**Design Decisions For CSV Import/Export Java Utility**

**Singleton**

I decided to split my project up into four classes to handle to tasks of:

* Starting the program while setting up the program log
* Control the behaviour and contents of the GUI
* Reading the CSV File
* Handling MySQL queries

For the CSV and MySQL classes, I used the Singleton class structure which ensures only one instance of each class is ever created. I considered trying to use an abstract class that both these classes could inherit, but I could not find sufficient common methods to implement.

**PK Field**

When dealing with how to import the data into MySQL, I noticed that DropReference was a unique field for each record. However, while I did consider making this field the primary key for the table, I decided to add an extra field, called PK, to act as the primary key instead. I did this because I believe that this would aid any integration with other tables in the future and would allow the DropReference data to be altered easier, should the need arise.

**GUI**

The idea to use a Graphical User Interface was one of my first ideas when thinking of how to implement this program. Not only would it increase the usability of the program, but it would also potentially increase the flexibility of it too. By extending the JFrame class and implementing the ActionListener class into my GUIManager class, I could create a simple GUI that would allow the user to easily select their parameters for the program before running the code to import the selected CSV File. The Browse button demonstrates this, as it prevents the need for the user to type in the path to the CSV File whilst also eliminating human error while typing it. The GUI also provides easy-to-understand feedback, that clearly informs the user of the outcome of the import attempt without the need for any coding knowledge.

**Logger**

To increase the ability to use the stand-alone program without the need for other tools, I created a log file to record when the program started, when the import operation started and all the details of any Exceptions that are thrown whilst the program is running. This complements the user-friendly approach of the GUI, as if the user has coding knowledge and would like specific information as to what went wrong after the program has stopped running, they can simply look at the log file that has been generated within the same directory as the program.

**JUnit4 tests**

The Junit tests were created simply to ensure that the public methods of CSVManager and MySQLManager can run even when passed invalid parameters.

**Import Functionality**

After ensuring that the program could successfully import the jobdata.csv file into an existing database, I proceeded to improve the flexibility of the program by creating the database with the specified table, before importing the file, if they don’t already exist. I then further expanded this functionality by reading the header of the selected csv file to create the SQL for the database table. The checkbox within the GUI determines whether the program will use the pre-coded SQL, intended specifically for the jobdata.csv file, as it provides more accurate types for the columns, or generate SQL from the csv file header. This enables the user of the program to import any CSV file into any database table without sacrificing the detailed SQL for the jobdata.csv file.

**Export Functionality**

After implementing and expanding upon the functionality required by the given task, I deduced that further relevant functionality could be added by including an Export function. This feature creates a CSV file, within the same directory as the program, that contains all the data stored in the specified database table.

**Validation**

The CSVManager class reads the CSV file whilst also ensuring that every row has the same number of columns. All Exceptions are caught, logged and the GUI communicates the failure to the user. These Exceptions are likely to be triggered by invalid parameters input into the GUI by the user and therefore also act as validation for these inputs.