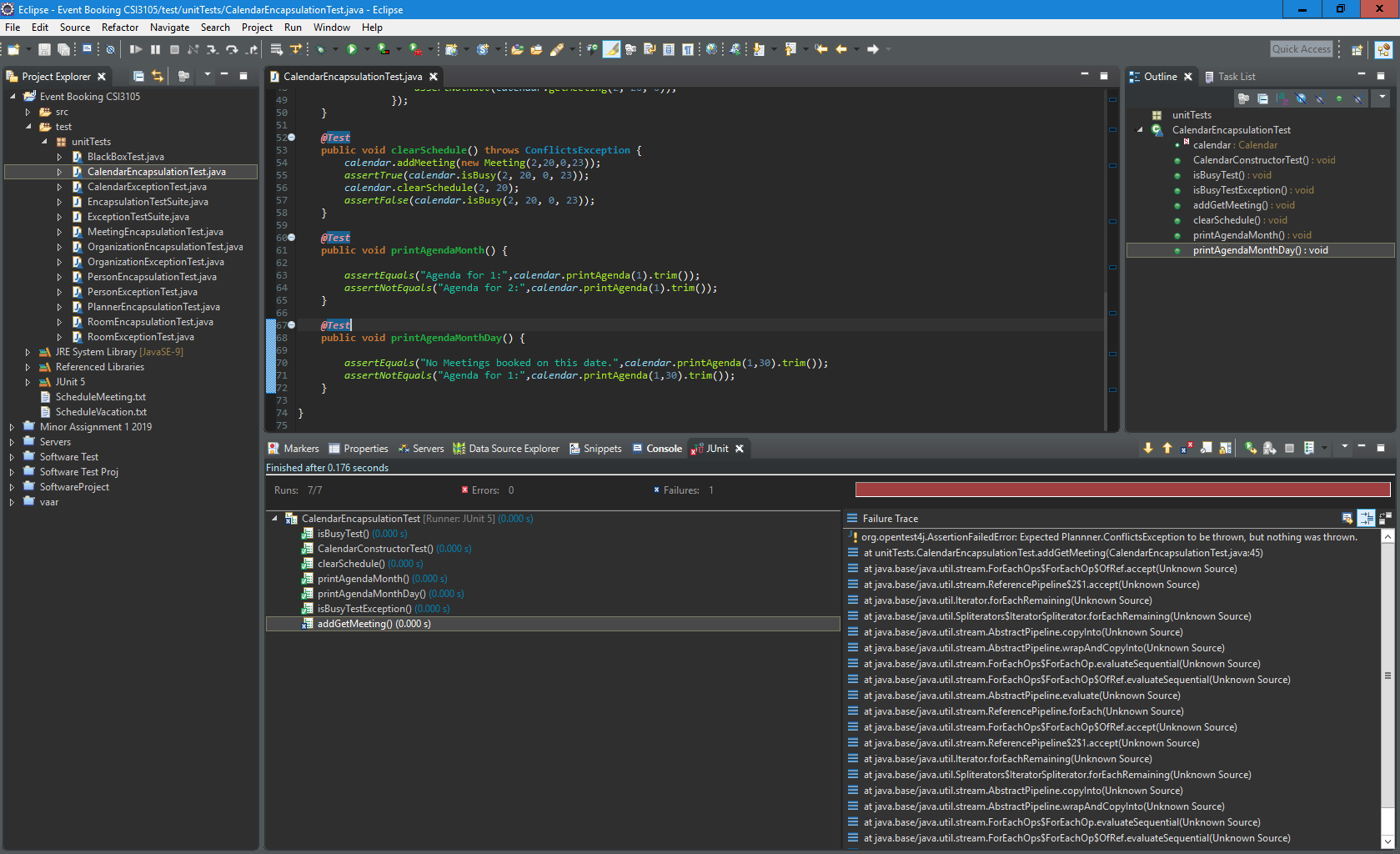


### Person:



### Organisation





### Calender:

.

# Test plan:

**T-01 = Schedule a meeting testing**

**T-02 = Book vacation dates testing**

**T-03 = Check room availability testing**

**T-04 = Check person availability testing**

**T-05 = Check agenda for a room testing**

**T-06 = Check agenda for a person testing**

**T-07 = Exit testing**

**T-01 = Schedule a meeting testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comments** |
| T-01.1 | Meeting.java  Main menu()  1.Schedule a meeting  User input “01” | After typing “01”, moving to Schedule a meeting section | As an expected result | No defect/fault  As result | Main menu integer “01” input testing |
| T-01.2 | Meeting.java();Schedule a meeting  **Month enter** | Able to enter month between1-12  User example;  input”5” | After input, move to next step: “enter day of meeting” | No defect/fault  As result | Meeting month integer “12” input testing |
| T-01.3 | Meeting.java();Schedule a meeting;  Day input. | Able to enter day between1-32; example input”12” | After input, move to next step: “enter starting hour of meeting” | No defect/fault  As result | Meeting day integer “12’ input testing |
| T-01.4 | Meeting.java();Schedule a meeting;  Starting hour of meeting input. | Able to enter hour between “0-24”  Example input”22” | After input, move to next step: “enter ending hour of meeting” | No defect/fault  As result | Meeting day hour integer”22” input testing |
| T-01.5 | Meeting.java();Schedule a meeting;  Ending hour of meeting input | Able to enter hour between “0-24”  Example input”24” | after input, display the rooms available at that time | No defect/fault  As result | Meeting end hour integer”24” input testing |
| T-01.6 | Meeting.java();Schedule a meeting;  Rooms selection | Able to select rooms example”ML13.218” | Display the next step that able to view the people who is available to attend at that time | No defect/fault  As result | Meeting room String input “ML13.218” input testing |
| T-01.7 | Meeting.java();Schedule a meeting;  Input persons who can attend to the meeting | Able to input people who is available from the list; example ”Ashley Martin” | After input, in this step that able to select multiple people until type ”done” | No defect/fault  As result | Meeting room String input “Ashley Martin” input testing |
| T-01.8 | Meeting.java();Schedule a meeting;  Input of description for the meeting | String, int, float types of input allow | After input, back to the main menu section | No defect/fault  As result | String, int, float types of input testing |
| T-01.3.1 | Meeting.java();Schedule a meeting;  Day input with string  Example ”abcde” | Unable move to next step, message display error | Error message display: “Please enter the number that corresponds to the option that you want to proceed with.” | As an expected result with error message display | Error testing |
| T-01.4.1 | Meeting.java();Schedule a meeting;  Starting hour of meeting input.  Example”50” | Unable move to next step, message display error:  Must be 0-24 digital number | Error message display: “Illegal hour.” | As an expected result with error message display | Error testing |
| T-01.6.1 | Meeting.java();Schedule a meeting;  Rooms select input rooms is not in the list example: MT18.111 | Unable move to next step, message display error:  “must select the room that is provide in the list” | Error message display :” requested room does not exist” | As an expected result with error message display | Error testing |

**T-02 = Book vacation dates testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-02.1 | Meeting.java  Main menu()  Input value”02” to enter Book vacation dates | After entering 02 then move to book vacation dates section | Move to next step | No defect/fault  As result | Able to enter integer |
| T-02.2 | after enter book vacation dates then Enter the month that the vacation starts between 1 -12  example “05” | Able to move to next step | Allow integer and move to next step | No defect/fault  As result | Able to enter integer |
| T-02.3 | after enter vacation dates then move to next section. Enter the day vacation between 1-31  Example “22” | Able to move to next step | Allow integer and move to next step | No defect/fault  As result | Able to enter integer |
| T-02.4 | After T-02.3. enter the vacation ends month between 1 – 12  Example “06” | Able to move to next step | Allow integer and move to T-02.5 | No defect/fault  As result | Able to enter integer |
| T-02.5 | After T-02.4. Enter the vacation ends day between 1-31  Example “25” | Able to move to next step | Display the detail of book cation | No defect/fault  As result | Able to enter integer |
| T-02.6 | After T-02.5. Enter the person name from the list that who is talking the vacation  Example “Edith Cowan” | Some message display from the screen | If the day is not existing. then display message “Day does not exist”. | No defect/fault  As result | Able to enter String |
| T-02.2.1 | after enter book vacation dates then Enter the month that is out of range  example “18” | Error message will display on the screen | Month does not exist display from the program | Month range 1-12 | Month must between 1-12 |
| T-02.3.1 | Enter the vacation day that is out of range  Example “38” | Error message will display on the screen | Day does not exist show in the program | Day of range 1-30 | Day must between 1 -30 |

**T-03 = Check room availability testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-03.1 | Main menu()  Input value”03” to enter check room availability | After entering 03 then move to Check room availability section | Move to next step with no error | No defect/fault  As result | Able to enter integer |
| T-03.2 | After T-03.1. enter the month of the meeting between 1-12. Example”11” | Able to move to next step | Able to enter “11” then more to next step | No defect/fault  As result | Able to enter integer |
| T-03.3 | After T-03.2. enter the meeting dating for checking between 1-31  Example”11 | Able move to next step | Move to next step | No defect/fault  As result | Able to enter integer |
| T-03.4 | Enter the starting hour after entering the day  Example “10” | Able move to next step | Move to next step | No defect/fault  As result | Able to enter integer |
| T-03.5 | Enter the ending hour of the meeting between 0-23  Example “05” | After enter value “05”. Then should move to next step | Display the result for the check room availability details  Message” The rooms available at the specified time are: Meeting starts before it ends.” | No defect/fault  As result | Display the result you enter |
| T-03.3.1 | String value input test for month enter.  Example “HTT” | Error message display | Error message display please enter number from1-31. | Fault testing method | Month method not allow string |

**T-04 = Check person availability testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-04.1 | Main menu()  Input value”04” Check person availability | After typing “04”, moving to check person availability | As an expect result | No defect/fault  As result | Able to enter integer |
| T-04.2 | After T-04.1. enter the month of the meeting between 1-12. Example”01” | After typing “01”  That should able to move next step | As an expect result | No defect/fault  As result | Able to enter integer |
| T-04.3 | After T-04.2. enter the day of the meeting between 1-31. Example”13” | after input that should move to next step | As an expect result | No defect/fault  As result | Able to enter integer |
| T-04.4 | After T-04.3. enter the starting hour for the meeting between 0-23.  Example “16” | Move to next step | As an expect result | No defect/fault  As result | Able to enter integer |
| T-04.5 | After T-04.4. enter the ending hour of the meeting 0-23.  Example “20” | Move to next step | The program will autumnally list the people who can attend the meeting on that time. | No defect/fault  As result | Able to enter integer |

**T-05 = Check agenda for a room testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-05.1 | Main menu()  Input value”05” to enter Check agenda for a room | Able enter to section 5(Check agenda for a room) | As an except result | No defect/fault  As result | Able to enter integer |
| T-05.2 | After entering to check agenda for a room. Then enter the month between (1-12). Example “08” | Able move to next step | As ab except result | No defect/fault  As result | Able to enter integer |
| T-05.3 | Enter the day for check agenda for a room between (1-31) or enter all  Example “08” | Able move to next step | List the room that you feel interested. | No defect/fault  As result | Able to enter integer |
| T-05.3.1 | Enter the day for check agenda for a room between (1-31) or enter all  Example “all” | List all the room that is available | As an except result | No defect/fault  As result | Able to enter integer/string = all |
| T-05.4 | Enter the room id as shown in the list  Example “ML13.218” | Move to next step | Message show that the agenda for 8: | No defect/fault  As result | Able to enter String of room ID |

**T-06 = Check agenda for a person testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-06.1 | Main menu()  Input value”06” to enter check agenda for a person | Able enter to section 6(check agenda for a person) | As an expect result | No defect/fault  As result | Able to enter integer |
| T-06.2 | Enter month between(1-12)  Example “11” | Move to next step | As expect result | No defect/fault  As result | Able to enter integer |
| T-06.3 | Enter the day(1-31) or enter all  Example “10” | Move to next step | As expect result | No defect/fault  As result | Able to enter integer |
| T-06.4 | After enter the day. The program will shown the person who you feel interested from the list example” Justin Gardener” | Move to next step | After enter the person name “Justin Gardener” then the message will shown in the program” No Meetings booked on this date.” | No defect/fault  As result | Result |

**T-07 = Exit testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Class/Method under Test** | **Expected Result** | **Actual Result** | **Description of defect/fault** | **Comment** |
| T-07 | Main menu()  Input value”0” to exit the program | Exit the program | Exit the program and shown the build total time execution | No defect/fault | Return to the program |

Test case

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Option | Month | Day | Start | End | Room | Attendee | Description |
| 1 | 1 | 05 | 26 | 08 | 20 | ML13.218 | Jaci Johnston  Travis Colin  Ashley Martin | SCHOOL(String test) |
| 2 | 2 | 05 | 12 | 6 | 20 | - | Jacquie Martin  Jaci Johnston | 123456(integer)test |
| 3 | 3 | 08 | 05 | 5 | 10 | - | - | Check room availability |
| 4 | 4 | 11 | 12 | 06 | 08 | - | - | People who available to attend at that time |
| 5 | 5 | 07 | 20 | - | - | ML13.218 |  | Check room abailable |
| 6 | 6 | 4 | - | - | - |  | Ashley Martin | Check agenda for a person |
| 7 | 7 | - | - | - | - | - | - | Exist the program |
| 8 | 2 | 5 | 25 | 6 | 10 | - | Ashley Martin |  |
| 9 | 4 | 2 | 10 | 10 | 20 | - | - | People who available to attend at that time |
| 10 | 5 | 1 | 20 | - | - | ML18.330 | - |  |

Output:

1. Schedule meeting: month is 5 day is 26,time 08-20,Room ML13.218: description:SCHOOL

Attending: Jaci Johnston, Travis Colin, Ashley Martin

1. Day does not exist
2. The rooms available at the specified time are:JO18.330,JO7.221,JO15.236,JO1.230,JO34.536,JO19.230,ML5.123,ML18.330,ML21.520,ML13.213,ML21.310,ML13.218
3. The people available to attend at that time are:Justin Gardener,Ashley Matthews,Mary Jane Cook,Rose Austin,Mike Smith,Helen West,Steven Lewis,Edith Cowan,Mark Colin,Jacquie Martin, Jaci Johnston,Travis Colin, Ashley Martin
4. No Meetings booked on this date.
5. No Meetings booked on this date.
6. Program exist, display run time
7. Day does not exist.
8. The people available to attend at that time are:Justin Gardener,Ashley Matthews,Mary Jane Cook,Rose Austin,Mike Smith,Helen West,Steven Lewis,Edith Cowan,Mark Colin,Jacquie Martin, Jaci Johnston,Travis Colin, Ashley Martin
9. No Meetings booked on this date.

# Defect report:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Test Type  (Class/Method) | Test Result | | Rate | | | Pass | Fail | Pass | Fail | | Calendar.java | 7 | 1 | 87.5% | 9% | | Conflicts Exception.java | 7 | 2 | 81.7% | 18% | | Meeting.java | 7 | 1 | 87.5% | 9% | | Organization.java | 7 | 2 | 81.7% | 9% | | Person.java | 7 | 1 | 87.5% | 9% | | Planner.java | 7 | 0 | 100% | 0% | | Room.java | 7 | 0 | 100% | 0% | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Subject Test Name** | | | | | | | | | | | | **Last** | | | **Open Defects** |
| **Failure** | **Status** | **Run** |
| Leap year Feb29 unable to book event | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Issues booking between 11:00pm–11:59pm | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Booking associated with 30th November | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Input checking: Year/Month/Day | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Input checking: Booking event | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Input checking: error messages | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Input checking: sensitivity Case | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Booking associated with certain days of month | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| Vacation booked start date can be 32nd | | | | | | | | | | | | 01/05/2019 | Passed | 07/05/2019 |  |
| **Main menu input 01 : Book vacation dates testing** | | | | | | | | | | | | | | | |
| **Month** | **Day** | | **Start Time** | | **End Time** | | **Room** | | **Attendee** | | | **Comments** | **Status** | **Date** |  |
| ✔ | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 02: Book vacation | | | | | | | | | | | | | | | |
| ✔ | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 03: check room availability | | | | | | | | | | | | | | | |
| ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 04: check person availability | | | | | | | | | | | | | | | |
| ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 05: check agenda for a room | | | | | | | | | | | | | | | |
| ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 06: check agenda for a person | | | | | | | | | | | | | | | |
| ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | Passed | 10/5/2019 |  |
| Main menu input 07: exit | | | | | | | | | | | ✔ | | | | |

# Test case Results:

Verifying the problems identified by the previous team of testers

* Feb 29 is valid but unable to book any event on that day.
  + No year input available and Feb 29 is by default marked invalid
* Issues associated with booking an event between 11.00pm –11.59pm
  + Hours Not valid from 23 (11 pm)
  + No facility to count midnight of day3.
* Bookings associated with 30th November
  + No issues found with booking for 30thof November
* Input checking and Ambiguous error messages
  + Input checking is done without any error messages
  + The control simply transfers to the main menu to restart the booking. Doesn’t tell the user what is wrong
* Bookings associated with certain days of the month
  + No Month has allowed to book for 31stwhere applicable
* A vacation can be booked with a start date of the 32nd day of a month
  + Vacation can be booked starting from 32ndof the month

Other identified defects:

* Vacation cannot be booked starting one month and ending in other gives with days not found error

1 of 2

# Conclusions:

Most of the issues identified by the previous team are present within the program. There is one error that was not found which is the issue booking on the 30th of November.

* The program does not consider Leap years
  + Every 3 years the program will be incorrect by one extra day
* Any time after 10:59PM is invalid and the rollover to midnight doesn’t work correctly on the 3rd day of the month
  + This may be a minor issue unless the company has serious deadlines to meet, operates for 24 hours or has to work at night or if the company has employees working overseas in different time zones
* Some error messages are either not present or lacking e.g. not allowing the user to know that their input is wrong
  + This leads to user confusion, frustration and misunderstandings
* Months with 31 days are unable to book on the 31st
  + This is a problem because bookings can’t be made 7 days in a year.
* Vacations can be booked on the 32nd day of the month, a vacation can also throws an error when the vacation is in a different month.
  + This is a problem because vacations are very likely to carry other to another month and the vacation is booked on a day that doesn’t exist.

Now that these errors have been identified the debugging process can begin then further testing is required. Since these errors will heavily affect the program the current version of the program is unsuitable to be released until these errors are fixed as it drastically effects the use of the program in its intended environment. With the current build number of days the program will be more and more incorrect each year, the program is intended to be used everyday which makes this a major issue.

Fault testing didn’t seem to have many issues however it did demonstrate the presence of many of the issues identified by the team previously such as not considering leap years and 11:00 to 11:59 PM being invalid hours.

Integrated testing went smoothly demonstrating that classes, functions and objects are initialised properly and important processes are executed correctly.

# MINUTES OF MEETING

## Meeting 1

Team: Bryce, Chak, Tinashe

Date: 7th of April

Time: 9:30PM

Venue: Skype

Participants: Bryce, Chak, Tinashe

Apologies: NULL

Absent: NULL

Discussion

* Templating the assignment
* Looking through the program
* Defect report draft
* Assigning Tasks

GENERAL

We set up a drop box and went through tasks to assign to one another to set out a schedule

NEXT

We will finish off creating a document for the assignment and look over the defect report draft

|  |  |  |  |
| --- | --- | --- | --- |
| ACTION | WHO | DUE DATE | STATUS |
| Documentation + Dropbox | Bryce | 12th of April | CLOSED |
| Defect Report | Chak | 12th of April | CLOSED |
| Plot out test cases | Tinashe | 12th of April | OPEN |

## Meeting 2

Team: Bryce, Chak, Tinashe

Date: 20th of April

Time: 8:32PM

Venue: Skype

Participants: Bryce, Chak, Tinashe

Apologies: NULL

Absent: NULL

Discussion

* Testing Approach
* Design test cases part 1
* Start coding test cases part 1

GENERAL

We started on the deliverables for the assignment part 1, test cases and testing approach

NEXT

We will continue on with the test cases for part 1 and the documentation after that we will move onto part 2

|  |  |  |  |
| --- | --- | --- | --- |
| ACTION | WHO | DUE DATE | STATUS |
| Testing Approach + design test cases | Bryce | 5th of May | OPEN |
| Coding Test cases part 1 | Chak | 5th of May | OPEN |
| Design + Coding test cases part 1 | Tinashe | 5th May | OPEN |

## Meeting 3

Team: Bryce, Chak, Tinashe

Date: 9th of May

Time: 7:30PM

Venue: Skype

Participants: Bryce, Chak, Tinashe

Apologies: NULL

Absent: NULL

Discussion

* Allocate part 2
* Allocate part 3
* Continue documentation
* Allocate Presentation

GENERAL

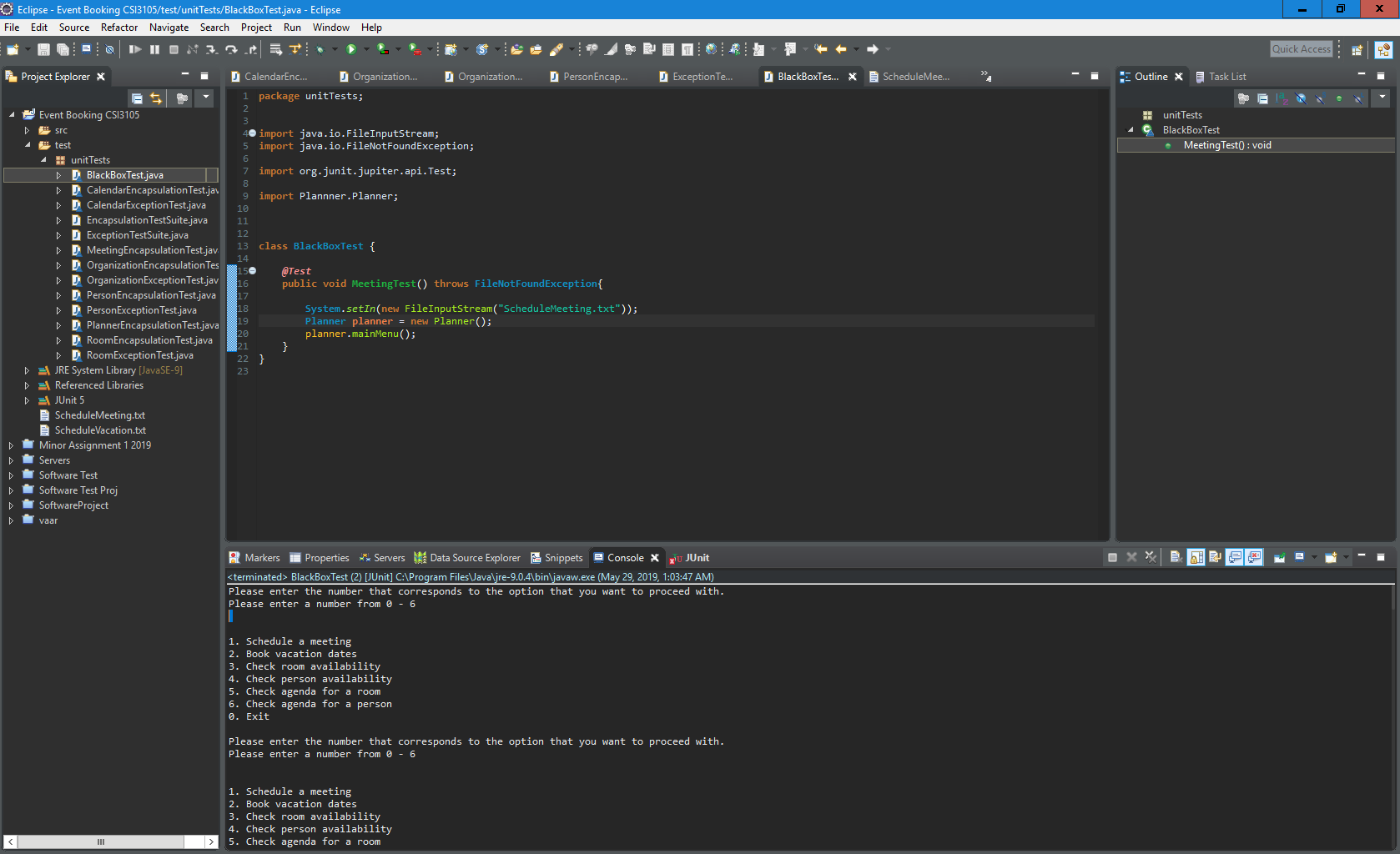
We set up a drop box and went through tasks to assign to one another to set out a schedule

NEXT

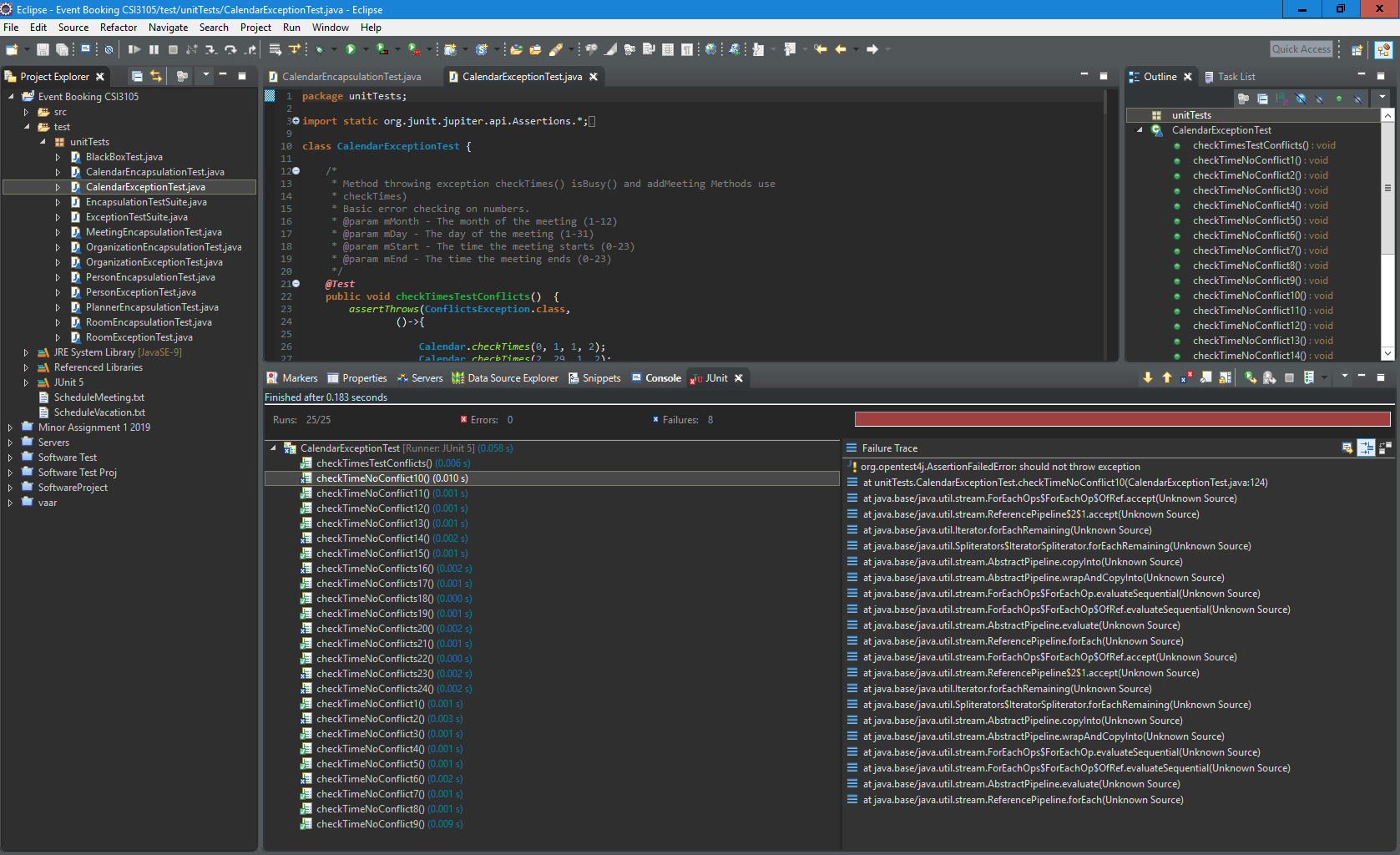
We will finish off creating a document for the assignment and look over the defect report draft

|  |  |  |  |
| --- | --- | --- | --- |
| ACTION | WHO | DUE DATE | STATUS |
| Documentation + Presentation | Bryce | 12th of April | CLOSED |
| Part 2 | Chak | 12th of April | CLOSED |
| Part 3 | Tinashe | 12th of April | OPEN |

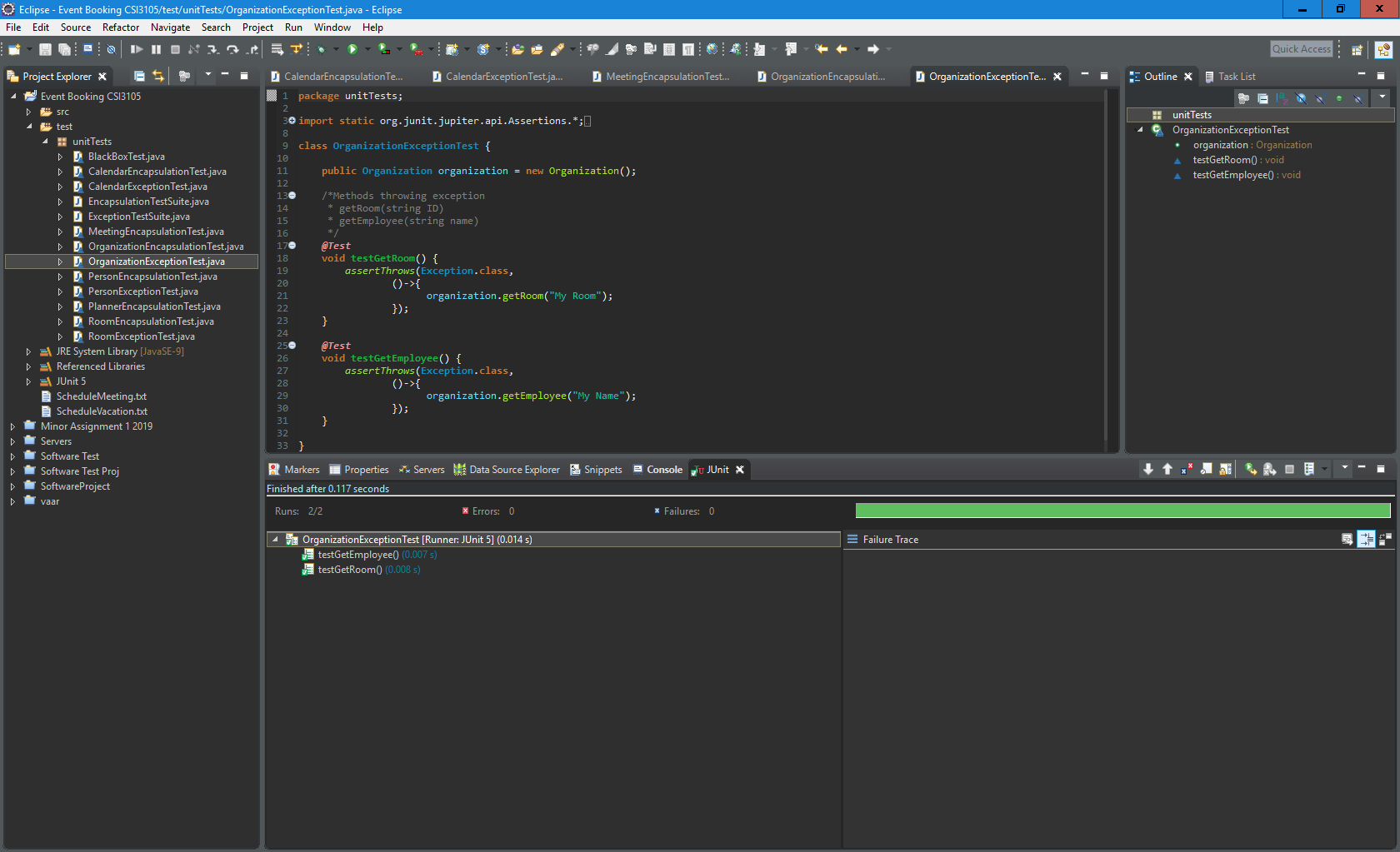
# APPENDIX 1 (Black-Box Test):



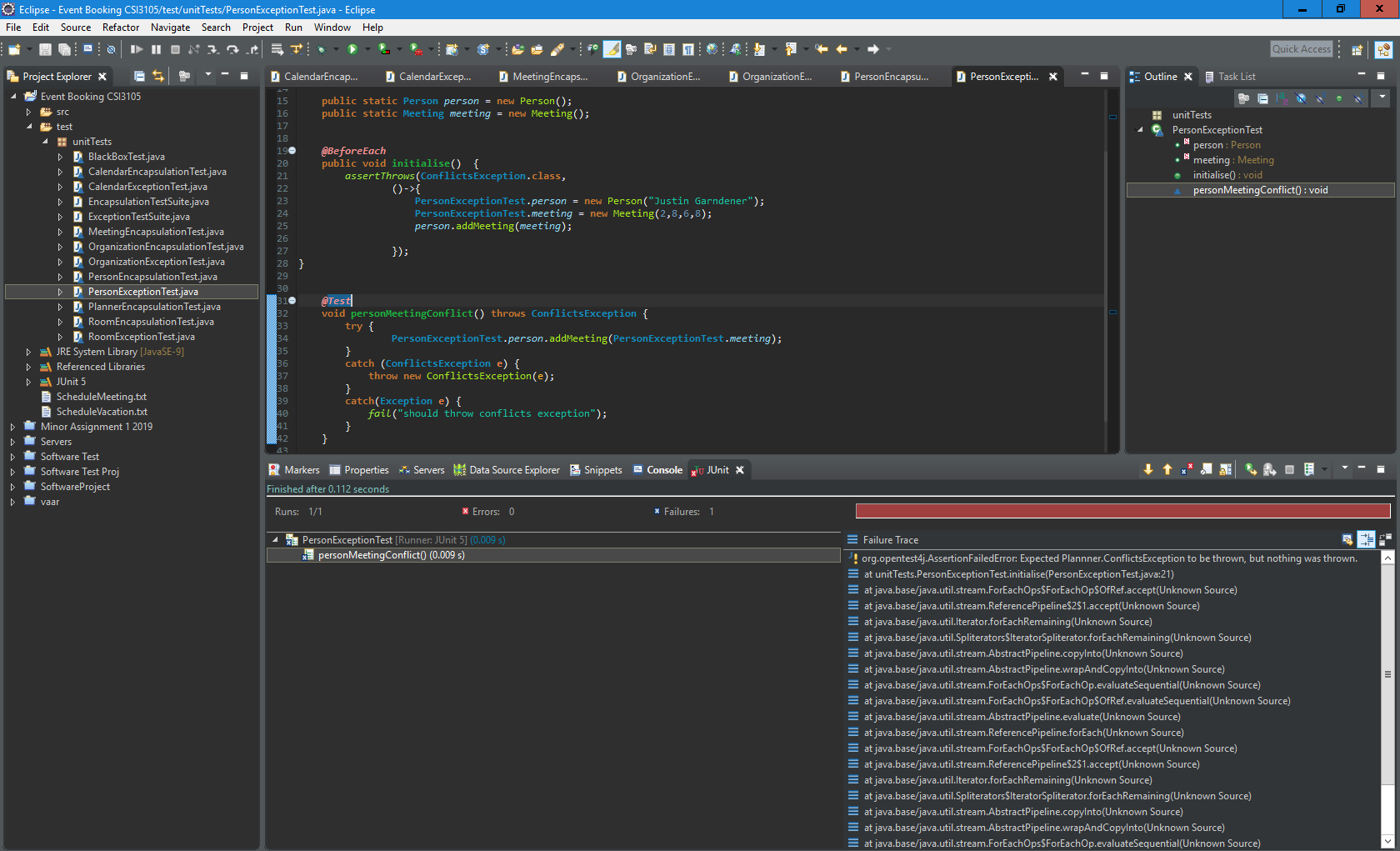
# Appendix 2 (Integrated Tests):

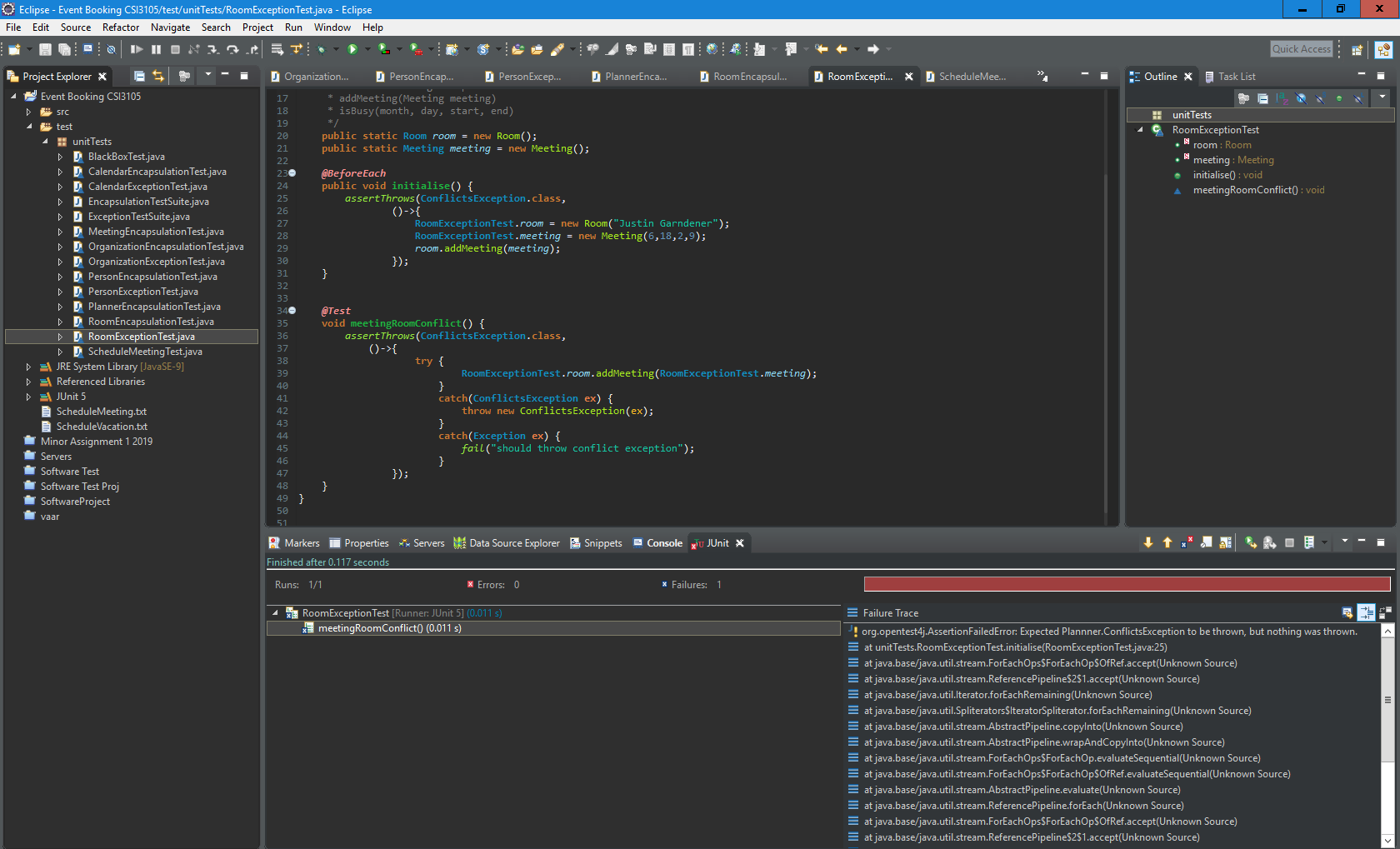
Calendar Exception testing:

Organisation Exception testing:



Person Exception Testing:



Room Exception testing: