MURDOCH UNIVERSITY

ICT373 Software Architectures
Semester 1, 2017
Assignment 1 (20%)

Due Date: 5:00pm Saturday 15/04/2017

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

Late penalty: 10% per day penalty for delayed submissions unless prior extension of deadline is obtained from the unit coordinator.

You should keep a copy of your work. Your submission must include a completed assignment cover sheet. An electronic copy of the assignment cover sheet is available at the unit LMS site.

The assignment consists of two questions. For both questions, you are required to submit design and implementation documentation as described below.

Question 1 [5 marks]

Use JavaScript and HTML to write a web page which collects the user's first name, family name/surname, email address, date of birth, and favourite sport (Athletics, Basketball, Cricket, Football, Hockey, Swimming or Tennis 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.

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
    (d) source code listing

  The above external documentation should be saved in a file in MS Word (or PDF) format
- the web page source code file

Question 2 [15 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 a (prototype) online dating service.

The service manages a list of customers.
Customers have login (names) and passwords.
A customer is either an advertiser or a responder.
All customers also have some personal details including gender, age and income.
An advertiser has a text advert, a description of partner sought and a list of reply messages.
A description of partner sought has a gender, range of ages and range of incomes.
A reply message has an owner (which is a responder) and some text.

When new customers are created, i.e., when someone has paid and is accepted, then they choose a new login and they are given their own default password. (We don't encrypt passwords in this prototype.)

When a new advertiser is created (when they log in) they enter the required details and they begin with no replies recorded for them.

When a new responder is created they enter their details. When a responder logs in for the first time or subsequently, the program finds all the matching descriptions sought by advertisers. The responder is then able to choose to send reply messages to any of those matching advertisers.

When an advertiser logs in later they get presented with their list of reply messages, old or new, which they can delete or keep to re-read. We assume that reply messages' text are supposed to contain other ways for advertisers to contact responders (e.g., phone, email) so there is no need for this service to pass messages that way.

There should be a class to manage the list of customers. The final product will use a database to store customer data but the prototype for this assignment just needs to manage a data-structure. It does NOT even have to store information in an external file.

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 (and explain on the screen as it does each step)

a) create a list of 6-7 different customers of both types with made-up details built in to the client program (Alternatively, get input from the user using the java.util.Scanner class),

b) get some matches for a responder, choose one match and send the match a message, then log in that advertiser to get the message,

c) add a new customer to the dating service,

d) delete an existing customer from the dating service,

e) display the details of all advertisers registered with the service,

f) display the details of all responders registered with the service,

g) repeat steps (b) to (d) above for different advertisers and responders to thoroughly test your program.

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.

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.
- **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.

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

- **Listings**: save all your java source code in a document in MS Word (or PDF) format.

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

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.