Algorithms & Data Structure

Iadt | Dun Laoghaire

CA2 Report

Jullian Engracio

2016

Table of Contents

[Introduction 2](#_Toc444764109)

[Content: Updates 2](#_Toc444764110)

[Day 1: 2](#_Toc444764111)

[Day 2: 2](#_Toc444764112)

[Day 3: 2](#_Toc444764113)

[Day 4: 2](#_Toc444764114)

[Day 5: 2](#_Toc444764115)

[Day 6: 3](#_Toc444764116)

[Day 7: 4](#_Toc444764117)

[Bugs encountered: 4](#_Toc444764118)

[Solution Tried: 5](#_Toc444764119)

[Day 8: 6](#_Toc444764120)

[Day 9: 7](#_Toc444764121)

[Problems Encountered: 8](#_Toc444764122)

[Test Runs: 8](#_Toc444764123)

[Generate from file (File input) 8](#_Toc444764124)

[Print to an Output File 10](#_Toc444764125)

[Summary: 11](#_Toc444764126)

# Introduction

This project is a java program that stores data in a database. Based from my case study – Event Management System. The project is required to create, read, update and delete Events, Event Locations, and Attendees to the database. The project also have inheritance and will have the ability to read files for generating details to be used as an input and will be able to output the details to an output file.

# Content: Updates

## Day 1:

I used the program I used from previous CA. Fixed all possible bugs that I could find. Refactor some of the codes because I made changes in my database.

## Day 2:

Implemented inheritance by making a super class Personnel.java which will be take in a persons’ details such as first name, last name, email address, and number.

The attributes and behaviours will be inherited by the Attendee. NewEventLocation.java class “*has a*” Personnel object within it.

## Day 3:

Implemented file input. The program will now have a feature to generate details from an input file. Within a switch statements, the user can either choose to input the details manually or generated from the file.

The file will contain two types of details, the details for attendees and for the location. The program is designed to distinguish between which details will go on the appropriate database. E.g. Attendee details on Attendee table.

## Day 4:

Tried implementing the output feature for the program. First try was to try take the details from the input file. I tried to output the details of the attendees only, but it didn’t work out. It also prints the location details in the output file.

## Day 5:

Made a change on output file feature. Instead of printing out from the input file I decided to make a change and output the details from the database instead.

I also implemented a case statement that will allow users to choose on whether they want to output the details into the console or on an output file.

Problem(s) encountered:

* The details is not being written in the output file

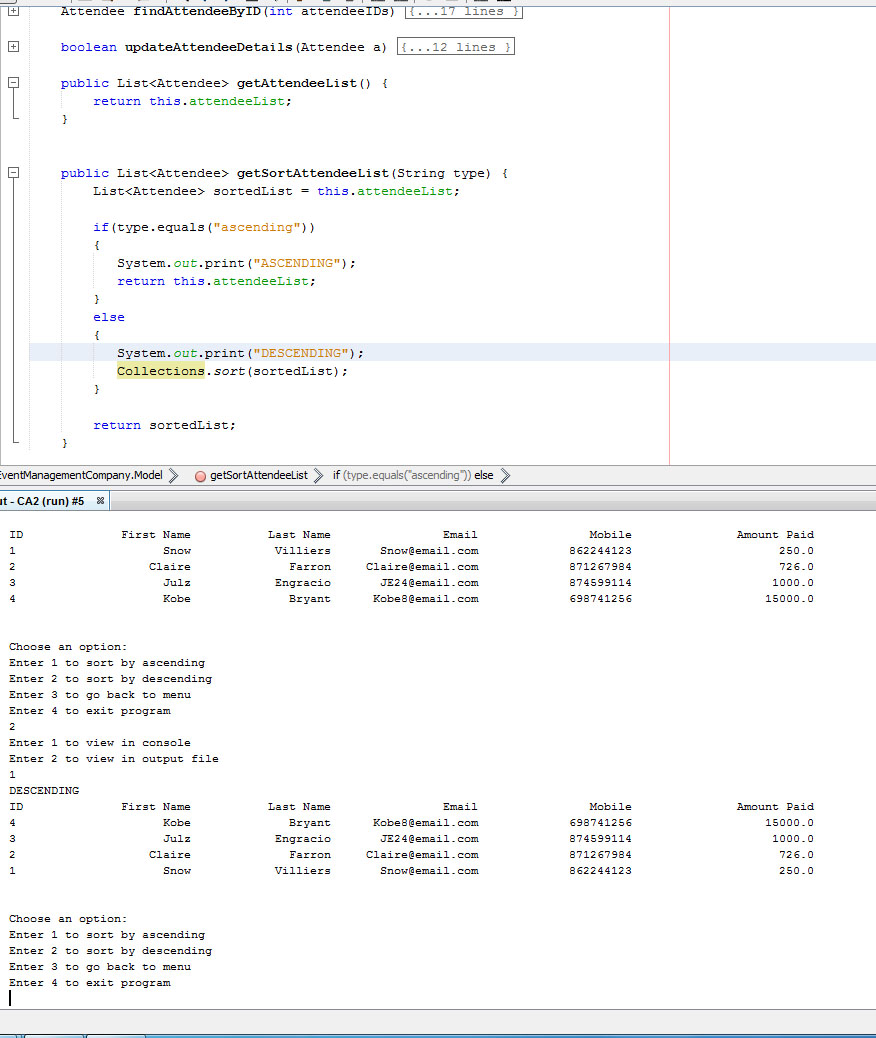
Solution(s):

* Added the code *out.close()* – this method closes the output function which stops the program for waiting for more details to be written on an output file

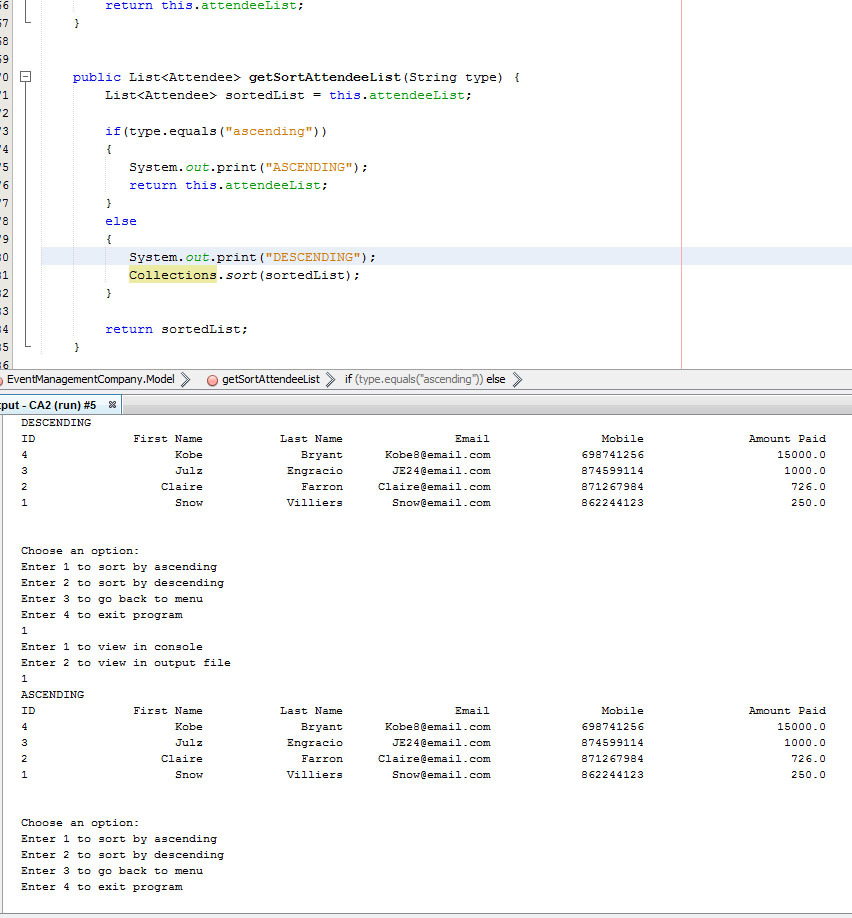
## Day 6:

I tried to implement sort algorithm, still having problems when doing the sort. First problem encountered is I can descend the order but when trying to print again in ascending order it does not work. I sorted by ID for trying out the code, but I get things fixed I will probably sort it by last name instead.

\*program run when sort by descending is called. Sorted by ID.



\*program run when sort by ascending is called after calling descending sort. It did not sort in descending order.



## Day 7:

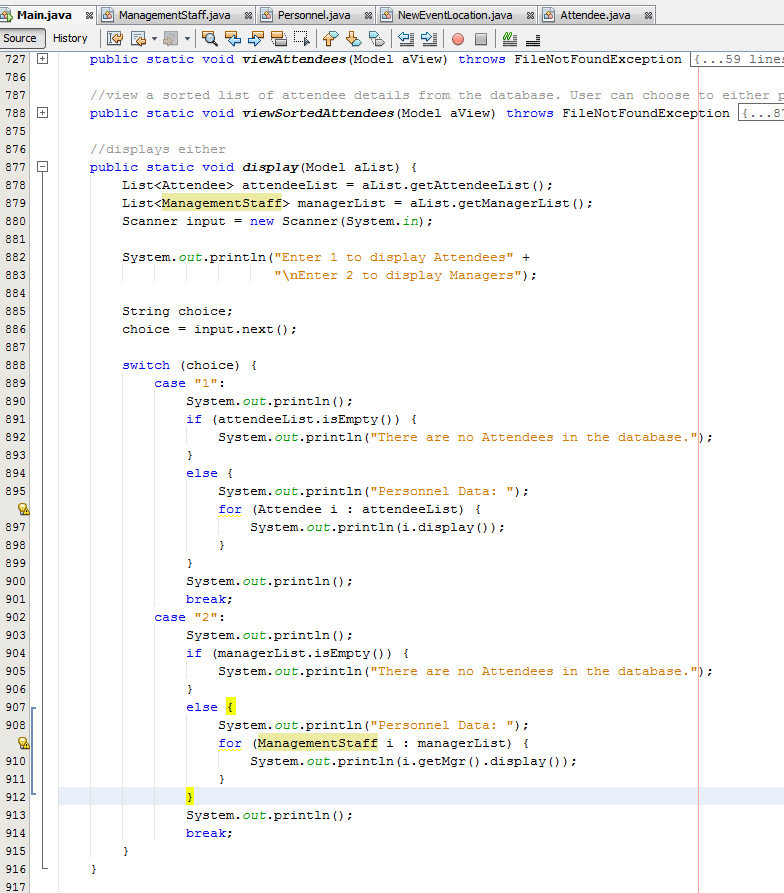
Tried implementing polymorphism, still having trouble finding my around on which data should be passed around and which ones to display.

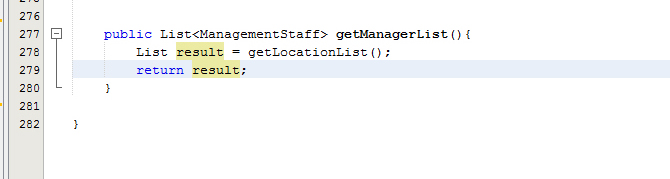
### Bugs encountered:

I can display the Personnel data of the attendees, but had trouble displaying the managers from the event location object.

### Solution Tried:

Made a new method in the Model class called getManagerList that will store the NewEventLocation arraylist of Personnel data into a manager array list. But then I ran this method and it won’t work that way.



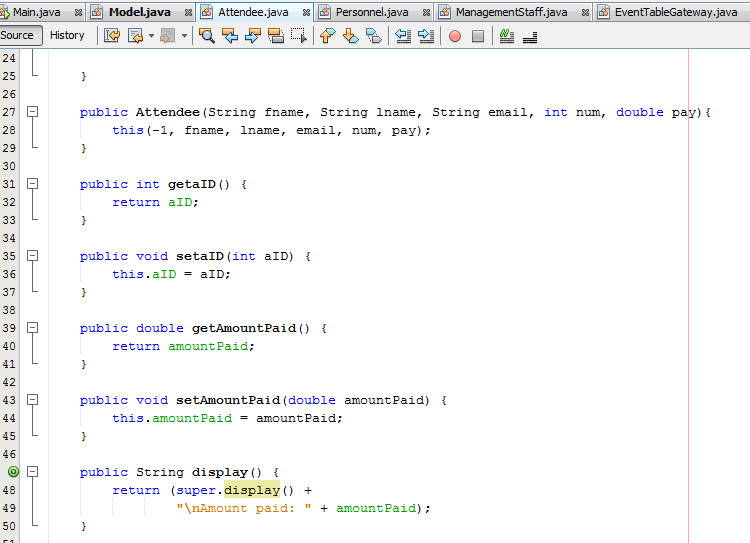


I get a run time error as a result.

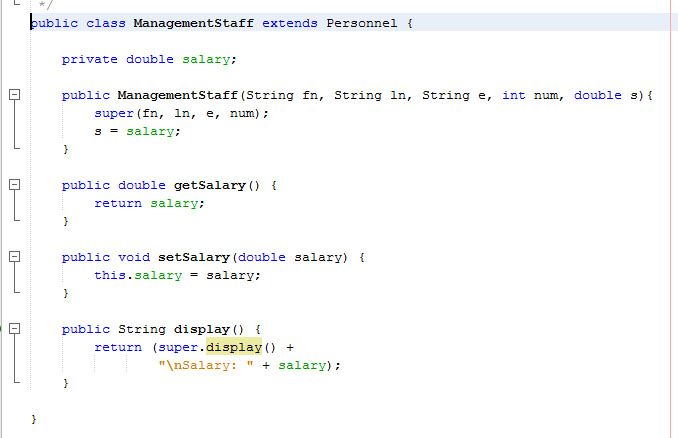
## Day 8:

Polymorphism Implemented. Both Attendee and ManagementStaff will have a display() method and a super.display() method that will display the details from the Personnel class and the unique details of each other sub class.

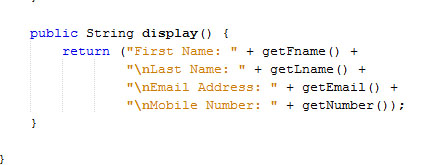
*\*Attendee class with the display() method that will return the amount paid and the details from its super class’ display() method.*



*\*ManagementStaff class with the display() method that will return the salary and the details from its super class’ display() method.*



*\*The display() method from the super class – Personnel.java*



## Day 9:

Created a manager table in the database to store the details for management staff. The purpose of this is to show an example of polymorphism. Before creating the database table, I was using an array list which is not connected to the database, but it will require for the user to type in and add data into the array list every run. Having database will just take the details from the database and store it to the array list, making it quicker to get the data.

### Problems Encountered:

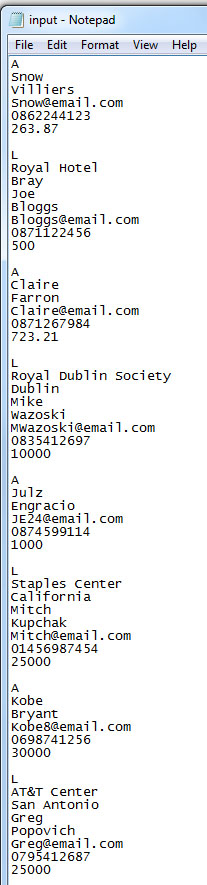
Salary column is not taking the values I enter, there was no error being thrown in the console. Making it difficult for me to figure out the problem.

## Test Runs:

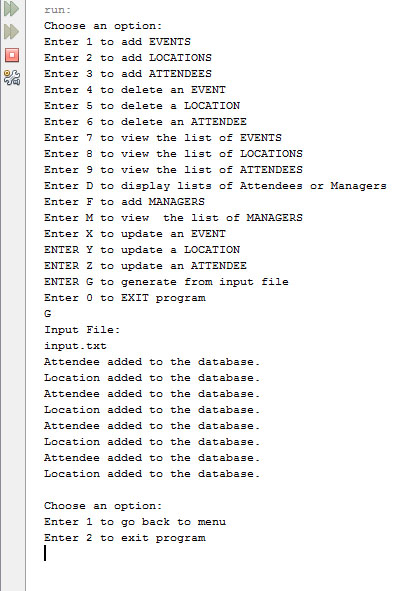
### Generate from file (File input)

Test run for file input. The file input is located at the root folder of the NetBeans project folder.

*\*File that is used as an input file. This file holds details of Attendees (“A”) and location managers (“M”).*



*\*Program run for generating details from input file.*



*\*Database table. All the details from the input file are stored in the database.*

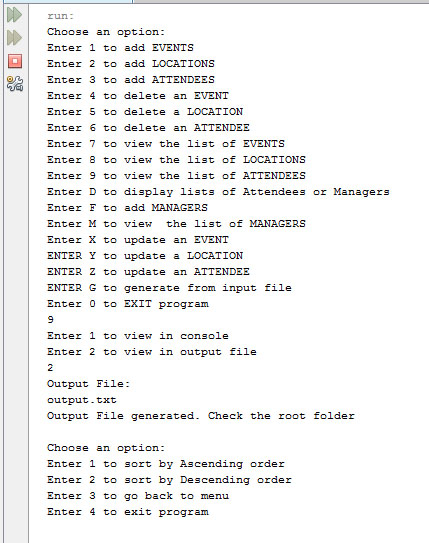




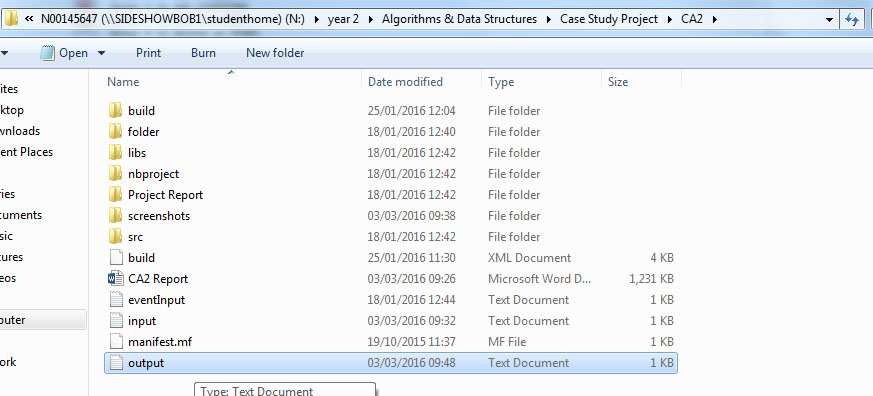
### Print to an Output File

Test run for generating the details from the database into an output file.

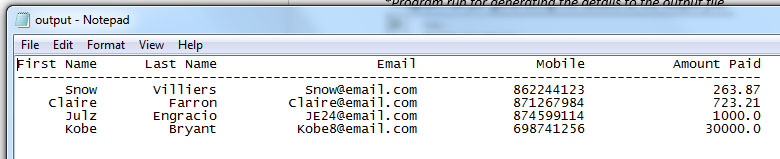
*\*Program run for generating the details to the output file.*



*\*output file generated in the root folder after the program run.*



*\*The contents of the output file. Which are the details of the attendees.*



# 

# Summary:

This program is a refined version of my CA1 assignment. The program will still be connected to the database provided by the college that was used in the first assignment.

This program contains a Super Class and two Sub class to show inheritance. One class will have a “has a” relationship – a class that contains an object from another class, which is similar to inheritance, but not completely an inheritance.

The program also have the ability to generate details from a file – File input, and will be able to output these details into an output file. The file input will contain details of Attendees and Location Managers. All the details will be added to the database. The program can only print the details of Attendee into an output file creating a list of Attendees stored in the database. . I did not implement output file for Location managers.

The program can also sort these details in descending order by last name of Attendees. But had trouble finding a way to give a user control on sorting these details on other different ways. If I had more time, I would implement different sort methods that will allow user to sort in ascending order or descending order, and will be able to choose which column to sort – by ID, first name, or last name.

In the end I still had trouble fixing the sort algorithm to the way I would have wanted it to work. The program will not have a run time error, it will just not give the user the ability to resort it back to normal or in ascending order.

The program will have polymorphism method. This method is called *display()* – this method will print out the details of the Personnel class – first name, last name, email, mobile number. The method is implemented to the two sub classes – *Attendee*, and *ManagementStaff* which will override the method and will have another unique details for each class to be added in the print out.

The Attendee will call the display() within it, and this method will call the display() method from its super class – Personnel.java. The printout in result will print the amountPaid which the Attendee class have. The process will be similar with the ManagementStaff class – will printout salary in addition.

I also implemented an interface class. That calls the *display()* method from the Super class – Personnel.