TU257 Object Oriented Software development – CA2

Submission date: Sunday 21st November 2021 @ 10.00pm

Brief description of the application:

A software company who has diversified into offering IT Support and Office Fitting needs a custom address book to manage all their business contacts. The company has a large number of existing contacts and this will be the basis for the creation of the new address book.

The new address book will run as a Python console application that will be menu driven. The following features must be catered for:

- View a full list of all contacts in the address book
- View a list based on Category: (Development | Support | Office Fitting)
- Offer full CRUD on the contacts in the address book
- Search the address book for all fields (eg. Search for a lastname)
- Create a simple console login to use the application (text based)
- It must be possible to sort the display on the address book by:
    - firstname, lastname, company name.

Structure of csv file:

- id
- first_name
- last_name
- company
- address
- phone(landline)
- phone(mobile)
- category (Development | Support | Office Fitting)
- date created
- date Updated
- modified by (Person's Name)

System Requirements:

- Create a program to automate the generation of the statistics listed above.
- Create a contacts.csv file to match the csv structure listed above. This must be used in the demonstration of your solution.
- The csv module should be used to help with the required csv operations
- Create a Contact class. When the program is run the data from the csv will be loaded into a list for use in the address book.
- Work with the list/csv file:
  - CRUD
  - Search
  - Presentation (sorting)
- Create a menu to let the user interact with the Address Book
- Write all changes to the csv file when the program closes (via menu)
- The program should follow good design principles and comply with the SOLID principles where possible.
- The application needs to be developed using Classes, Methods and follow good programming principles.
  - Comment Code
  - Good naming convention
  - Principles (KISS, DRY, SoC)
- The application must be created using PyCharm.

Documentation (20%):

There is a template document provided on Brightspace. This main focus of the document is to provide some insights into your program design and implementation.

Grading Rubric:

	70+	60-69	50 – 59	40 – 49	0 - 39
Program Structure and Conventions (10%)	Code is clean, understandable, well-organized. All aspects of the solution follow best practices.	Code is clean, understandable, well-organized. All aspects of the solution follow best practices. Some minor issues with the program structures and/or conventions followed.	Code is reasonably clean with a good structure. Some issues with the program structures and/or conventions followed.	OK program and structure. More could have been offered to increase re readability of the code and the structure of the solution.	Minimal attempt in following programming conventions and providing a structured solution. More needed.
CSV File Functionality (25%)	The CSV reading and writing functionality is fully functional and operates all the required tasks as expected.	The CSV reading and writing functionality is mostly functional and operates all the required tasks as expected.	The CSV reading and writing functionality is functional but there are some omissions in its operation.	The CSV reading and writing functionality is somewhat functional but there are some major issues and/or omissions in its operation	Minimal attempt at csv functionality
Classes (15%)	Classes are fully encapsulated and functionality is offered through specific methods, not by mutating their internal variables. Single responsibility has been fully implemented in the solution provided.	Classes are encapsulated and functionality is mainly offered through specific methods, not by mutating their internal variables. Single responsibility has been implemented in the solution provided with some minor issues.	Classes are used and functionality is mainly offered through specific methods, not by mutating their internal variables. Single responsibility has been implemented in the solution provided with some issues in the approach taken.	Classes are used in the implementation and/or the solution offers features via code blocks and functions. The solution could have used classes and SoC more in the implementation.	Minimal attempt in using classes and methods in the solution offered.
Program Functionality – Address Book (30%)	The functionality and reporting required by the application works exactly as expected.	The functionality and reporting required by the application works mostly as expected. Some minor issues identified in the solution.	The functionality and reporting required by the application works. Some issues identified in the solution and/or some features omitted from the solution.	The functionality and reporting required offers a minimal set of operational code. Some major issues identified in the solution and/or some features omitted from the solution.	Major issues with the functionality on offer in the application.
Doc (20%)	The documentation is well written and clearly explains all architectural choices and functionality of the system	The documentation is well written. Could have explained the code and the principle in more detail.	The documentation is acceptable. Could have explained the code and the principle in more detail. Omissions of content or misinterpretation of the principle demonstrated.	The documentation is minimal or not focused on the problem description. Could have explained the code and the principle in more detail. Omissions of content or misinterpretation of the principles demonstrated.	The documentation is simply comments embedded in the code and does explain the code or the principle. Minimal attempt in all aspects.