CRITERION A: PLANNING

The Scenario

Client A is a high school counselor that always have trouble assigning seats for her students. It costs much time when she does it manually; instead, she uses an old program written by myself years ago. However, that program has several drawbacks. It really costs time to produce an output of seating scheme, and the output in terminal is rather ugly. The results printed is sometimes weird as some students' names are not included – they may be blanks. Additionally, she demands that some students who like to talk with each other to be separated, but the program cannot do this and she has to separate them away manually. She has been frustrated by this dilemma for months, in a dilemma with either assigning seats manually or using that terrible program, until she found me, a student studying in CS.

After consulting with her (transcript in appendix), I proposed to design a "seating assignment" program that helped her assigning seats. From her perspective, it saves plentiful amount of time on this trivial task; she can also select her preferred seating scheme, such as random assignment, shift the class by rows or columns, and slightly adjust the seats. **My client** approved my proposal, then I discussed with the advisor and we settled down on developing a Java project.

Rationale for Proposed Solution

My product can solve **client A**'s problem because it can operate and display a seating table. Various different options can provide more suitable plans to **the client**. **The client** does not need to spend a long time adjusting the minor mistakes in the output of the previous code; instead, she can set the configuration for once, then run the program later to get the desired seating plan for unlimited times.

After comparing with other fundamental programming language, such as Python and C++, I eventually selected Java due to the following reasons:

- I have experience of coding using Java and I'm familiar with abstraction used in Java.

- Java supports Object-Orientated Programming (OOP), which enables me to use inheritance and encapsulation to make programming efficient. For example, encapsulate attributes for a "layout" to a class and manage the class objects is efficient.

- Java has good portability, it can be transplanted to most platforms so more users can use the product.

- Java supports Graphical User Interface (GUI), and it is easy to implement using NetBeans Integrated Development Environment (IDE). GUI makes my product more approachable to clients.

- Java supports file input and output, which means that the program can produce editable configuration files and printable seating tables for the client.

- The debugger in NetBeans is efficient to locate errors and fix bugs.

Success Criteria

Based on client A's demand, I decided that for my program to be successful, it has to pass those criteria:

1. The client can set up the configuration of a new seating plan, including row and column number and the empty positions.

2. The client can clear and reset the setting of a seating plan.

3. The client can select different seating scheme, such as random assignment, or moving by independent rows/columns.

4. The client can choose the type of layout used in the seating plan.

5. The program can export the plan to a .csv (comma-separated values) file.

6. The client can adjust the seating plan manually by directly editing the produced .csv file.

7. The program can display the seating plan on a GUI.

(578 words)