COMP1618 (2020/21)	**Software Tools and Techniques**	**Faculty Header ID:**	**Contribution: 100% of course**
Module Leader: Dr Tuan Vuong	**COMP1618 Coursework**		**Deadline Date: Friday 11/12/2020**
This coursework should take an average student who is up-to-date with tutorial work approximately 50 hours Feedback and grades are normally made available within 15 working days of the coursework deadline			
Learning Outcomes: A, B, C and D			

Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

Coursework Submission Requirements

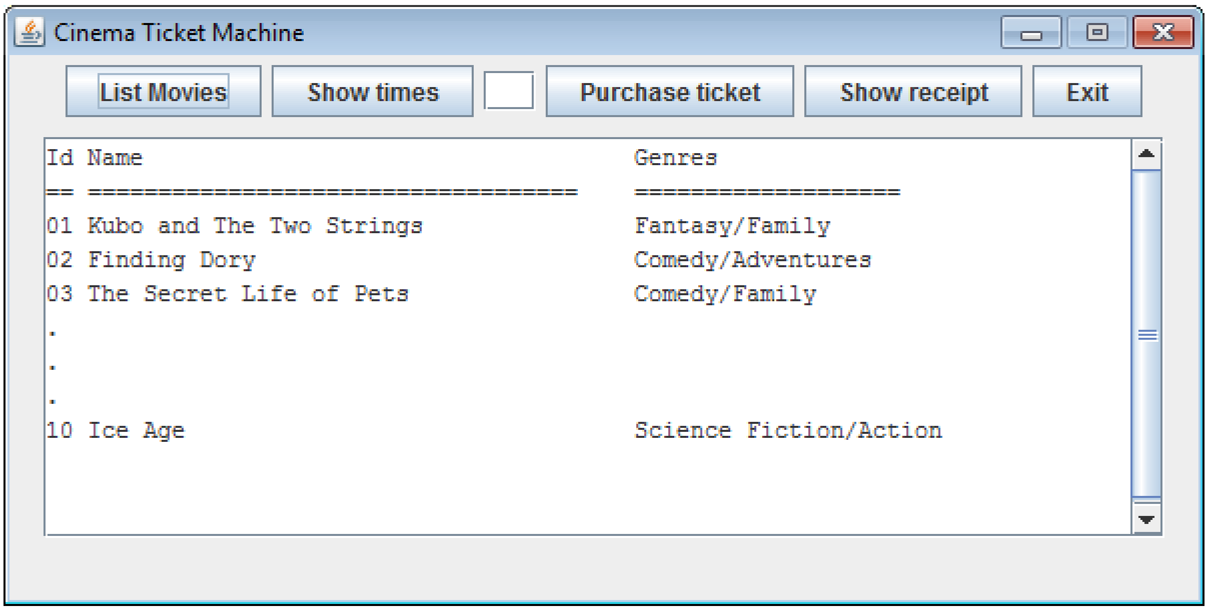
- *An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of **Friday 11/12/2020** using the link on the coursework Moodle page for COMP1618.*
- *For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.*
- *For this coursework you must also upload a single **ZIP** file containing supporting evidence.*
- *There are limits on the file size (see the relevant course Moodle page).*
- *Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.*
- *Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.*
- *You must NOT submit a paper copy of this coursework.*
- *All courseworks must be submitted as above. Under no circumstances can they be accepted by academic staff*

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

Cinema Ticket Machine Simulation

You are to produce a simulation of a cinema ticket machine that allows a customer to buy movie tickets. Each movie might have different show times for the next few days from today. Tickets have different types such as Adult, Senior or Student with the appropriate price. After selection, the customer can see a receipt with a total amount on machine screen and/or get a print out (as in text file). Please note that a customer might want to buy more than one ticket in one transaction.

The machine GUI might look like this when running:



```
Id Name                                     Genres
== ====================================     ====================
01 Kubo and The Two Strings                 Fantasy/Family
02 Finding Dory                             Comedy/Adventures
03 The Secret Life of Pets                  Comedy/Family
.
.
.
10 Ice Age                                  Science Fiction/Action
```

The **List Movies** button just refreshes the list. When the customer enters a valid message ID into the text field and clicks the **Show times** button, a selection of different dates such today, the next consecutive days and available time slots are displayed. The **Purchase ticket** button allows the customer to select different ticket types such as Adult, Senior or Student to complete the transaction. The **Show receipt** shows the current movie tickets purchased within the transaction with a total amount. There should be a cancel and print receipt button in this GUI.

Coursework Stages

There are a number of stages to the development and not all students will manage all the stages. Each stage will be awarded the marks up to the maximum allocated ONLY when handed in AND accompanied by a relevant section in the final report.

In addition, you cannot get marks for a particular stage unless you have made a reasonable attempt at **all** of the previous stages. You may also lose marks if your code contains runtime errors or your report is missing one or more sections.

Stage 1 Basic understanding

Design a suitable GUI for this application with sketches of the layout. You may wish to consider real examples of cinema ticket machines. You should allow for your design to simulate input of the selection of movies, show time, and ticket type including a way of specifying which day, time slot, and ticket type. You will also need to include a ‘cancel’ facility that allows the user to reset the machine, returning any selection made.

Stage 2 Outline implementation

Implement the system which you designed in stage 1 in prototype form WITHOUT functionality (just the GUI appearance). You may use any of the examples supplied in the course material as a basis on which to get started – typically you will need radio buttons and/or a drop down menu (JComboBox), buttons, and text fields. You might also consider using spinners.

Stage 3 A basic working version

Implement the system designed in Stage 2.

Stage 4 Testing and validation

Adapt the code to perform validation, i.e. checking for bad input such as a value of 'four' rather than '4'.

Design *white box testing* of your system (using a test table) and provide evidence of both the functionality of your program and the testing results.

Stage 5 Saving data externally

Extend the program further with an auxiliary class, which uses a text or CSV file to load a movie data and save the print out receipt as text for each transaction including movie name, date/time, type, and amount or more.

Stage 6 Innovations

Extend your program further with two of the following features:

- Allow listing and search for movie by name or filter by genre (e.g. using a JCombobox).
- Allow the program to keep track of the number of available tickets per each movie's show time and save the change to data file.
- Using a database such as JavaDB (Derby): Details are to be stored in a suitably designed database, which you should design and populate with at least 20 records.
- Enhance the GUI by using images, audio or combine GUIs into a single one.
- An innovation of your own devising **which you have discussed with your tutor beforehand**.

Interim Deliverables

There are three interim deliverables.

All interim deliverables should be uploaded to the coursework upload system as a zipped NetBeans project and subsequently demonstrated to your tutor.

Interim DELIVERABLE A (Stages 1 & 2)

For Deliverable A you must upload and present your design sketches for stage 1 and the outline implementation for stage 2 (in a zipped NetBeans project).

Note that the coursework upload system may not allow you to upload a zip file until you have uploaded a pdf document. As a work around, paste your sketches for stage 1 into a Word document, save it as a pdf and upload that first of all.

Interim DELIVERABLE B (Stage 3)

For Deliverable B you must upload and present a basic working version of the code (in a zipped NetBeans project).

Interim DELIVERABLE C (Stage 4)

For Deliverable C you must upload and present a full white box test table (in a pdf document) and your code with error checking/validation included (in a zipped NetBeans project).

Your lecturer will give you a list of dates on which the interim deliverables are to be submitted and demonstrated. Typically these will be 2 or more weeks apart, with interim deliverable C due at least 2 weeks before the final deliverable.

There are no marks for interim deliverables. However, your tutor will give you verbal feedback on what you have done so far and help if there is a problem that you cannot fix, so make the most of the opportunity.

Final DELIVERABLE

To cover as much of stages 1 – 6 as you have completed:

A **demonstration** of your product to your tutor. This is compulsory! Your lecturer will assign you a demo date/time slot.

A **zip file** containing working code in the form of .java and .class files, or a zipped NetBeans project folder, together with any CSV files you have used for external storage.

A **written report** in a separate pdf file (do not include the report in the zip file), containing the evidence of all completed stages, which should include the following **four sections**:

- **Introduction**
- **Design and Development:** a description of how you designed and developed the final code with suitable screen shots of the program in operation
- **Testing and Faults**: a summary of the white box testing (the actual table and results will be listed in appendix B) and a discussion of any faults and failures, including those that you managed to correct, and those which are still unresolved.
- **Conclusions, further development and reflection**: Give a summary of the program and discuss what you would do if you had another three months to work on the program. For the reflection you should write at least 500 words, answer either (a) or (b) from the following:
    a) What did I actually achieve with this element of learning? Which were the most difficult parts, and why were they difficult for me? Which were the most straightforward parts, and why did I find these easy?
    b) What have I got out of doing this element of the course? How have I developed my knowledge and skills? How do I see this element of the course helping me in the longer term?
- Appendices should contain:
    A. The commented version of the code that you that you submitted for interim deliverable A.
    B. White box test **table** and results – this should be updated from the version you submitted for interim deliverable C to cover any changes you have made to the code since then

The written part (excluding appendices) should be no more than 3,000 words and there should be no more than 10 screen shots. There will be a penalty for going over either of these limits.

Grading and Feedback

Indicative Grading Criteria

70-100%: A well-designed and implemented program, together with an excellent accompanying report, which shows a very good understanding of programming concepts and demonstrates some imaginative uses of programming techniques. To gain a mark in this range you must have made:

- a very good attempt at all stages, with an excellent report.

60-69%: A well-designed and implemented program, together with a good accompanying report, which shows a good understanding of programming concepts and demonstrates typical uses of programming techniques. To gain a mark in this range you must have made:

- a good attempt at stages 1-5, with a good report; OR
- a reasonable attempt at all stages, but the report is weak; OR
- a reasonable attempt at all stages, but the program is hard to use; OR
- a reasonable attempt at all stages, but the code contains minor runtime errors

50-59%: A reasonably well-designed and implemented program, together with a reasonable accompanying report, which shows some understanding of programming concepts and demonstrates some uses of programming techniques. To gain a mark in this range you must have made:

- a good attempt at stages 1-4, with a good report; OR
- a reasonable attempt at further stages, but the report is missing a section; OR
- a reasonable attempt at further stages, but the code contains serious runtime errors

30-49%: A poorly implemented program, together with an accompanying report, which shows a basic understanding of programming concepts and demonstrates limited uses of programming techniques. The code may contain some errors or the report may be missing a section. To gain a mark in this range you must have made:

- an attempt at stages 1-3, with a reasonable report; OR
- a reasonable attempt at further stages, but the report is missing several sections; OR
- a reasonable attempt at further stages, but the code cannot be run by your tutor

10-29%: A poorly implemented program, together with an accompanying report, which barely shows any understanding of programming concepts and/or only demonstrates very basic uses of programming techniques. To gain a mark in this range you must have made:

- an attempt at stages 1-2

0-9%: A virtually non-existent program and report. To gain a mark in this range you must have made:

- an attempt at stage 1

Feedback

Your work will be graded using the following feedback sheet.

COMP1618 Coursework Feedback sheet

ID: Overall mark: %

There are **no specific marks for each stage** and your overall mark is based on **your performance as a whole**. The following table indicates which Grading Band your coursework falls into and why. However, you may receive marks in a higher band if your tutor feels some of the work justifies it.

Grading Band	Indicative Grading Criteria	Your work
70-100%	a very good attempt at all stages, with an excellent report	
60-69%	a good attempt at stages 1-5, with a good report	
	a reasonable attempt at all stages, but the report is weak	
	a reasonable attempt at all stages, but the program is hard to use	
	a reasonable attempt at all stages, but the code contains minor runtime errors	
50-59%	a good attempt at stages 1-4, with a good report	
	a reasonable attempt at further stages, but the report is missing a section	
	a reasonable attempt at further stages, but the code contains serious runtime errors	
30-49%	an attempt at stages 1-3, with a reasonable report	
	a reasonable attempt at further stages, but the report is missing several sections	
	a reasonable attempt at further stages, but the code cannot be run by your tutor	
10-29%	an attempt at stages 1-2	
0-9%	an attempt at stage 1	

Demonstration of your product	Excellent	Very good	Good	Weak	None
The functionality of the programme.					
Does the system do what it is supposed to do?					
Usability:					
Is your system straightforward and easy to use?					
Are all messages/labels clear and unambiguous?					
Is the output formatted appropriately?					
Is bad input data handled appropriately?					
Is the system free from crashes and uncaught exceptions?					
Quality of the Java code:					
Inclusion of meaningful comments.					
Use of sensible naming standards.					
Clear code layout and formatting.					
Innovations quality (top 3)					
Innovation 1:					
Innovation 2:					
Innovation 3:					
Demonstration:					
Have you progressively demonstrated your interims (A, B, C)?					
How well is the final application demo with the given time?					
How well are you able to address the questions in the demo?					

Written report	Excellent	Very good	Good	Weak	None
Quality and completeness of the report:					
Introduction and format (Is the documentation clear and concise?)					
Design and Development (evidence on how you designed and developed the application, and results)					
Testing and Faults (Have you included evidence of appropriate testing? Have you discussed any faults or failures?)					
Self-evaluation (Have you done a fair evaluation?)					
Conclusions, further development and reflection (Have you reflected on the development process?)					

Overall Feedback	
Final Deliverable (Feedback on your work as a whole: an indication of the best features of your coursework and a suggestion of what could be improved in the program)	