University of Wolverhampton
Faculty of Science and Engineering
School of Mathematics and Computer Science

Module Assessment

Module	5CS019
Module Leader	Dr John Kanyaru
Semester	2
Year	2021
Assessment	Coursework
% of module mark	80%. The other 20% is from workshop tasks.
Due Date	This coursework constitutes a number of software development activities in which you express your learned knowledge in aspects of software development lifecycle issues, associated notations and tools. You will construct analysis and design models using appropriate tools and notations, as well as implement an object oriented application to realise your designs. You're encouraged to take an iterative approach to development, and seek feedback as you progress. Feedback will be provided mainly via online collaboration tools such as MS Teams and Canvas. To be submitted via Canvas by 14:00 on the due date.
Hand-in – what?	1. Your analysis and design models in either pdf format 2. A brief report as described in the assessment criteria 3. Java source code and unit tests as described in the assessment criteria Everything to be submitted to canvas as 1 zip file named [surname.studentnumber].zip
Cheating, plagiarism, collusion	Any evidence of these offences will not be tolerated. The work you submit must be your own work.
Pass mark	40% is required overall to pass the module
Method of retrieval	Resit will be via one portfolio task only (no workshop tasks) which will not necessarily be the same as the original task.
Feedback	Available during scheduled sessions
Collection of marked work	Via Canvas

Task Description

You are required to design and implement a program to act as an electronic address book for personal and professional use. The program must be able to:

- add a friend entry (name, phone number and address),
- add a work contact entry (name, job title, phone number, company name and company address),
- remove a friend or work contact entry,
- amend a friend or work contact entry,
- retrieve a friend's details after being given a friend's name,
- retrieve a work contact's details after being given a company name and surname,
- remember all the entries between program invocations.

You are required to design and implement a solution, including a graphical user interface (GUI). Analysis and design models should be expressed in UML notations. The program must be written in Java, and the address book entries must be stored in a file (or files), and not in a database.

Design issues for you to consider:

- Will the entries be separated into two (or more) files (personal and professional)?
- Will the new entries be written to file piecemeal or at the end of each session?
- Will the program display a list of possible names or companies as a prompt?
- Which part of a name will be used to retrieve the data?
- Will the program set any limit on the number of entries stored in the address book?
- What object oriented techniques best support the design and implementation?

Assessment Criteria/Marking Scheme

The following documents are to be produced for marking:

a) A report (normally not more than 4000 words):
   i) Documenting your assumptions and rationale for design decisions.
   ii) Discussing your use of appropriate object oriented techniques.
   iii) Discussing issues pertaining to user interface design.
   iv) Analysis and design models using the UML notation. A full use case and class diagram *must* be included.

b)Well commented Java source code

c) Unit tests using JUnit. No need to test GUI classes.

Marks will be weighted as follows:

- Analysis model - use case diagram 10%
- Design model - class diagram 10%
- Code
  - functionality 30%

- code layout and internal comments 10%
- Report 15%
- GUI design and implementation 10%
- Unit tests (at least 80% coverage) 15%

A more detailed marking guide is provided below:

	Core functionality, Unit tests, code layout	**Data Storage & collections**	**Exception Handling**	**Use of OOP concepts & UML Design, analysis**	**Graphical User Interface**
90 – 100%	A robust implementation of contact entry, amendment, intuitive searching, ability to delete, filtering based on say company, Comprehensive unit tests. Code style is clear and internal comments used in suitable places. Criteria for 80-89% is met.	Data is well persisted between program invocations; Separation of application data pertaining to contact types,	Besides checked exceptions, there are other exceptions handled to ensure correct data inputs, and to obviate unreasonable contacts numbers or addresses (postcodes) and email addresses Criteria for 80-89% is met.	The UML analysis & design diagrams are comprehensive & detailed; OOP concepts such as inheritance, encapsulation, polymorphism are used; also constructs such as abstract classes and interfaces are exploited	The GUI is interactive and useful in creating contacts, retrieving them, deleting, etc Evident use of a number of layout managers. Criteria for 80-89% is met.
80-89%	Code is laid out well, and internal comments are succinct; Core functionality meets the requirements of the scenario and there is a clear distinction of the two contact types. Unit tests also do provide good account of a working implementation with a high coverage	Criteria for 70-79% is met.	Comprehensive exception handling. Different types of exceptions considered in different code regions. Some robustness with regards to data type checks could be added. Criteria for 70-79% is met.	Standard OOP concepts used well and analysis and analysis & class diagrams do have appropriate level of detail. Minimal use of advanced concepts such as abstract classes and interfaces.	GUI provides access to various application functions, is interactive and handles user input well. Ability to store, retrieve, amend and delete records. Various events handled (e.g., click event, itemselection event)
70-79%	Application provides core functions, and code is well written in a clear style with use of internal comments as appropriate. Unit tests are also clear with coverage below 70%	Contacts data, persisted well. Suitable collections used to store data in memory. Criteria for 60-69% is met	Criteria for 60-69% is met.	Clear use of OOP concepts and a good match between implementation and design diagram. A detailed analysis model is done.	Criteria for 60-69% is met
60-69%	Criteria for 50-59% is met.	Criteria for 50-59% is met.	Checked exceptions are handled; reasonable work in handling other possible exceptions within constructors or appropriate methods where these might occur	Criteria 50-59% is met	Besides handling application functions, the GUI also provides intuitive messages and tooltips to the user.

50-59%	Criteria for 40-49% is met. Some issues such as duplicate contacts are not handled; some functions such as delete or amend causing exceptions.	Data persists between program invocations, ability to add or amend records; choice of at least 2 appropriate collections and their use.	Criteria for 40-49% is met	UML design diagram captures the problem well, associations among classes are reasonable; Minimal use of OOP concepts but the diagram and code do match; analysis model is reasonably clear.	Criteria for 40-49% is met.
40-49%	Basic features are available, such as adding contacts, and amending them. Unit tests are done, but coverage is low.	Criteria for 26-39% is met. Use of IO classes to store data, and minimal variety of collections used.	Besides checked exceptions, there are some additional ones to handle incorrect data inputs in at least 2 places	Criteria for 26-39% is met	Criteria for 26-39% is met. GUI is able to provide some features e.g., adding, amending contacts. Few layout managers are used.
26-39%	Some classes identified and implemented. No meaningful associations to produce functionality; unable to perform actions such as contact deletion and amendment	Persistence of data is partial, unable to store data about contacts	Criteria for 0-25% is met	Criteria for 0-25% are met. UML design diagram partly matches implementation; inadequate analysis model	Criteria for 0-25% is met. Some event handling is done.
0-25%	Minimal work is done, code has compilation errors	No persistence of data, or storage is partial and does not capture a complete use case for the application	No exception handling, except for checked exceptions	UML class diagram and implementation do not match; also, inadequate for the scenario	No GUI, or it is non-responsive