CSC8011 Programming Coursework: Track and Trace

Learning Outcomes

- To be able to read a case study, and sectioned instructions and translate a problem into a viable programmatic solution
- To be able to use a development process with version control to develop a solution to the described problem
- Apply Java Obejct Oriented fundamentals to a project in order to produce a solution to the problem

Skills Outcomes

- Use applied Object Oriented Programming to develop a solution to the problem
- Use correct programming and software engineering techniques to develop a solution
- Use control flow to build a usable command line interface
- Describe primitive tests for the program
- Be able to reflect on the development process and highlight processes in the reflective cycle to describe for preparing for future assignments

Case Study

As we continue living in a COVID-19 world, the reality of Track and Trace in the United Kingdom is something that has become of daily lives. Track and Trace is a system that is intended to alert those whose have been recorded in the same establishment on the same day as someone who has tested positive for Corona Virus. The purpose of this is so that the afflicted person is to quarantine and anyone who else who has came into contact with them in an establishment on a given date.

This is quite a complex process and is difficult to implement programmatically without constant human intervention to monitor the records . Furthermore, this requires users to fill in correct information and be trusted to contact those in their party who also visited the establishment on a given date, since it is only required for one person to record their attendance in the party

The purpose of the following system is to manage incoming data using a naive and primitive command line recording system written in Java. This system would require an engineer to input information recorded by an establishment or even hired by the establishment themselves. The process would be that the engineer would record an ‘Event’, which is the recording of ‘User’ information at a given ‘Establishment’. This recording would include information such as; recorded ‘User’ name, DOB, contact information, and party size. Information recorded about the event would include; ‘Event’ date occurred, time occurred, User, and Establishment. An ‘Establishment’ recording would include; name of establishment , address, postcode, and maximum occupancy

The produced solution isn’t a fix to the problems that Track and Trace have . Instead it’s purpose is to get students to think about problems occurring in the real world and to start thinking of programmatic solutions to these problems.

If you have any concerns about the chosen topic then please get in touch with Jordan Barnes at jordan.barnes@newcastle.ac.uk

Restrictions:

- All input and output must be handled within the `IO` class. Any printing or input which occurs in other classes will be suffer penalties.
- You must **not** store User object information in any collection. This includes `User[]` or `ArrayList<User>`.

Section 1: Class and Object Design

Class and Object Design

You have been given the following classes:

- `Event`
- `User`
- `Establishment`

You must define the aforementioned classes using correct programming class design principles. This includes encapsulation of variables, constructor overloading, composition, and overriding methods.

Define the `Event` class with appropriate fields, methods, constructors

- `Event(User, date, partyNumber, Establishment)`
- `Event(User, partyNumber, Establishment)`
- Store and Retrieve information about an event. Including its:
  - User
  - eventDate
  - eventTime
  - partyNumber
  - Establishment
- Behaviour which includes:
  - printing of the instance details so that it looks like this in the console window

```
Event ID: 1638925269 | Recorded User
    Name Jordan Barnes
     Date of Birth 24/09/1995
     Email jordan.barnes@newcastle.ac.uk
     Contact Number 07123456789
     Age 24
 Date 20/06/2020
 Time 12:00
 Party Size 2
 Establishment
        Name: Some Coffee House,
        Address: 1 King Street FA1 3KE
```

Define the `User` class with appropriate fields, methods, constructors

- `User(name, dob, emailaddress, phonenumber)`
- Store and Retrieve information about a user. Including their:
  - name
  - date of birth
  - contact details (email, phone number)

    - contact age
- Behaviour which includes:
    - calculating the age of a user when given a date of birth and storing it as an instance variable
    - printing of the instance details so that it looks like this in the console window

```
Name Jordan Barnes
 Date of Birth 24/09/1995
 Email jordan.barnes@newcastle.ac.uk
 Contact Number 07123456789
 Age 25
```

    - single line printing of the `name` and `phone number` so that it looks like this

```
Jordan Barnes | Contact Number 07123456789
```

Define the `Establishment` class with appropriate fields, methods , constructors:

- `Establishment(name, firstLineAddress, postCode, maxOccupancy)`
- `Establishment(name, address(firstLineAddress, postCode), maxOccupancy)`

This class should have the following functionality:

- Store and Retrieve information about an establishment. Including its:
    - name
    - address (firstLineAddress, postCode)
    - maximum occupancy
- Behaviour which includes:
    - printing of the instance details so that it looks like this in the console window

```
Name: Some Restaurant Place,
Address: 1 Queen Street FA2 3KE
```

Section 2: Program Control and IO

You have been given the following classes:

- `Controller`
- `IO`

Define the `Controller` class with appropriate methods and constructors

- `Controller()`
- `Controller(String establishmentCSVFileURI)`

This class should have the following functionality:

- store and retrieve information, including:
- list of establishments
- list of events
- behaviour which includes:
    - ability to parse a csv file with the given format, and store the values as an appropriate object

```
name,firstAddressLine,postcode,occupancy
Some Coffee House,1 King Street,FA1 3KE,5
Some Restaurant Place,1 Queen Street,FA2 3KE,15
Some Book Place,1 Regal Street,FA4 3KE,20
```

    - add an `Establishment` object to the list of establishments, if the object already exists then return false
    - add an `Event` object to the list of events, if the object already exists then return false

Define the `IO` class with appropriate methods and constructors

- `IO()`
- behaviour which includes:
    - instantiating an instance of Controller that is private and immutable upon creation of an `IO` object

Section 3: Command Line Interface

This section is about expanding the `IO` class so that it offers a command line interface, and handle the input and output of the program. Expand the class to include the following:

- a command line program that takes in numerical user input to select options in a menu and doesn't terminate unless an option is chosen in the menu. The menu should include the following options:
    - 1. Record an Event
    - 2. Add Establishment
    - 3. Filters (Expanded upon in section 4), sub-menu ->
        - 1. Records by Establishment
        - 2. Records by Date
        - 3. Records by Name
        - 4. Go Back
    - 4. Print Events
    - 5. Print Establishments
    - 6. Exit the program
- check for valid input:
    - check that a date of birth is not in the future
    - check for a valid email string by inclusion of @ symbol
    - check for valid number of 11 characters long and doesn't include international calling codes
- main method which does the following:
    - creates a new instance of IO and runs the program as normal
    - creates a new instance of IO and runs a debug method
- debug method. This method should include the following examples of basic testing of your program:
    - hard code several user objects
    - hard code several event objects with establishment objects
    - use valid, boundary, erroneous and extreme value examples for testing

Section 4: Filtering Information

This section is about offering filtering options to the stored data about events, users, and establishments that have been stored in the system . Expand the `Controller` class so that it offers the following functionality:

- filter the events and present the users who have visited a given establishment by String input of an Establishments name. Below is some pseudocode of whats expected

```
filterEventByEstablishment(nameOfEstablishment)
{
   Collection filteredRecords

   getEventRecords{
       addFilter
   }
   return filteredRecords
}
```

- filter events that occurred at a given date by String input of a valid date format of dd/mm/yyyy

```
filterEventByDate(date)
{
   Collection filteredRecords

   getEventRecords{
       addFilter
   }
   return filteredRecords
}
```

- filter events that a user have been recorded at by String input of a valid name and email address

```
filterEventByUser(name, email)
{
   Collection filteredRecords

   getEventRecords{
       addFilter
   }
   return filteredRecords
}
```

- you are also required to expand the `IO` class to allow for extended menu functionality for the aforementioned filters

Coursework Report

Produce a 500 word report outlining the following reflective practice:

- brief explanation of the system
- what were you thinking and feeling before you started the coursework but after reading the specification
- description of what went well and what went wrong
- what could you have done better in hindsight
- if you were given the problem again, what would you do differently

You are to write this using markdown and include this file inside your project. It should be called **reflective-report.md**

```

# Heading 1

Some text in a markdown paragraph
- List
    - sub list

## Heading 2
### Heading 3
```

For more information about how to write using markdown - see this link https://www.markdownguide.org/getting-started/

Steps for Obtaining Java Files

NUCode link - https://nucode.ncl.ac.uk/scomp/msc-computer-science/csc8011 Canvas link - NESS Link -

/coursework/csc8011-track-and-trace

1. Go to https://nucode.ncl.ac.uk/scomp/msc-computer-science/csc8011/coursework/csc8011-track-and-trace
2. Click Fork
3. Select your Portfolio Space
    - this might take a few moments
4. git clone https://nucode.ncl.ac.uk/{BXXXXXXX}/csc8011-track-and-trace.git into IntelliJ
    - replace BXXXXXXX with your B number
5. You should now have the project in IntelliJ, as well as your own

Here is a video explaining the process

Video can be found here - https://web.microsoftstream.com/video/4ecb0839-6fd9-48bc-b563-3ab2122ef011

Marking Criteria

Code 90%

- Section 1 : 20%
- Section 2 : 20%
- Section 3 : 25%
- Section 4 : 25%

Report 10%

Deliverables

Submit a .zip file which includes all of your source code and reflective report to NESS by the deadline specified on NESS and Canvas. If you do not submit on time then a late cap will be added to your mark.