University of Westminster
School of Electronics and Computer Science

4COSC005/W Programming Principles 2 – Coursework/Test - Arrays – Booking Program

Module leader	Artie Basukoski
Unit	Coding Assignment with in-class test (Coursework/Test item 1)
Weighting:	50% of the module
Qualifying mark	You must get an average of 40% in assessments 1 and 2.
Description	Train seating program using Methods and Arrays.
Learning Outcomes Covered in this Assignment:	LO1, LO3, LO4, LO5.
Handed Out:	10th Feb 2020
Due Date	Code due on Blackboard coursework upload Tuesday 10th March 2020 1pm. In-class test during your Theory Tutorial Session week commencing 9th March 2020.
Expected deliverables	Java program code upload as text files, plus in-class test answers.
Method of Submission:	Blackboard + In-class test
Type of Feedback and Due Date:	Your in-class test mark (worth 80%) and java code mark (worth 20%) should appear on Blackboard Gradecentre within 3 weeks of the test. Individual written feedback will be available via the Blackboard Gradecentre. If you would like extra feedback please speak to the tutor where you did your in-class test. **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

All coursework code on this module is submitted via Blackboard. It may be automatically scanned through a text matching system (designed to check for possible plagiarism).
- You DO NOT need to attach a copy of the CA1 form;

To submit your assignment:

- Log on to Blackboard at http://learning.westminster.ac.uk; and follow the instructions below.

If you are unable to submit your work on Blackboard due to a finance hold you must email your work to ecs-registry@westminster.ac.uk by the same deadline, putting on the subject line the module code, assessment number, and your name. This shows that you have completed your work by the deadline. After the finance hold is lifted you must then submit the same work as normal on Blackboard, otherwise it will not be marked and you will get a fail for that part of the assessment.

Arrays – Train booking program - Coursework and In-Class Test

You are to be assessed on how well you know methods, arrays, and array manipulation code. **Most of the marks will come from your in-class test result.** Please refrain from copying other student's code. It is OK to discuss how to implement the solution, but the code you write must be your own. A good way to test your understanding of the code is to make sure that you can write the code yourself without needing to look at your friend's code, or follow a Youtube video. Concentrate on developing the code step by step and make sure you always have a running version of your program. That way, if you add code and it causes problems you will know that the problem was caused by your new code. Write your test cases as you are developing each method and thoroughly test your code before implementing additional methods.

Train seat booking program.

Task 1. Design and implement a menu driven program for booking seats on a Train with 8 seats. You will need to adhere to good programming style and conventions, for example, avoid magic numbers in your code by declaring a global constant (final) to represent the number of seats for your Train.

```
static final int SEATING_CAPACITY = 8;
```

All other variables must be local. Pass variables as parameters if they are needed in other methods. Use good naming conventions for your variables and methods, and add suitable comments.

The Train's seating will be represented by an array of strings.

```
String [] Train = new String[SEATING_CAPACITY];
```

In this representation, Train[4] = "e", means that seat 4 of the Train is empty, Train[1] = "Mary Drew" means that Mary is occupying seat 1. Seat 0 may not make sense, but ignore it for now. (Hint. You may start by using code which is similar to the hotel code given in your notes)

Once the basic code runs then add code to 'Views All seats' and 'Add customer to seat' into separate methods, and test it works. You can build up your test cases as you develop your program (see 2 below).

Then add a menu system which will allow the user to choose what they want to select. Enter an 'A' to add a customer to a seat, and a 'V' to view all seats. Ensure that all menu options call a separate method to execute the option. When an 'A' is pressed it should call the Add method, a 'V' should call the View method.

One by one, add extra methods to do each of the following menu options. The user should be able to choose from the menu what the program does, until they enter 'Q' which should quit the program.

E:	Display Empty seats
D:	Delete customer from seat
F:	Find the seat for a given customers name
S:	Store program data in to file
L:	Load program data from file
O:	View seats Ordered alphabetically by name. (Using the bubble sort algorithm in the notes)

Task 2. Create a table of test cases showing how you tested your program (see below). Write a brief (no more than one page) discussion of how and why you chose your test cases and comment on your test coverage. Also discuss whether you would "sign off" to having your program used commercially, and if not, what still needs to be done so that you can confidently "sign off".

Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
(Seats Initialised correctly) After program starts, Press 'E'	Displays 'e' for all seats	Displays 'e' for all seats	Pass
(Add customer "Bob" to seat 5) Press 'A', enter "Bob"	Press 'V' Displays "Bob" for seat 5	Displays "Bob" for seat 4	X
….etc….	…..	…..	…..

Submission Instructions:

1. Java Program code to be submitted as a single text file by the date shown above on Blackboard 4COSC005 'Submit Coursework' link. For this submission you should copy and paste your program into a text file, type the filename at the top of each piece of pasted code, and at the very top of the text file type your name and ID number. Call this saved file by your ID number (e.g. . w123456.txt) and ensure it is saved as plain text (not .doc or .zip). Do not change your code after it has been submitted.

2. Submit your test table and report separately as a pdf or word document.

3. Submit your self assessment form as a separate word document.

Tutorial in-class test.

There will be a short in-class test in the theory tutorial session following the submission of the coursework. We need to know that you fully understand the code you use, and have not just stumbled on a solution, or copied it from elsewhere. For the test you will be assessed on how well you know arrays, array manipulation code, and methods, so look at the examples in the notes as well as understanding your Train program.

Do not attempt to go to the wrong tutorial group without the tutor's permission. If you miss the test there will be no other opportunity to redo the work in a later week unless you have approved mitigating circumstances. Sign-in on the class list during the test.

Marking Criteria

80% of the marks will come from the in-class test result. If you score 30% or less in the in-class test then your code solution mark will be limited to a maximum of 50% as the test will have proved you did not really understand the code.

20% of the marks will come from your 'Train' code solution.

Code marking (see self assessment form):
20% - A basic menu which prompts for input until 'Q' is pressed.
+25% A reasonable attempt at (A V E D F) options (5% each).
+20% For options (S L) Save and Load of the Train seating configuration (10% each)
+20% For option 'O' – bubble sort implementation (do not use library sorting routines).
+10% Test cases and test coverage report.
+5% Self assessment and demonstration.

Warning: All code must be done by yourself to ensure you can answer questions during the demonstration. You will be capped at 50% if you cannot explain the code or received less than 30% in the in-class test.
Pass grade: A basic code solution and 40% in the in-class test.
Distinction grade (>=70%).
