cs3307a – Object oriented analysis and design

Design Inspection Instrument

Instructions:

- The purpose of this document is to assist in the inspection of object-oriented design.
- Under each question is a choice of answers; please choose one (either replace the box with a checkmark or highlight it)
  ☐ yes ☐ no ☐ partly, could be improved
- Two types of comments are required under each question. One is your analysis. The other is your finding (in the form of a comment). The analysis would typically show how you arrived at the finding.
- Add new lines as necessary for your analysis or findings.

Scope of the system to be considered for inspection:

- With reference to Appendix B – Dashboard Screens, take Demo 1 feature, focusing on that part of the code that produces one Dashboard summary.
- Visualisation code is out of scope of this inspection.

++++++++++++++++++++

Inspector 1: Jaidonn

Structural correspondence between Design and Code:

Are all the classes and interrelationships programmed in the application explicitly represented in the class diagram of the system?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The UML diagram was clear in its interactions. It correctly connected all code present at the demo1 stage.
Comment on your findings: The UML explicitly represented the intrarelationships of the application.

Functionality:

Do all the programmed classes perform their intended operations as per the requirements?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: While demo1 was limited in functionality, it was only designed to parse the professor csv. It was capable. Unfortunately, due to its design it cannot csv's that are structured differently.
Comment on your findings: The program performs the indented operations, but fails to create a data structure that is expandable.

Cohesion:

Do the methods encapsulated in each programmed class, together perform a single, well defined, task of the class? (High-Cohesion: the functionalities embedded in a class, accessed through its methods, have much in common, e.g., access common data)

☐ Yes ☐No ☐Partly (Can be increased)

Comment on your analysis: Yes, the classes do a good job of passing information throughout the various classes. Given that it was the main objective of this demo, there was a lot of focus to ensure a high level of cohesion occurred.
Comment on your findings: The classes are highly cohesive.

Coupling:

Do the programmed classes have excessive inter-dependency? (High Coupling: In this case a class shares a common variable with another, or relies on, or controls the execution of, another class.)

☐ Yes ☐No ☐Partly (Can be reduced)

Comment on your analysis: No, common variables are shared amongst classes.
Comment on your findings: No coupling was found

Separation of concerns:

Is the scoped problem decomposed into separate concerns where each concern is encapsulated in a construct such as a class with well-defined interface and cohesive functions with minimal of connections with other concerns?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The structure of the class layout separates concerns well.
Comment on your findings: After review, it is clear that there are minimal connections with other concerns.

Do the classes contain proper access specifications (e.g.: public and private methods)?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: Classes correctly implement the use of both public and private methods to protect data used within the program.
Comment on your findings: There were incorrect uses of either private or public methods.

Reusability:

Are the programmed classes reusable in other applications or situations?

☐ Yes, most of the classes ☐No, none of the classes ☐Partly, some of the classes ☐Don't know

Comment on your analysis: The code only functioned for a single type of csv. The data structure was not adaptive and many variables were hard coded.
Comment on your findings: This data structure will have to be redone in order allow other csv's to be parsed.

Simplicity:
Are the functionalities carried out by the classes easily identifiable and understandable?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The functionality of the classes isn't impossible to understand, it is simply poor. The parsing of data is done in an unclear manner
Comment on your findings: The functionalities of the classes are not intuitive.

Do the complicated portions of the code have /*comments*/ for ease of understanding?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The commenting in particular areas of the code is subpar. There are a few lingering comments that simply do not provide enough understanding for an external user to follow. Some of the comments provided were bad, eg. "kill me".
Comment on your findings: Poor commenting all around

Maintainability:
Does the application provide scope for easy enhancement or updates? (e.g., enhancement in the code is not anticipated to require too many changes in the original code)

☐ Yes ☐No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis: As previously stated, the code was only functional for a single csv. Its structure was hard coded meaning it had no adaptive capabilities moving forward.
Comment on your findings: This code is not maintainable. It must be altered.

Efficiency:
Does the design introduce inefficiency in code (e.g., causes too many nested loops or delays in concurrent processing)?

☐ Yes ☐No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis: The code was quite efficient (for the csv that it was able to parse). It loaded the expanded data set quite quickly.
Comment on your findings: The data structure chosen proved to be efficient.

Depth of inheritance:
Do the inheritance relationships between the ancestor/decedent classes go too deep in the hierarchy? (The deeper a class in the hierarchy, the greater the number of methods it will probably inherit from its ancestors, making it harder to predict its behaviour).

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The inherence level does not extend to a point of concern. It is traceable and clear.
Comment on your findings: Inheritance is properly used.

Children:
Does a parent class have too many children classes? (This could possible suggest an abstraction problem.)

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The code does not have a large number of children clause. This could be impart to methods used to parse the data.
Comment on your findings: The parent classes do not have too many children.

Inspector 2: Jeremy

Structural correspondence between Design and Code:
Are all the classes and interrelationships programmed in the application explicitly represented in the class diagram of the system?

■ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis:
I looked through the class diagrams as well the actual code from the program.

Comment on your findings:
The classes and interrelationships programmed in the application are all explicitly represented in the class diagram of the system

Functionality:
Do all the programmed classes perform their intended operations as per the requirements?

☐ Yes ☐No ■Partly (Can be improved)

Comment on your analysis:

I tested the program as a customer by trying out different options on the UI. I found the data to be incorrect so I looked through the code to find the issue.

Comment on your findings:
The CSVparser had issues when trying to read in the data. The parsing of the data seem to be correct but the proper data was not being read in.

Cohesion:
Do the methods encapsulated in each programmed class, together perform a single, well defined, task of the class? (High-Cohesion: the functionalities embedded in a class, accessed through its methods, have much in common, e.g., access common data)

■ Yes ☐No ☐Partly (Can be increased)

Comment on your analysis:
I looked through the code.

Comment on your findings:
There does not seem to be any issue with cohesion. All the methods perform a single well defined task of the classes.

Coupling:
Do the programmed classes have excessive inter-dependency? (High Coupling: In this case a class shares a common variable with another, or relies on, or controls the execution of, another class.)

☐ Yes ■No ☐Partly (Can be reduced)

Comment on your analysis:
I went through the header files and the classes.

Comment on your findings:
There weren't any coupling issues. All the definitions were in the header files and thus the implementation does not affect the overall program too much.

Separation of concerns:
Is the scoped problem decomposed into separate concerns where each concern is encapsulated in a construct such as a class with well-defined interface and cohesive functions with minimal of connections with other concerns?

■ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis:
I looked through the classes.

Comment on your findings:
Everything seems fine.

Do the classes contain proper access specifications (e.g.: public and private methods)?

■ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis:

Comment on your findings:
The private and public methods are used.

Reusability:
Are the programmed classes reusable in other applications or situations?

☐ Yes, most of the classes ☐No, none of the classes ■Partly, some of the classes ☐Don't know

Comment on your analysis:
The classes, particularly the data parsers are impossible to reuse due to everything being hard coded in.

Comment on your findings:

Simplicity:
Are the functionalities carried out by the classes easily identifiable and understandable?

■ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis:

Comment on your findings:

Do the complicated portions of the code have /*comments*/ for ease of understanding?

☐ Yes ■No ☐Partly (Can be improved)

Comment on your analysis: _____
Comment on your findings: _____

Maintainability:
Does the application provide scope for easy enhancement or updates? (e.g., enhancement in the code is not anticipated to require too many changes in the original code)

☐ Yes ☐No ■Partly (Can be improved) ☐Don't know

Comment on your analysis:
Enhancements and update difficult are hard due to the hard coded components which makes it tedious and difficult for increasing the scaling.

Comment on your findings:

Efficiency:

Does the design introduce inefficiency in code (e.g., causes too many nested loops or delays in concurrent processing)?

☐ Yes ■No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis:

Comment on your findings:
The code is efficient and the all the functions run at less or equal to O(n^2).

Depth of inheritance:
Do the inheritance relationships between the ancestor/decendent classes go too deep in the hierarchy? (The deeper a class in the hierarchy, the greater the number of methods it will probably inherit from its ancestors, making it harder to predict its behaviour).

☐ Yes ■No ☐Partly (Can be improved)

Comment on your analysis: _____
Comment on your findings: _____

Children:
Does a parent class have too many children classes? (This could possible suggest an abstraction problem.)

☐ Yes ■No ☐Partly (Can be improved)

Comment on your analysis: _____
Comment on your findings: _____

Combined Inspection

Structural correspondence between Design and Code:
Are all the classes and interrelationships programmed in the application explicitly represented in the class diagram of the system?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The UML diagram was clear in its interactions. It correctly connected all code present at the demo1 stage.
Comment on your findings: The UML explicitly represented the intrarelationships of the application.

Functionality:
Do all the programmed classes perform their intended operations as per the requirements?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: While demo1 was limited in functionality, it was only designed to parse the professor csv. It was capable. Unfortunately, due to its design it cannot csv's that are structured differently.
Comment on your findings: The program performs the indented operations, but fails to create a data structure that is expandable.

Cohesion:
Do the methods encapsulated in each programmed class, together perform a single, well defined, task of the class? (High-Cohesion: the functionalities embedded in a class, accessed through its methods, have much in common, e.g., access common data)

☐ Yes ☐No ☐Partly (Can be increased)

Comment on your analysis: Yes, the classes do a good job of passing information throughout the various classes. Given that it was the main objective of this demo, there was a lot of focus to ensure a high level of cohesion occurred.
Comment on your findings: The classes are highly cohesive.

Coupling:
Do the programmed classes have excessive inter-dependency? (High Coupling: In this case a class shares a common variable with another, or relies on, or controls the execution of, another class.)

☐ Yes ☐No ☐Partly (Can be reduced)

Comment on your analysis: No, common variables are shared amongst classes.
Comment on your findings: No coupling was found

Separation of concerns:
Is the scoped problem decomposed into separate concerns where each concern is encapsulated in a construct such as a class with well-defined interface and cohesive functions with minimal of connections with other concerns?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The structure of the class layout separates concerns well.
Comment on your findings: After review, it is clear that there are minimal connections with other concerns.

Do the classes contain proper access specifications (e.g.: public and private methods)?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: Classes correctly implement the use of both public and private methods to protect data used within the program.
Comment on your findings: There were incorrect uses of either private or public methods.

Reusability:
Are the programmed classes reusable in other applications or situations?

☐ Yes, most of the classes ☐No, none of the classes ☐Partly, some of the classes ☐Don't know

Comment on your analysis: The code only functioned for a single type of csv. The data structure was not adaptive and many variables were hard coded.
Comment on your findings: This data structure will have to be redone in order allow other csv's to be parsed.

Simplicity:
Are the functionalities carried out by the classes easily identifiable and understandable?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The functionality of the classes isn't impossible to understand, it is simply poor. The parsing of data is done in an unclear manner
Comment on your findings: The functionalities of the classes are not intuitive.

Do the complicated portions of the code have /*comments*/ for ease of understanding?

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The commenting in particular areas of the code is subpar. There are a few lingering comments that simply do not provide enough understanding for an external user to follow. Some of the comments provided were bad, eg. "kill me".
Comment on your findings: Poor commenting all around

Maintainability:
Does the application provide scope for easy enhancement or updates? (e.g., enhancement in the code is not anticipated to require too many changes in the original code)

☐ Yes ☐No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis: As previously stated, the code was only functional for a single csv. Its structure was hard coded meaning it had no adaptive capabilities moving forward. However, the other code is maintainable and provides a good framework moving forwards.

Comment on your findings: Parts of this code is not maintainable. It must be altered.

Efficiency:
Does the design introduce inefficiency in code (e.g., causes too many nested loops or delays in concurrent processing)?

☐ Yes ☐No ☐Partly (Can be improved) ☐Don't know

Comment on your analysis: The code was quite efficient (for the csv that it was able to parse). It loaded the expanded data set quite quickly.
Comment on your findings: The data structure chosen proved to be efficient.

Depth of inheritance:
Do the inheritance relationships between the ancestor/decedent classes go too deep in the hierarchy? (The deeper a class in the hierarchy, the greater the number of methods it will probably inherit from its ancestors, making it harder to predict its behaviour).

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The inherence level does not extend to a point of concern. It is traceable and clear.
Comment on your findings: Inheritance is properly used.

Children:
Does a parent class have too many children classes? (This could possible suggest an abstraction problem.)

☐ Yes ☐No ☐Partly (Can be improved)

Comment on your analysis: The code does not have a large number of children clause. This could be impart to methods used to parse the data.
Comment on your findings: The parent classes do not have too many children.

Behavioural analysis:
From the system's requirements, **create several scenarios** starting from the **user's** point of view: consider identifying one or more **typical** scenarios (e.g., those expected to be used with high frequency) and one or more **low-frequency** scenarios .

Each scenario is described as follows:

i) Title of scenario
ii) Anticipated frequency of use (high, normal, low)
iii) End-user trigger (starting point) for the scenario.
iv) Expected type of outputs.

v) List of bullet points linking end-user inputs and identifying all the key features of the system expected to be "touched" by the scenario and producing the anticipated outputs.

Follow the code (structured walkthrough) to ascertain whether this scenario is properly implemented both in terms of logic and design.

Comment on your findings, with specific references to the design/code elements/file names/etc.:

i) Display publication summary
ii) High
iii) Select a csv and display the information
iv) A summary table of data
v) Key features
    a. GUI, Model class, Controller class, View Class, CSV, Data parsers

---

i) Filter by Count
ii) Low
iii) Select the data to be displayed, then use the filter to filter by count
iv) Data filtered using the count filter
v) Key features
    a. GUI, Model class, Controller class, View Class, CSV, Data parsers

i) Help
ii) Low
iii) Select help button on the task bar
iv) Offers assistance on how to use the software
v) Help window
vi) Key features
    a. GUI, View Class

END.