Assessment Specification

Module	Introduction to Forensic Scripting
Course	BSc Computer and Digital Forensics
Level	4
Module Leader	Asher Rashid
Year	2017-18
Assessment	Component 1 – ICA
Title	Pizza Man
Code & Report Submission Before	• Group Submission • 23.00, Friday 27th April 2018
Marks Available	100 marks
Module Weighting	50%

NOTE:

The ‘Introduction to Forensic Scripting’ module is assessed through two assessment components

- Component 1: Group ICA (worth 50%)
- Component 2: Individual Blackboard Examination (worth 50%)

The module mark will be a weighted combination of marks from both components, for which a weighted minimum of 40% overall is required to pass the module assessment.

Module Learning Outcomes

This component assesses learning outcomes 3, 4 and 5 from the following:

Knowledge & Understanding

1. Recognise the use of key concepts and constructs within a script/programme
2. Explain the differences between various scripting/programming constructs and concepts

Cognitive & Intellectual Skills

3. Analyse the requirements of a simple forensic problem and design an appropriate solution

Practical & Professional Skills

4. Make use of a text-editor to code a solution to a problem
5. Execute solution code and ensure that it satisfies requirements.

1. Scenario

‘Pizza Man’ specialises in taking late night orders from hungry students and delivering direct to the door within 30 minutes of order confirmation. It has a basic online ordering service for registered students, which allows them to order a customised Pizza by Base, Size and Toppings.

Students are required to simulate the Pizza Man service by means of a Java Console Application, making use of different classes and delimited text files:

2. Requirements

2.1 Start up

Upon staring the application should display a basic menu to the user:

- Login
- Register
- Exit

2.2 Login

If the user selects login, then the application should prompt the user to enter both their email address and password. The user input values are checked against values held in the data file **registration.txt** and if correct the user is logged in as a customer and presented with the main menu.

2.3 Register

If user wishes to register, they are prompted for following items of information:

- Forename
- Surname
- Email address
- Password

Upon obtaining the above inputs the customer details will be added to the registration.txt file, along with a unique five-digit ID number. If this is successful, then the new customer will be logged in and forwarded to the main menu and their details stored in the **registration.txt** file.

Note the ID starts at 60001 for first customer and is incremented for each new customer

2.4 Exit

If selected, the application should prompt the user to confirm exit request. If the request is confirmed, then the application terminated.

2.5 Main Menu

After a successful log in or registration, the customer should be presented with the main menu with the options

- Order a Pizza
- Cancel an Order
- View Orders
- Edit Registration
- Exit

2.6 Order a Pizza

If selected, the customer is taken through a four-stage process of ordering a pizza. At any stage of the process the customer should be able to exit and return to the main menu. The first step of the ordering process allows the customer to order the pizza base. The application should use the file **base.txt**, to present a menu.

The second step is for the customer to select the toppings for their pizza, for which the application should use the file **toppings.txt** to present a menu. The customer must select at least one topping, after which they should be prompted if they wish to add another topping or not. A maximum of five toppings are allowed. The customer may choose the same topping two or more times, e.g. Triple Magic Herbs, Double Garlic Butter, etc.

The third step in the process is to select a size, using the file **size.txt** to present a menu along with appropriate total cost, using the calculation.

- *Total Cost = Pizza Size Multiplier * (Pizza Base Cost + Total Toppings Cost*)

The final step in the ordering process is for the customer to be presented with details of their pizza and requested to confirm or cancel the order. If the user decides to cancel the order, a suitable message is displayed, and the main menu redisplayed

If the customer confirms the order then the order must be added to the file **orders.txt**, storing:

- Date Time Stamp in format: *yyyy-MM-dd HH:mm:ss*
  - (e.g. 2018-02-03 23:59:59)
- Customer Id number (e.g. 60001)
- ID number of pizza (e.g. P1517702399)
  - The pizza ID format is the uppercase letter P followed by the number of seconds that the timestamp is since the Unix epoch
- Total Cost of pizza (e.g. 14.10)

An example line entry would be:

- 2018-02-03 23:59:59,60001,P1517702399,14.10

In addition, a **pizza file** must be created, storing details of the pizza ordered:

- Size of the pizza (e.g. 2)
- Base of the pizza (e.g. 2)
- Toppings for the pizza (e.g. 2,2,2,3,3)

The pizza file name would be in format:

- *pizzaID-CustomerID.*

For example:

- P1517702399-60001.txt

And the file would contain:

```
2
2
2,2,2,3,3
```

After data has been successfully saved, a suitable message is displayed to the user and the main menu redisplayed

2.6 Cancel an Order

If selected this option should list pizza orders for the customer which are no older than 10 minutes, (i.e. difference between current date time and saved ordered date time in seconds is <= 600).

Upon selecting a pizza, the order is immediately cancelled, which requires

- The appropriate entry to be removed from the orders file
- The relevant pizza file to be deleted

Finally, an appropriate message is displayed followed by the main menu

2.7 View Orders

Upon selecting this option, a summary report of all pizza orders (within the file **orders.txt**) for the customer is to be displayed. The report should be (tabular) formatted using the column headers

- Date
- Size
- Base
- Price

2.8 Edit Registration

This option allows user to alter their email and their password.

2.9 Exit

Exit should log out the customer and display the login menu.

2.10 General Issues

All file data should be read into the program by means of data class objects

- Thus, each data file should be modelled by an appropriate data class
- Files containing multiple records should be stored into appropriate data structure
- Data in files can be assumed to be validated and correct.

All command line menus should be accompanied by suitable personalised instructions and allow the user to exit by inputting 0 as a command line menu option.

Email addresses and password data are case-sensitive, and no two users can share the same email address. Note there is no need to validate a provided email address is in a valid email format.

Suitable validation supported by exception handling if appropriate, should be implemented to prevent runtime errors caused by inappropriate input.

3. Data Files

3.1 registration.txt

This file holds registration details of each customer, example data in the file is:

```
60001,Mo,Salah,ms@liverpool.fc.uk,skouse
60002,Joe,Root,jr@yorshire.cc.uk,tyke
60003,Chris,Froome,ch@tdf.bike.uk,wheels
60004,Johanna,Conta,jc@slough.tennis.uk,serve
```

The semantic of each line is:

ID Number	(e.g. 60001)
Forename	(e.g. Mo)
Surname	(e.g. Salah)
Email	(e.g. ms@liverpool.fc.uk)
password	(e.g. skouse)

Note this file may be changed whilst a program is running:

- A new customer record is added to the file
- Existing customer may change emails address and / or password

3.2 size.txt

This file holds details of pizza sizes and cost. The data in the file is not to be changed whilst the program is running. The data in the files is:

```
1,standard,1
2,medium,1.5
3,deluxe,2.0
```

The semantic of each line is:

- size id (e.g. 1)
- size (e.g. standard)
- multiplier (e.g. 1)

Note the multiplier is used in calculating the total cost of the pizza using the calculation:

- *Total cost = multiplier * (base cost + toppings cost)*

3.3 base.txt

This file holds details of different pizza bases and their costs. The data in the file is not to be changed whilst the program is running. The data in the files is:

```
1,Yummy,5.00
2,Deep Pan,7.50
3,Square Thin,10.00
4,Cheese Crust,12.50
5,Sharer,15.00
```

The semantic of each line is:

- base id (e.g. 1)
- base (e.g. Yummy)
- price (e.g. 5.00)

3.4 toppings.txt

This file holds details of toppings that can be added to the pizza. The data in the file is not to be changed whilst the program is running. The data in the files is:

```
1,Chorizo Sausage,2.00
2,Magic Herbs,0.30
3,Garlic Butter,0.50
4,Green Peppers,0.50
5,Ground Meat,2.00
6,Jalapeno Peppers,1.00
7,Meatballs,3.00
8,Mushrooms,0.50
9,Onions,0.50
10,Pastrami,2.00
11,Pepperoni,2.00
12,Pineapple,0.50
13,Sweetcorn,0.50
14,Tandoori Chicken,2.00
15,Tomato,0.50
16,Tuna,2.50
17,Kebab,1.50
18,Veggie Chunks,1.00
19,Chips,1.50
20,Mango Cubes,0.50
```

The semantic of each line is:

- topping id (e.g. `1`)
- topping (e.g. `Chorizo Sausage`)
- price (e.g. `2.00`)

3.5 orders.txt

This file holds an overview of orders taken; all new orders are added to this file.

- Line Format: *Date Time Stamp*, *Customer ID*, *Pizza ID*, *Cost*
- Example entry: `2018-02-03 23:59:59,60001,P1517702399,14.10`

Note:

- The pizza ID format is the uppercase letter P followed by the number of seconds that the timestamp is since the Unix epoch
- If a customer successfully cancels an order, then the appropriate line will be removed from the file.

3.6 pizza files

Upon ordering a pizza the details of the pizza are saved into an unique text file (e.g. `P1517702399-60001.txt`). The descriptive part of the filename of the text file consists of *pizzaID*, a *dash* and then *customerID*.

Each pizza file contains three lines

- First line specifies size ID of the pizza
- Second line specifies base ID of the pizza
- Third line specifies the toppings IDs

For example:

```
2
2
2,2,2,3,3
```

The above specifies a pizza which is of a `medium` size, has a `deep pan` base, a `triple` topping of `magic herbs` and `double` topping of `garlic butter`.

If a customer successfully cancels an order, then the relevant pizza file will need to be deleted.

4. Marking criteria

Marking of the assessment is split between coding/testing, report and evidence of team working. To which an individual rating will be applied based on the combination of peer review (see Appendix A) and staff observation

Coding and Testing marks (65 Marks)

Code Layout	5 marks
Documentation within class files	5 marks
Login	5 marks
Register	5 marks
Quit	3 marks
Order Menu Processing	10 marks
Display Orders	5 marks
Cancel Order	5 marks
Change Registration	5 marks
Exit	2 marks
Files (orders and pizzas)	5 marks
Housekeeping	10 marks

Report Marks (25 Marks)

Introduction	2 marks
Analysis / Decisions	4 marks
Design	4 marks
Coding	4 marks
Testing	4 marks
Conclusions	2 marks
Referencing	2 marks
Presentation, Examples, Figures and Tables	3 marks

Evidence of teamworking (10 Marks)

PDF scans of signed minutes of weekly meetings	4 marks
Team action plan	3 marks
Gantt Chart of work done for each group member	3 marks

Note:

- Marks are awarded for the quality of any of the above criteria provided within a group's code, report and teamwork evidence

- With regards to referencing, specific marks are including within documentation for coding and in the report

- If no references are used within either the code or report
    - No marks will be awarded for referencing
    - Students should certify (and sign) the specific aspects of the work were entirely then own, as an appendix within the report

- If the references or certification do not support content within the solution and or report, then the submission will be subject to **marking penalties**; which may result in **zero marks** be awarded for the relevant marking criteria.

- Housekeeping marks for code, relate to
    - Structuring of scripts within solution
    - File names (including names of submitted work see page 14)
    - Appropriate use of folders to maintain relative file paths
    - Removal of redundant temporary files and folders after execution of the solution
    - Ensuring all variables, constants, arrays, methods and classes declared, are meaningfully used beyond initialisation/instantiation.

- If a script file cannot be opened, then zero marks will be awarded for relevant marking criteria

- If the solution fails to execute then zero marks will be awarded for relevant marking components

Group Submission

Before	23.00, Friday, 27th April 2018 (Week 11)
Where	Module VLE/Blackboard Site (eat.tees.ac.uk)
What	• Report in DOC, DOCX or PDF format • Code and Date files compressed within a zip file • Evidence of team working, compressed within a zip file

Individual Submission

Before	23.00, Friday, 27th April 2018 (Week 11)
Where	Module VLE/Blackboard Site (eat.tees.ac.uk)
What	• Completed Peer Rating pro forma in DOC, DOCX or PDF format

Note:

- Only one submission of group code/report/evidence per group is required
- All students to submit peer rating
- Peer rating file name
    - Format: IFS-Surname-Group*Number*-Rating
    - Example: IFS-Rashid-Group11-Rating.docx
- Team evidence zip file name:
    - Format: IFS-Group*Number*-Evidence.zip
    - Example: IFS-Group01-Evidence.zip
- Report file name:
    - Format: IFS-Group*Number*-Report
    - Example: IFS-Group01-Report.docx
- Code zip file name:
    - Format: IFS-Group*Number*-Code.zip
    - Example: IFS-Group01-Code.zip

- Class names and file names
    - Students are expected to apply conventions relevant to each class and file name, unless specified otherwise with sections 2 and 3 above
- If a group wishes to submit contemporaneous notes for any practical research, it should be submitted as a PDF fie for which the filename is in the following format:
    - Format: `IFS-Group`*Number*`-Notes.pdf`
    - Example: `IFS-Group01-Notes.pdf`

Extensions

- Students may apply for short or long extensions, in case genuine circumstances have lost the student the opportunity to work on Component 1 during term time.
- Short extensions (for no more than seven days) must be requested through Blackboard from the Module leader using a short extension form before the submission date for the ICA
- Long extensions (for more than seven days) are submitted through SSED Reception for the attention of the Assistant Head of Science Department (Learning & Teaching); after a discussion with the Programme Leader. The Long extension form must be supported by written evidence and submitted before the relevant submission date(s) for Component 1. Students should inform the module leader that they are requesting a long extension.
- If requesting an extension, students are advised to submit any work they have done on the assessment before the relevant submission deadline.
- If a short or long extension is granted, then the affected student will be allowed to upload / present their revised assessment work before the extended deadline. When doing so it is advisable to also upload a PDF copy of the extension approval
- Students can make a late submission, no later than seven days after the specified submission date. Any late submission will be subject to a lateness penalty.

Note:

- As this Component is based on group work, all students in a group will be required to get an extension, instead of an individual student.
- If circumstances mean a student is unable to participate with their group, e.g. illness, the students should seek guidance from the Module Leader.

Appendix A. Peer Review

All students, on an individual basis, will rate all members of their team (including themselves) using the below form. Note if in rating yourself and or other members of the team a rating of 0, 1, 4 or 5, justification should be provided for the rating. If any justification does not align with evidence of team working, then that rating will be ignored.

A separate pro forma is available on Blackboard, however below is an example of what needed to be completed in rating one member of the group

Team member		
Area of work	**Rating [0 to 5]**	**Justification**
Analysis		
Design		
Coding		
Testing		
Report		
Teamwork		
Total		