Total for assignment 30 marks

CRA RUBRIC FOR ASSIGNMENT 2 (INB370)

Criteria for Part 1: the model classes – 20 marks in total

	Excellent (Grade 7)	Good (Grade 6–5)	Average (Grade 4)	Unsatisfactory (Grade 3–1)
Code functionality and process (12 marks)	The classes pass all, or nearly all, of our unit tests. Logs show good evidence of co-operative development and regular commits (12-10 marks)	The classes fail some 'extreme' unit tests, although they pass most 'normal' cases. Good evidence of regular, co-op development. (9-7 marks)	The classes fail a large number of our unit tests, including some 'normal' cases. Limited evidence of regular or co-op development (6-4 marks)	The classes are largely incomplete or fail most of our unit tests. No evidence of co-op development or regular commits (3–0 marks)
Effectiveness of unit tests (5 marks)	The unit tests detect a large fraction of the errors of our test programmes and exhibit good coverage (5 marks)	The unit tests miss several of the more obscure errors in our test programs, but otherwise show good coverage. (4 marks)	The unit tests miss many of the more obscure errors in our test programs and some obvious ones. (3 marks)	The unit tests miss a large number of the errors in our test programs, including many obvious ones. (2-0 marks)
Code Quality (3 marks)	The program code is presented to a professional standard, with self-explanatory identifiers, and clear, concise commenting applied only where needed. Messages in thrown exceptions describe the problem clearly and concisely. (3 marks)	The program code is generally well-presented, but a few identifiers are cryptic, or the level of commenting is inapproprate in places (insufficient or merely repeating the code). Most exception messages are self-explanatory. (2 marks)	The code is reasonably well-presented, but: some identifiers are cryptic; some code is unnecessarily complex; or there is insufficient or unhelpful commenting. Some exception messages are obscure. (1 marks)	The code is poorly presented and hard to understand because: identifiers are not self- explanatory; the code is much more complex than necessary; comments are largely absent or incomprehensible; or messages for thrown exceptions are unclear. (0 marks)

Part 2, the GUI – 10 marks in total; Marked out of 20 and then halved

	Excellent (Grade 7)	Good (Grade 6–5)	Average (Grade 4)	Unsatisfactory (Grade 3–1)
Functionality of program code (10 marks; 7 marks) Note that the red marks indicate levels for those who do not use the charting libraries.	The GUI runs successfully on all of the tests specified by the marker. The submission is in the format specified. Exceptions are handled carefully through an appropriate mechanism, Displays are consistent with the state after each operation, and updates are rapid. (7 marks) (10-9 marks)	The GUI runs successfully on most of the tests specified by the marker. The submission is in the format specified. Exceptions are usually handled carefully through an appropriate mechanism, Displays are consistent with the state after each operation, and updates are fairly rapid. (6-5; 8-7 marks)	The GUI runs successfully on a number of the tests specified by the marker. The submission is in the format specified. Exceptions are sometimes handled reasonably, Displays are usually consistent with the state after each operation, but there may be errors, and updates may betray flaws (4-3; 6-4 marks)	The GUI is not properly functional or markedly incomplete and many operations do not perform as specified. (3-0 marks, both cases).
Quality of the GUI design, layout and user experience. (4 marks)	The GUI design is consistent with the data to be handled and displayed, with an excellent match between data types and display and control widgets. The overall design is clean and uncluttered. Information is well organised and appropriately grouped, and the user is quickly aware of status changes which affect operation. (4 marks)	The GUI design is consistent with the data to be handled and displayed, with a good match between data types and display and control widgets. The overall design is adequate, but is perhaps cluttered or lacking a clean, well laid our feel. Information is usually well organised but some groupings may not be appropriate The user is aware of status changes which affect operation. (3 marks)	The GUI design is broadly consistent with the data to be handled and displayed, but there are some clumsy mismatches between data types and display and control widgets. The overall design is adequate, but is cluttered and not especially clean. Information is grouped badly or not at all, and the layout has not been well thought out. The user is provided with status changes which affect operation, but this may not be readily apparent. (2 marks)	The GUI is poorly laid out, with minimal thought given to its organisation, widget selection and usability. Some aspects may be markedly incomplete (0-1 marks)

Quality of testing (4 marks)	The test scenarios are described precisely and provide excellent coverage of the application functionality. . (4 marks)	The test scenarios are described well and provide broad coverage of the application functionality. . (3 marks)	The test scenarios are described adequately and provide fair coverage of the application functionality. . (2 marks)	The test scenarios are poorly described or ambiguous. The coverage is weak or non-existent (1-0 marks)
Code Quality (2 marks)	The program code is of a professional standard, with self explanatory identifiers, and clear, concise commenting applied only where needed. Authors of each class are identified. The GUI components are well organised around a set of panels hosted by a Frame based class created in the main method. Event listeners are well structured to support the event sources. (2 marks)	The program code is generally presented well, but some of the identifiers are cryptic, or the level of commenting is inapproprate in places (insufficient or merely repeating the code). Authors of each class are identified. The GUI components are reasonably well organised around a Frame based class, but additional panels may have been appropriate, and the event handling is clumsy. (1.5 marks)	The code is reasonably well-presented, but some identifiers are cryptic; some code is unnecessarily complex; or there is insufficient or unhelpful commenting. Authors of each class are identified The GUI components and event handling have some reasonable basic structure (1 marks)	The code is poorly presented and hard to understand because: identifiers are not self explanatory; the code is much more complex than necessary; comments are largely absent or incomprehensible. Authors of the classes are not clearly identified. The GUI components and event handling are poorly organised. (0 marks)