Audit Trail

Programme	Digital & Technology Solutions Degree Apprenticeship	
Module Name	Object Oriented Programming	
Level	4	
Module Leader	Pete Behague	
Assessment Author	Pete Behague	
Reviewer	Name	Date
	Steph Ferguson	09/11/20

Learning Objectives of module:

Learning Outcome Specified in Module descriptor	Which Assessment question maps to this learning outcome. e.g. Question 1,4	Learning Objective covered within Exam.	Topic(s) when material is covered. e.g. Topic 2-3
L01	Task 3, Task 4	N/A	All Weeks
L02	Task 3, Task 4	N/A	All Weeks
L03	Task 3, Task 4	N/A	All Weeks
L04	Task 2	N/A	All Weeks
L05	Task 1	N/A	All Weeks

Review Checks

		Reviewer Comments:
1	Does the assessment structure match module descriptor?	yes
2	Does the question relate to intended learning outcomes?	yes
3	Is the language simple, clear, unambiguous and straightforward?	yes
4	What are the key verbs describing the task? Are they clear? (Analyse, explain, evaluate etc.)	Create, explain, justify
5	Is the language used easy to understand, including by candidates for who English is not their first language? (e.g. does it use colloquial phrases)	yes
6	Is the punctuation and grammar correct, as this can markedly change the meaning of sentences?	yes
7	Can the answers be marked consistently and reliably?	yes
8	Is the marking scheme clear and user friendly? You may have an external marker.	yes
9	Is the division of marks between questions appropriate and fair?	yes
10	Are there any questions with too many marks, where students will be penalised if they give up in the first section?	yes
11	**Exams Only:** Can the questions be completed in the time available (including reading, thinking and reviewing time), including those for who English is not their first language?	N/A
12	Does the question lead to answers which will distinguish between weak and strong candidates e.g. are there elements for candidates to demonstrate distinction-level skills/knowledge?	N/A
13	Does the assessment structure match module descriptor?	N/A

DTS Management Review

EAP Review

1	Do the Assessment Instructions match the standardised guidance?	
2	Does the Word Count match the Module Descriptor?	
3	All Appropriate Checks Have been completed	

Digital & Technology Solutions Degree Apprenticeship

Object Oriented Programming
Level 4
20 credits

Written by:	Pete Behague
Checked by:	Checker
Programme Leader Approval:	PL/DPL
UoR Approval:	
Approved for Review (Multiple Use):	12 months

Assessment Brief

This assessment brief provides details of the overall assessment for your module. It will provide details of the coursework elements. Section 1 provides the detail of the assessment and Section 2 provides general assessment brief guidance.

Component: Coursework Assignment (100%)
Description: For the report, the overall word count is 1500 words.

A mark of at least 40% must be achieved to pass the module.

Submission details

Component	**Date**	**Time**
Coursework	Friday Week 10	14:00

Module Learning Outcome Assessment Matrix

Learning Outcome	Assignment
Demonstrates a broad understanding of Object-Oriented technologies, terminologies and discourse, including awareness of ongoing debate and advances in OO paradigm, and the contributions of new tools such as the use of UML	X
Identifies principles and concepts underlying the theoretical frameworks of OOD/P, and debates limitations, including an analysis of strengths and weaknesses	X
Collects information from a variety of authoritative sources to inform a choice of solutions to standard problems; advances the knowledge of OOP; and is familiar with a variety of research methods such as qualitative and quantitative.	X
Demonstrates comprehensive analytical knowledge of design and implementation of object-oriented programmes, taking quality and reusability into account using API's or object repositories.	X
Employs a structural approach to test an OOP, using a test plan and test log, monitoring expected and actual results.	X

Summative Assessment Brief

Deans Beans

You are advised to read all instructions carefully before starting work and to check with your tutor(s), if necessary, to ensure that you have fully understood what is required.

Assignment set up: A scenario is provided for candidates in the form of a company specification for a service they require.

Scenario

You are required to make changes to an ordering system used by “Call Centre” staff of the Deans Beans coffee company who sell beans and ground coffee to discerning customers. The company needs an application that will allow staff to take orders for their beans and ground coffee related products from wholesale customers who dial in to place orders. The application can only be used by wholesale customers who have already registered with the company.

The application will allow the call centre staff to select a customer from a list, progress to a second form which will allow them to select products from the company’s range and then add them to a basket. The product details are held in a database (called DeansBeans). Once the call centre employee has selected all the products and quantities the customer has asked for, the employee should click a confirmation button which will save the order and its order items to the database and display a confirmation message.

The Deans Beans Database

You can find a database diagram and table breakdown in appendix B

C# Only

The following script can be used to create an SQL Server, DeansBeans database complete with some test data which can be found in the following file:

SQLServerDeansBeans.sql

Java Only

The following script can be used to create a MySQL Server, DeansBeans database complete with some test data which can be found in the following file:

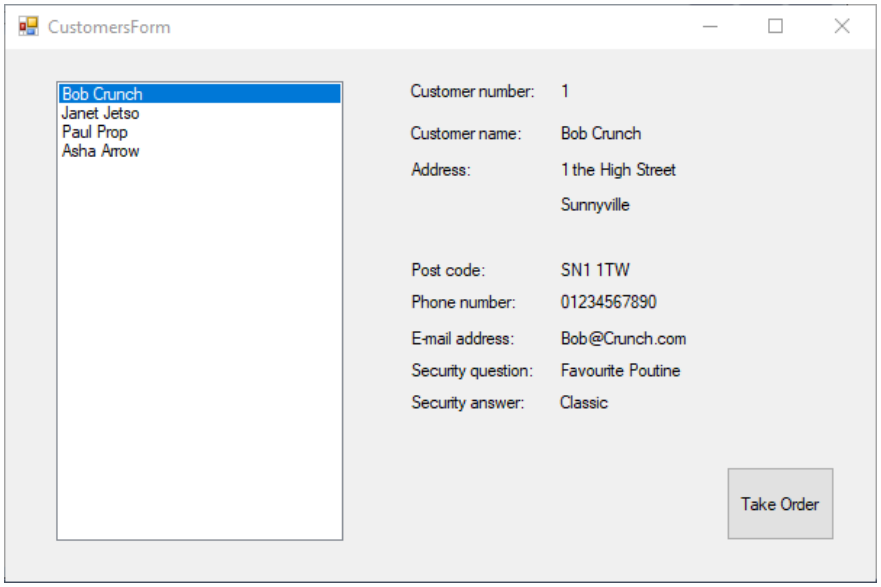
MySQLDeansBeans.sql

Functionality

Your program should provide the following functionality:

Customers Form:

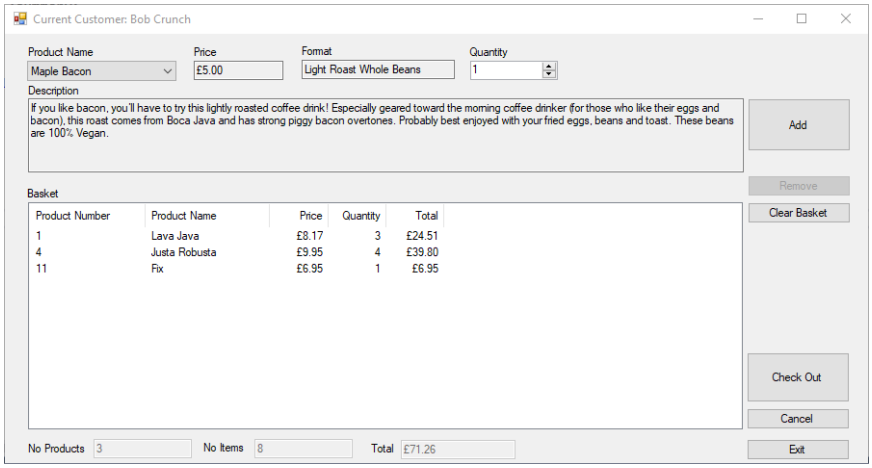
When the program first starts it displays a form that presents a list of customer names (retrieved from the Customers table in the database) from which the call centre employee can select the name of the customer who is on the phone. On selecting a customer, the operative will be shown some customer details from which they can ask a number of security questions and if all is OK, they can progress to the Order Basket form.



Order Basket Form

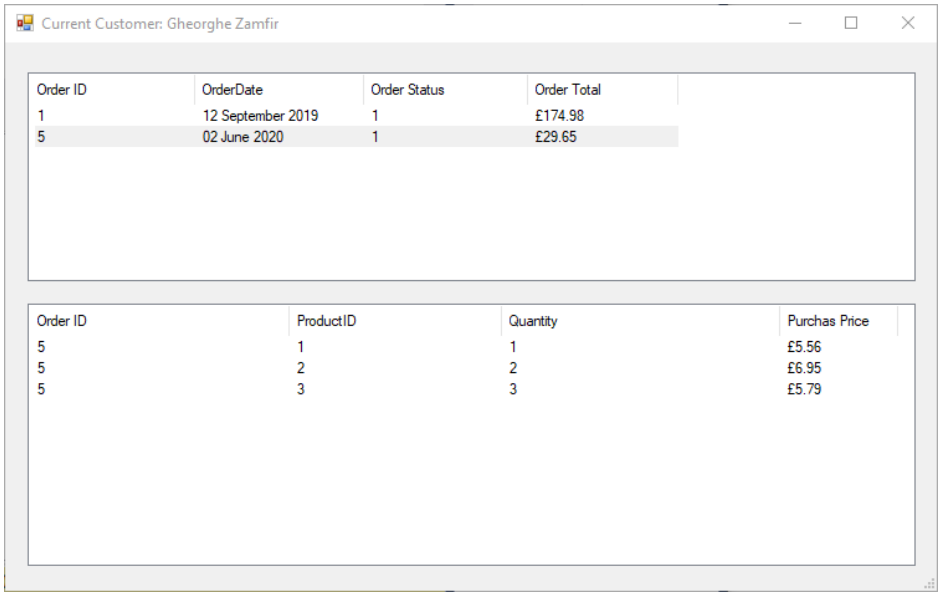
When first displayed, the form allows the user to:

- Select different products from the combo box and display their wholesale price and description in the associated controls.
- Specify the quantity of a selected product they wish to put into the basket.
- Specifiy the Format they would like the coffee to come in (e.g. Whole Beans, Coarse Ground, Medium Ground, Fine Ground, Dust or various forms of pod). These values come from the Format's table in the database. All coffee products are available in all of the specified formats.
- Specifiy the degree of roasting they would like the coffee to come in (e.g. Raw, Light, Medium, Dark, Burnt, Cremated). The higher the numeric value, the darker the roast and range from 1 to 6. These values (along with associated descriptions) come from the DegreesOfRoast table in the database. All coffee products are available in all of the specified values.
- Add an item to their basket and display the basket's content(s) in an appropriate control.
- Remove an item completely from the basket.
- Empty the whole basket.
- Cancel the order and return the user back to the customer form (whilst hiding or unloading the Order Basket form).
- Terminate the program.
- Confirm the order, adding the details to the Orders and OrderItems tables in the database and displaying the CustomerOrderHistory form. **NOTE: The logic that saves the order and associated order items to the database has not yet been written (see task 3)**



CustomerOrderHistory Form

When displayed, the CustomerOrderHistory form displays two “list type” controls. One that displays all the orders made by the current customer (including those that have been recently added) and, on clicking a row in this control, the second displays all the orderItems associated with the selected order



Assessment Requirements

For Tasks 1, 2 and 4 select from the sets of alternatives provided.

Task 1: - Testing

The program as given (see appendix C for both C# and Java versions) contains the source code for both a BasketItem and an OrderBasket class. You will also find an incomplete set of unit tests for these two classes. The OrderBasket and BasketItem classes live in their own independent project. There is an intention to reuse the classes in other development scenarios, so checks will need to be

made against some of the values (such as product name, quantity, etc.) being passed into the BasketItem class to ensure they are acceptable. With this in mind look at the code that:

Select one from each of the three sets of alternative requirements:
Section A

- Covers the setting and retrieval of the quantity. Change the logic to ensure an attempt to set the quantity to a negative, zero or too high a value is rejected by the BasketItem code. Use your experience and judgement to come up with a sensible upper limit. Create appropriate unit tests to ensure the code is robust.
- Covers the setting and retrieval of the wholesale price. Change the logic to ensure an attempt to set the wholesalePrice to a negative, zero or too high a value is rejected by the BasketItem code. Use your experience and judgement to come up with a sensible upper limit. Create appropriate unit tests to ensure the code is robust.

Section B

- Covers the BasketItem code that increases and decreases the quantity. Explain why the logic is problematic and create unit tests to demonstrate this. Finally, fix the issue and prove the unit tests now work.
- Covers the setting of the desired Degree Of Roast (DegreeOfRoastID). Explain why the logic is problematic and create unit tests to demonstrate this. Finally, fix the issue and prove the unit tests now work.

Section C

- Deals with addition of new BasketItems to the OrderBasket. Currently the code simply adds each passed in BasketItem to its collection. There is a requirement that if the basket already contains an item for the same product, the code should adjust that item's quantity rather than adding it as a new row. **NOTE: Items with the same productID but differing Formats are to be treated as different products**. Change the logic to achieve this requirement and create appropriate unit tests to ensure the code works correctly.
- Deals with addition of new BasketItems to the OrderBasket. Currently the code simply adds each passed in BasketItem to its collection. There is a requirement that if the basket already contains an item for the same product, the code should adjust that item's quantity rather than adding it as a new row. **NOTE: Items with the same productID but differing DegreesOfRoast are to be treated as different products**. Change the logic to achieve this requirement and create appropriate unit tests to ensure the code works correctly.
- Deals with addition of new BasketItems to the OrderBasket. Currently the code simply adds each passed in BasketItem to its collection. There is a requirement that if the basket already contains an item for the same product, the code should adjust that item's quantity rather than adding it as a new row. **NOTE: Items with the same productID but differing Formats AND DegreesOfRoast are to be treated as different products**. Change the logic to achieve this requirement and create appropriate unit tests to ensure the code works correctly.

For all three of the above changes, fully explain and justify the changes you make by adding them as comments to the code. Include appropriate screenshots that show both the code and evidence of the tests passing and/or failing. (Recommendation for allocation of word count 250 words).
(30 Marks)

Task 2: - Coding Requirements

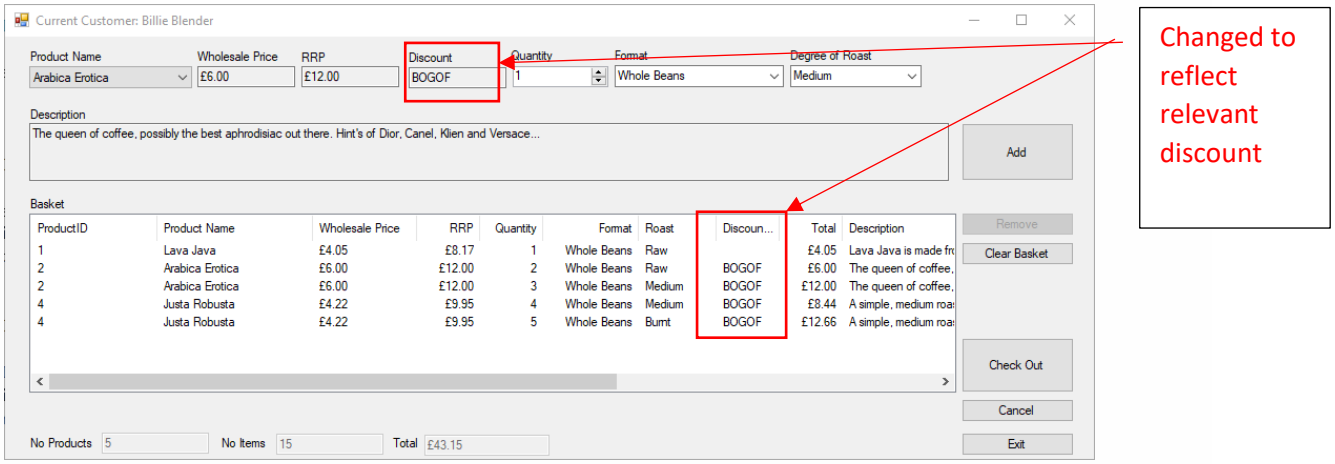
Select one from the differently coloured options of the below alternatives for Task 2
Dean's Beans are looking to introduce a series of discounts to some of their products including Buy One Get One Free (BOGOF), Percentage Discount, Three (of the same product) For The Price Of Two (TFTPOT) and Four (of the same product) For The Price Of Three (FFTPOT) offers. You will note the Products table in the database has already been designed to cater for this (as well as for a

number of other discount types which **you do <u>not</u> need to worry about**). The table contains a DiscountType column which contains the following values:

- “B” - part of the BOGOF (Buy One Get One Free) promotion.
- “P” - part of the percentage discount promotion.
- “T” - part of the TFTPOT (Three For The Price Of Two) promotion.
- “F” - part of the FFTPOT (Four For The Price Of Three) promotion.
- If the column is empty then no discount should be applied

Provide examples of code with explanatory comments that will cater for the new requirement. You are encouraged to implement this by creating a specialised version of the BasketItem class called BOGOFBasketItem, PercentageDiscountBasketItem, TFTPOTBasketItem, FFTPOTBasketItem and a set of additional unit tests to ensure the new functionality works with evidence in an appendix.

You must also update the UI logic such that the OrderBasketForm looks something like the following:



Note the addition of a control that shows whether a selected product is part of the BOGOF, percentage discount, TFTPOT, FFTPOT promotion and an additional column in the control that displays the contents of the order basket.

Show relevant snippets of your code along with any additional explanations you feel are relevant in your answer. (Recommendation for allocation of word count 250 words)
(**25 Marks)**

Task 3: - Accessing the database

The code behind the Order Basket Form’s Checkout button makes a call to the repository object’s SaveOrderToDatabase method. This method currently only contains a single line of code that returns -1.

Briefly explain the mechanism you could use to persist the order to the database in an OO manner using an ORM (e.g. EntityFramework (C#); Hibernate (Java)) and explain the benefits and disadvantages of using this approach over the more traditional use of ADO.NET (C#) / JDBC (Java). Using the already referenced database access technology add code to the SaveOrderToDatabase method that creates an order and associated order items and saves them to the database. Feel free to amend interfaces and add any additional functions to the logic as you see fit.

Fully justify the approach you take and provide appropriate code snippets along with evidence of the code working including explanations. (Recommendation for allocation of word count 600 words)

(**20 Marks)**

Task 4:UI recommendations

One from the differently coloured options below will be selected for Task 4

Provide an outline of 3 key areas that could be developed when looking to improve the user and customer experience based around the Customers, Customer Order History form. Fully justify your recommendations (Recommendation for allocation of word count 400 words)

(**15 Marks)**

Academic Conventions

Throughout your assignment (both your program code and supporting documentation), marks will be awarded for correct academic conventions being shown in the context of, spelling, punctuation and grammar, academic referencing, and academic presentation. Your program code should adhere to basic programming guidelines such as naming conventions, indentation, comments and refactoring to avoid duplication of logic. All sources of knowledge used **MUST** be referenced using the Roehampton version of the Harvard System

(10 **Marks**)

Evidence to be uploaded (via 2 different "assignment" uploads):

- **Zip of your entire Visual Studio or Eclipse Solution.**
- **A word document that contains your answers to tasks 1 to 4 with appropriate appendices.**

Marking Rubric – Student's copy – to be adjusted accordingly to the relevant term and variants selected for Task 1 and Task 2 above, do not publish the full table below to the students.

Task	80-100%	70-79%	60-69%	50-59%	40-49%	0-39%
TASK 1 The BasketItem's quantity wholesalePrice property must only accept values that lie between appropriate limits. Code that • sets the desired coffee format • increases and decreases BasketItem's quantity • sets the desired degree of roast Code that ensures non repetition of products in the order basket. Explanation of problems; associated unit tests; explanation of fixes. **Max 30 Marks**	Outstanding: Throughout all three tasks the code goes the extra mile in ensuring the requested validation and functionality will be correct in **all** situations and products only turn up once in the basket. Explanations are full, clear and precise. Solution is extremely well tested and works intuitively from the user perspective. **24-30 Marks**	Excellent Throughout all three tasks the code ensures the requested validation and functionality will be correct in **all** situations and products only turn up once in the basket. Explanations are clear and precise. Solution is robust and very well tested. **21 - 23 Marks**	Very Good: For each of the three tasks the code goes a long way to ensuring the requested validation and functionality will be correct in **all** situations and products are not repeated in the basket. Explanations are clear. Solution is well tested. **18 - 20 Marks**	Good: For each of the three tasks the code largely ensures the requested validation and functionality will be correct and products cannot be repeated in the basket. Good explanations are given. Core tests are in place. **15 – 17 Marks**	Basic: An attempt has been made to ensure the requested validation and functionality will be correct and products only appear once in the basket, but there are obvious gaps. Explanations are perfunctory. Basic testing is being done but the gaps are not being picked up. **12 - 14 Marks**	Unsatisfactory: Little or no attempt has been made to enhance the existing code. There are no explanations where the code has been changed. Unit tests are missing or buggy **0-11 Marks**
Task 2 • Handling discounts. BOGOF • Percentage Discount • TFTPOT • FFTPOT items are treated independently of the other products	Outstanding: Demonstrates complete understanding of the issues; the solution covers all eventualities and is thoroughly tested. Discussions and explanations are clear, precise and to the point. **20-25 Marks**	Excellent: The new functionality is excellently implemented and covers all eventualities. The additional testing is thorough. Explanations are clear and to the point. **18-19 Marks**	Very Good: Demonstrates a very good understanding of the issues; the solution covers most eventualities, and they are well tested. Discussions and explanations are relevant and demonstrate very good understanding. **15-17 Marks**	Good**:** The new functionality is well implemented and covers most eventualities. Likewise, for the testing. However, the solution does not use a specialised version of the BasketItem class. The discussion is clear but misses some significant points. **13-14 Marks**	Basic An attempt has been made to address the problem and is partially successful. However, no attempt has been made to use a specialised version of the BasketItem class. There is evidence of some basic testing being done. Discussions are perfunctory.	Unsatisfactory : Little or no attempt has been made to get to grips with the issues. Discussions are lacking. Testing has either not been documented or done **0-9 Marks**

with appropriate calculations being carried out. The UI reflects the new requirements. The code has been fully unit tested. **Max 25 Marks**					**10-13 Marks**	
Task 3 Discussion on ORM's vs traditional approaches. Database logic to save order to database. Details and full justification of approach taken. Evidence of testing. **Max 20 Marks**	Outstanding: Implemented solution fully delivers and complements existing logic. Clear, precise and to the point discussion on the merits of the possible approaches. Convincing justification of approach taken. Clear and complete evidence of testing having been done. **20-25 Marks**	Excellent: The database update logic is well designed fitting in with the existing functionality with all the necessary functionality in place. There is a thorough discussion on the merits of the possible approaches. The approach taken is well justified. There is full evidence of testing having been done. **18-19 Marks**	Very Good: The required database update logic is well designed meeting most of the necessary requirements. The discussion on the merits of the possible approaches is well presented and the justification for the approach taken is convincing. Evidence of testing having been done is convincing. **15-17 Marks**	Good: Implemented solution only partially delivers and may not complement existing logic. Discussion on the merits of the possible approaches is attempted but it is not complete. Justification of approach taken is not completely convincing. There is evidence of testing having been done. **13-14 Marks**	Basic: Implemented solution partially delivers on requirements but does not take account of existing logic. The discussion on the merits of the possible approaches makes some valid points but lacks precision. An attempt at justifying the approach taken has been made. There is evidence of some testing having been done. **10-13 Marks**	Unsatisfactory: Little or no attempt has been made at meeting the requirements. Some elements may have been partially attempted. **0-9 Marks**
Task 4 Discussion and outline of 3 key areas of development to enhance the user and customer experience based around the • Customers • Customer Order History **form. Max 15 Marks**	Outstanding Outstanding discussion with strong, well-argued justifications for all three suggestions. Demonstrates an outstanding understanding of the issues involved. **12 - 15 Marks**	Excellent Excellent discussion with well-argued justifications for all three suggestions. Demonstrates an excellent understanding of the issues involved. **11 Marks**	Very Good Very good discussion in part supported by some justifications for all three suggestions. Demonstrates a broad understanding of the issues involved. **9 - 10 Marks**	Good Good discussion with some good arguments though not all the suggestions were convincing. Demonstrates some understanding of the issues involved **8 Marks**	Basic Basic discussion though a bit superficial. None of the suggestions were backed by convincing arguments. Demonstrates a basic understanding of the issues involved. **6 - 7 Marks**	Unsatisfactory Little or no attempt made at discussion beyond the three suggestions. Shows major gaps in understanding the issues involved at this level. **0 - 5 Marks**
Academic Conventions Max 10 Marks.	Outstanding, well presented code, diagrams and discussions that contain all key elements. Wide range	Excellent, well presented code, diagrams and discussions that contain all or most key elements. Research informed literature integrated into	Very good presentation of code, diagrams and discussions that contain most key elements. Literature used accurately but descriptively. Accurate and assured use of academic conventions.	Good presentation of code, diagrams and discussions that contain some key elements. Some evidence of reading, with superficial linking to given	Basic presentation of code, diagrams and discussions that contain a few key elements. Little evidence of reading and/or indiscriminate use	Poor presentation and report with little or no use of literature and academic conventions with significant errors are

	of relevant literature used to inform work. Consistently accurate and assured use of academic conventions. **8-10 marks**	work. Consistently accurate use of academic conventions. **7 marks**	**6 marks**	text(s) Some academic conventions evident and largely consistent, but with some weaknesses **5 marks**	of sources. Academic conventions used weakly. **4 marks**	present in formatting and referencing. **0-3 marks**
Total points: 100						

Section 2: General Assessment Brief Guidance

Supporting Assessment documentation, rules and regulations.

To view the academic rules and guidance documents for the topics listed below please follow this link to the Degree Apprenticeship Handbook (DAH) module in Canvas: https://canvas.qa.com/courses/1041

If you are unable to access this module please contact qaadegreeadmin@qa.com who will be able to resolve this for you.

Guidance found in the DAH:

- University of Roehampton Academic Regulations
- Regulations & Quality Assurance Overview
- Key contacts
- Mitigating Circumstance documentation
- Academic misconduct Procedure
- Final degree award calculation
- Appeals guidance
- Examination regulations
- Student feedback committees
- External examiner reports

Appendix A

ASSIGNMENT COVER SHEET

Student's name	(First name)	(Last name)
Module name		
Title of assignment		
Complete Word Count in my assignment		
Date submitted		

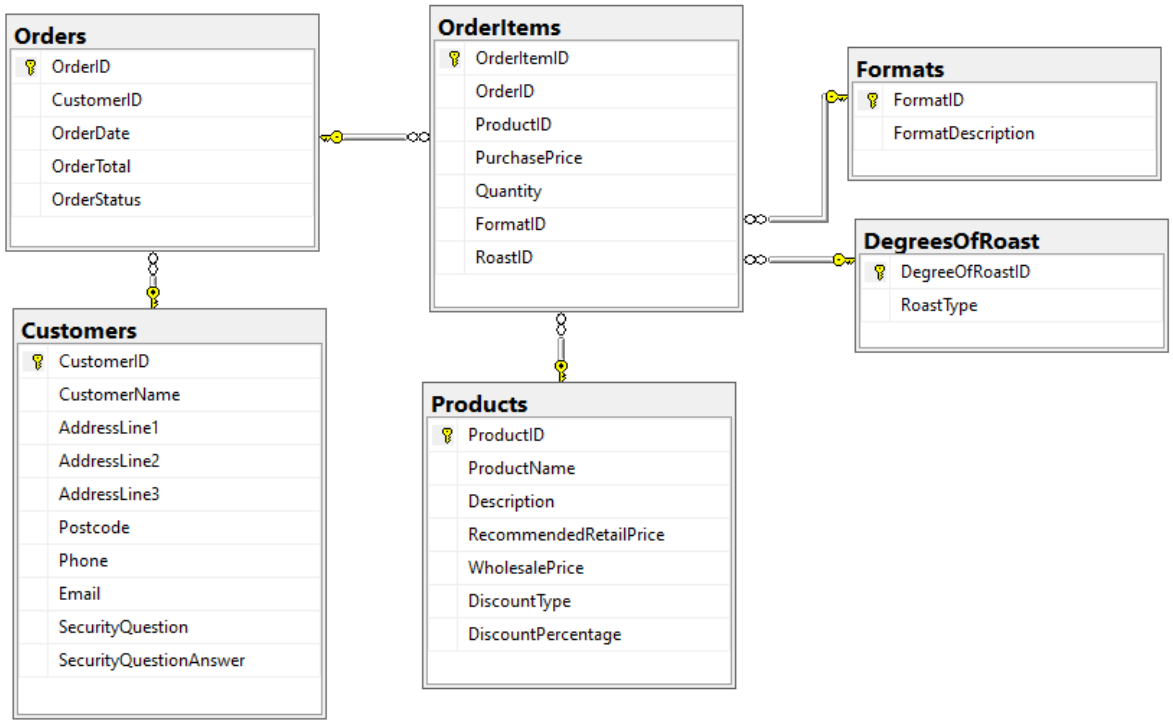
All work must be submitted by the due date. If an extension of time to submit work is required, a Mitigating Circumstances Extension Form must be submitted.

Has an extension been approved? **Yes** ☐ **No** ☐ **If yes, please give the new submission date ……./……./…….**

IMPORTANT: THIS STATEMENT MUST BE READ & SIGNED
Academic Integrity Statement Academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework which is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly. **I declare that the work I am submitting is my own work, is properly referenced and has not been submitted elsewhere.**
Student Signature (Full Name): **Date:**

Appendix B

Database Diagram



Customers Table

CustomerID: Unique numerical identifier (integer), primary key

CustomerName: Name of customer (string)

CustomerAddressLine1: First line of customer address (string)

CustomerAddressLine2: Second line of customer address (string – nullable)

CustomerAddressLine3: Third line of customer address (string – nullable)

Postcode: Customer's Postcode (string)

Phone: Customer's phone number (string)

Email: Customer's email address (string)

SecurityQuestion: A security question selected by the customer (string)

SecurityQuestionAnswer: Customer's answer to their security question (string)

Orders Table

OrderID: Unique numerical identifier (integer), primary key

CustomerID: Numerical identifier (integer), foreign key

OrderDate: Date order was placed (DateTime)

OrderTotal: Total value of order (money) (including any discounts)

OrderStatus: Numerical value indicating status of order (taken = 1, dispatched = 2)

OrderItems Table

OrderItemID: Unique numerical identifier (integer), primary key

OrderID: Unique numerical identifier (integer), foreign key to Orders table

ProductID: Unique numerical identifier (integer), foreign key to Products table

PurchasePrice: Wholesale price of product at time of purchase (money)

Quantity: Quantity ordered (integer)

FormatID: Unique numerical identifier (integer), foreign key Formats table. It is safe to assume the degree of FormatID is a set of consecutive numbers that range from 1 to 9.

DegreeOfRoastID: Unique numerical identifier (integer), foreign key to DegreesOfRoast table. It is safe to assume the RoastID is a set of consecutive numbers that range from 1 to 6.

Products Table

ProductID: Unique numerical identifier (integer), primary key

ProductName: Name of product (string)

Description: Short description of product (string)

RecommendedRetailPrice: Suggested retail price of product (money)

WholesalePrice: Price of product to wholesalers (money)

DiscountType: Single character that denotes what (if any) discount to apply (string), (B: BOGOF, P: Percentage, Null: No discount)

DiscountPercentage: Amount of discount to apply to product (int). Note, in the database this is implemented as a nullable integer.

Formats Table

FormatID: Unique numerical identifier (integer), primary key

FormatDescription: Coffee format (Beans, Coarse Ground, Medium Ground, etc.) (string)

DegreesOfRoast Table

RoastID: Unique numerical identifier (integer), primary key

RoastType: Roasting Style of (original) beans (Raw, Light, Medium, Dark, etc.) (string)

Appendix C

Project Source code

C#

Note the zip file contains a set of Microsoft Visual Studio projects and solution file. The version of Visual Studio used to create the code is Microsoft Visual Studio 2019 Community Edition using the .NET Framework 4.7.2. The application is designed to use a SQL Server database (2017) managed by V17.9 of SQL Server Management Studio configured to use Windows Authentication

DeansBeansC#Q12021Starter.zip

Java

Note the zip file contains a set of Eclipse projects (version 2020-09 (4.17.0)) and JavaSE – 12 (zulu-15). The application is designed to use a MySQL community server database (V8.0.13) managed by V8.0 of MySQL Workbench. The database has a user called “root” with a password of “password”.

DeansBeansC#Q12021Starter.zip