MURDOCH UNIVERSITY

ICT373 Software Architectures
TNE 2025
Assignment 1 (25%)

Due Date: 15/06/2025

All Students: Submit the Assignment via LMS by the due date.

Late submission: After a review, a new university policy for late assessments has been released. Note well that this means that across the university, we no longer apply late penalties to assessments submitted after the due date. Assessments submitted late will receive an automatic mark of 0%, but will still receive feedback.

Extension: However, students can apply for an extension of up to 5 business days (from the original due date) without supporting documentation. If you think your assessment will be late, or it is late, please email the unit coordinator as soon as possible and explicitly state that you need an extension, how long you would like it to be (up to 5 business days), the reason(s) why you need the extension, and how you plan to avoid future late submissions.
More than anything, this is an exercise in communication and accountability and is more akin to how deadlines function in the workplace. As such, when applying for the extension, make sure you are open and transparent in your reasoning.

Policy link: Student Assessment Support Procedure v.3.

Question 1 [6.25 marks]

Use JavaScript and HTML to write a web page which collects the user's name, phone number (with any prefix codes), birth date, and favourite pastime (Surfing the Web, Playing Sport, Listening to Music, Watching TV, Playing Games, Community Service, Daydreaming, Reading, or Meditation only) and sends them off to a server. Make some reasonable assumptions about the content and format of these data and enforce the assumptions from within your page. The address of the test CGI destination is http://www.it.murdoch.edu.au/cgi-bin/reply1.pl.

Note, for the birth date: you need to validate the date, month and year. Do not use a Date Object, check for other data integrity issues e.g., leap year and future dates.

Required Documentation for Question 1

The documentation for Question 1 is to include:

- External documentation consisting of
  - (a) a brief description of the problem
  - (b) a description of your solution approach including structured design/pseudo-code for the Javascript specific code
  - (c) proof of testing and sample results

  The above external documentation should be saved in a file in PDF format
- the web page source code files

Question 2 [18.5 marks]

- For the question given below, you are required to submit design and implementation documentation as described over the page.

Design, implement in Java, test and document a set of classes for use in a program to manage an online weekly personalized magazine service. Each week, the main part of the magazine is made available to each customer along with some extra supplements as chosen by the customer.

- A customer has a name, an email address and a list of supplements which they are interested in.
- A paying customer has a payment method and a list of associate customers whom they also pay for.
- An associate customer is a customer whose subscription is paid for by a specified paying customer.
- A payment method could be by a specified credit card or direct debit from a specified bank account.
- A supplement has a name and a weekly cost.
- The magazine also has a weekly cost for the main part.
- Each week, each customer gets an email telling them their magazine is ready to look at and listing the supplements that they currently subscribe to.
- Each month, each paying customer gets an email telling them how much is being charged to their card or account for the month and itemizing the cost by supplements for them and any of their associate customers.

Design and implement enough functionality in the classes to allow the operation of the following test program (which you also design, implement, test, and document).

The client program should do the following:

a. construct a magazine with an array of 3-4 supplements with made-up details built in to the program (keep provision of inputs from the user using the Java Scanner class),

b. construct an array of 5-6 different customers of various types with made-up details built in to the program (keep provision of inputs from the user using the Java Scanner class),

c. print out the text of all the emails for all customers for four weeks of magazines,

d. print out the text for the end of month emails for the paying customers,

e. add a new customer to the magazine service,

f. remove an existing customer from the magazine service,

Display enough information during the running of the program to allow checking of its operation.

Note that the program should only communicate via command line (i.e., the IDE output window). There is no need for any sophisticated user interface: we are only testing the way these classes work with each other. Also, note that you can use the java API classes (such as ArrayList) instead of arrays to store the above information.

It does NOT have to store information in an external file.

Required Documentation for Question 2

Please remember to submit the Java source code (whole package[1]) and executable version of your program (i.e. the whole NetBeans project). The final version of the program should compile and run under Java SE 8 (JDK 8). Internal students should test compilation and running on the University lab machines. **The source code should be all your own: you can call the java (SE 8) library classes but do not use any source code in your classes generated by the IDE.**

For internal documentation (i.e. in the source code) we require:

- a beginning comment clearly stating title, author, date, file name, purpose and any assumptions or conditions on the form of input and expected output;
- javadoc and other comments giving useful low-level documentation and describing each component;
- well-formatted readable code with meaningful identifier names and blank lines between components (like methods and classes).

Your test code should also include a method (e.g., displayStudentDetails( )) to output your student details (name, student number, mode of enrolment, tutor name, tutorial attendance day and time etc) at the start of program output.

Required External Documentation:

- **Title**: a paragraph clearly stating title, author, date, file names, and one-line statement of purpose.
- **Requirements/Specification**: a paragraph giving a more detailed account of what the program is supposed to do. State any assumptions or conditions on the form of input and expected output.
- **User Guide**: instructions on how to compile, run and use the program.
- **Structure/Design**: Outline the design of your program: describe why you chose one approach rather than other possible approaches. Give a written description. Use diagrams, especially UML, and use pseudocode if you have any non-trivial algorithms.
- **Limitations:** Describe program shortfalls (if any), eg, the features asked for but not implemented, the situations it cannot handle etc.

[1] Please copy the folder on another computer (e.g., in the lab) and make sure it works on an independent machine.

- **Testing**: describe your testing strategy (the more systematic, the better) and any errors noticed. Provide a copy of all your results of testing in a document saved in Word format. Note that a copy of the sample data and results from a test run of the program is required (copy from the program output window and paste to the Word file). Your submitted test results should demonstrate a thorough testing of the program.

Note that all of the above external documentation must be included in a file saved in pdf format.

- **Listings**: Submit the entire Java project so that it can be run and tested.

The external documentation together with all files for both questions must be compressed in a .zip file before submitting. Make sure that all necessary files are submitted so that the programs can be viewed, compiled and run successfully. Note that the whole NetBeans project folder can be zipped.

Remember to complete and sign the Assignment Cover Sheet and submit it with your work.