Solent University
Coursework Assessment Brief

Assessment Details

Unit Title:	Object Orientated Design and Programming
Unit Code:	COM504
Unit Leader:	Dr Craig Gallen
Level:	5
Assessment Title:	Time Constrained Assessment
Assessment Number:	AE1 (COM504TCA-2019-01)
Assessment Type:	Time Constrained Assessment
Restrictions on Time/Word Count:	2 Hours
Consequence of not meeting time limit:	It is essential that assignments keep within the time limit stated above. Any work beyond the maximum time length permitted will be disregarded and not accounted for in the final grade.*
Individual/Group:	Individual
Assessment Weighting:	40%
Issue Date:	
Hand In Date:	
Planned Feedback Date:	4 weeks after submission
Mode of Submission:	on-line
Anonymous Marking	This assessment: **Is exempt from anonymous marking.**

Assessment Task

Getting Started

You are required to use the class PC's and if necessary you should set up your machine to use maven as described previously in class. (For your convenience a maven-setup project is also provided in this COM504TCA-2019-01-student tca folder).

You will be given access to a ZIP FILE containing a maven project.

Download and extract this zip file to a folder in your user account on the C: drive on your machine.

Before you begin the assessment, make sure you can compile and run the project from netbeans and that you can access the web page of the application.

Contact your tutor if you have any problems at this stage.

You will have 2 hours to complete this assessment.

You may, if you find it useful, access your previous work through git or use any other on line resources which helps you to complete this exercise.

At the end of the assessment you will upload your corrected code to SOL

If you have any problems getting started, the instructions are not clear or you get seriously stuck because of a technical problem not related to this exercise, talk to your tutor.

Assessment

For this assessment you will be required to:

1. Read the scenario
2. Complete the use case and robustness diagrams
3. Complete the provided code demonstrating you understand how a complex object oriented project is constructed.

The full exercise begins in the section below; Store Project

Finishing up

At the end of the allocated time you will be asked to stop work.

To make saving the project manageable, clean out all target folders before zipping by using

mvn clean

Zip your cleaned git project, Name the zip file WITH YOUR NAME and submit it to sol. e.g. yourname-date-COM504TCA.zip

Where 'yourname' should be replaced with your name and date should be replaced with the current date. e.g. craiggallen-27-12-19-COM504TCA.zip

Upload the zipped file to SOL.

Assessment criteria

We will allocate 50% of marks to design documentation and 50% to implementation. There is no requirement to complete the documentation before you complete the code. However any documentation should reflect your implementation. Do as much as you can of all the tasks to maximise your overall score. So don't spend too much time on any one task.

Design and documentation (UML) (50%)

1 Complete use-case diagram 20%

2 Complete robustness diagram 25%

3 Ensure markdown references refer to your diagrams (gif) which describe usage of your code 5%

Implementation (50%)

4 Complete the user JSPs for modifying an item (25%)

The JSP's for this project are not complete and do not function as specified. Examine the JSP code and modify them to match the use case.

5. Complete the Rest Service so that the following URL works (25%)

GET http://localhost:8084/rest/example/retrieve?id=1

(where id can be an integer matching the id of a created item)

Project Brief: Store Inventory Project version 0.1

Scenario

A simple inventory system is required to keep records for a store of items. Each item has the following data;

- ID database / system id for the object.
- Stock Keeping Unit (SKU) - Unique identifier for the item.
- Description - description of the item.
- Price - unit prices of one instance of the item.
- Quantity - number of units in stock.

The user should be able to create new items in the store.

The user should be able to list all items in the store. Each entry in the list must also display the calculated value of the total number of items of each that type (price*quantity for each item). At the bottom of the list the system must also display the total value of ALL items in the store.

The user should be able to select and modify an item.

The user should be able to select and delete an item.

In addition, a ReST interface should be able to search for items using a template item object. The template allows a partly populated Item to be used to search for similar items. A search requires an exact match of fields in the template. All of the fields in the template are ANDed but null fields are ignored.

Tasks

In this exercise you will complete the design documentation and implementation of such a simple inventory.

All the draw.io diagrams and easyUML models should be placed in the uml sub-project

You are provided with a UML class diagram which documents the boundary interfaces that the system must use. These interfaces are already implemented for you.

The classes in the UML class diagram have been created in the model project.

You are provided with implementation code which supports the service described by the model.

In this exercise you will be asked to complete the JSPs in the web interface to allow a store keeper to add / delete or /search and update an item.

You also will be asked to complete the missing functionality for the ReST interface.

Part 1 - Design documentation

You are required to document the use case using a UML use case diagram with draw.io (https://www.draw.io/). You must save the drawing as both an image and a drawio.xml

It is STRONGLY suggested that you save your work as you go along to avoid any disasters.

You should start with the template draw.io drawings provided in the library-uml project.

You should save your use case diagram as a draw-io xml file along with draw-io images in the uml project (template files are provided).

You should then create a robustness diagram to expand the use case and show how JSPs and processes interact with the boundary interfaces provided. To guide you a partly completed robustness diagram is provided.

When you have finished, save both the drawio-xml and image files to your project so that they can be picked up by the README.MD file and displayed on git.

Your documentation should reflect your actual implementation and you may modify your documentation as you proceed.

Part 2 - Implementation

You should then proceed to implement the various methods required for the use cases.

Read the Project Structure section below for more information on what is provided and how to complete your code.

Skeleton code is provided for you and you may also reuse any examples from previous classes.

(If you choose to use any other external examples, code or libraries you MUST ATTRIBUTE THE WORK and incorporate the relevant LICENCE both in licence comments at the top of the relevant class where the work is used and in the README for the code).

store-service

The ServiceFacadeImpl.java mostly proxies methods from the ItemDAO which have already been tested in the item-dao-jaxb module. So there is no need to repeat tests of the proxied methods.

store-web

The web application is assembled in the store-web project. To run this application use netbeans to run tomcat.

The application should be available at http://localhost:8084/

Most of the web application has been implemented and it's behaviour can be used to help you complete the use case and robustness diagrams.

You will need to complete the implementation of the ListItems.jsp page.

You will need to complete the implementation of the ReST service ExampleProjectRestImpl.java

Project Structure

The following notes provide additional help with understanding the project structure within which you are to work.

This (store-parent) folder contains a partially completed project which you will complete as part of this exercise.

In the store-parent project, run maven with

```
mvn clean install
```

The project should build.

Import the store-parent project and its maven sub projects into netbeans. Also import the uml project.

The store-parent is a multi-module maven project which means that the parent project causes the sub projects to be built in the order determined by the following section in the pom.xml

```
    <modules>
        <module>model</module>
        <module>dao-jaxb</module>
        <module>service</module>
        <module>web</module>
        <module>rest-client</module>
    </modules>
```

store-uml

The store-uml project contains the easyUML uml model, use cases, robustness diagrams and class diagrams for this project. Do not change the class diagram but do use the robustness diagrams and use case diagram templates. Save your completed use cases and robustness diagrams here both as xml and as gif images so that they show up in the README.md in git.

store-model

The library-model project contains model classes and interfaces which have been generated and then expanded from the store-uml class diagrams. You should implement your code using interfaces from this model. DO NOT change this module.

store-dao-jaxb

An implementation of the ItemDAO is provided for you to use in the store-dao-jaxb project.

Look at the tests to understand how the DAO can be used in your service.

DO NOT change this module.

store-service

The store-service project provides an implementation of the ServiceFacade which is a proxy for the StoreDAO.

store-web

The store-web project contains partly completed JSP's.

Some of this JSP is implemented for you however you must Complete the JSP's to allow a store person to update the store.

The store-web project contains a partly completed ReST Service. You should complete the ReST service functionality.

Learning Outcomes

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptors.

Late Submissions

Students are reminded that:

i. If this assessment is submitted late i.e. within 5 working days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
ii. If this assessment is submitted later than 5 working days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
iii. If this assessment is being submitted as a referred piece of work then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-principles-and-regulations.pdf?t=1534423842941

Extenuating Circumstances

The University's Extenuating Circumstances procedure is in place if there are genuine circumstances that may prevent a student submitting an assessment. If students are not 'fit to study', they can either request an extension to the submission deadline of 5 working days or they can request to submit the assessment at the next opportunity (Defer). In both instances students must submit an EC application with relevant evidence. If accepted by the EC Panel there will be no academic penalty for late submission or non-submission dependent on what is requested. Students are reminded that EC covers only short term issues (20 working days) and that if they experience longer term matters that impact on learning then they must contact the Student Hub for advice.

A summary of guidance notes for students is given below:

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2p-extenuating-circumstances.pdf?t=1534423896787

Academic Misconduct

Any submission must be students' own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University's Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. Students should check this link before submitting their work.

Procedures relating to student academic misconduct are given below:

http://portal.solent.ac.uk/support/official-documents/information-for-students/complaints-conduct/student-academic-misconduct.aspx

Ethics Policy

The work being carried out by students must be in compliance with the Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then students will need an ethics release or an ethical approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:
http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2s-university-ethics-policy.pdf

Grade marking
The University uses a letter grade scale for the marking of assessments. Unless students have been specifically informed otherwise their marked assignment will be awarded a letter grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-annex-2-assessment-regulations-grade-marking-scale.pdf?t=1534424273208

Guidance for online submission through Solent Online Learning (SOL)

http://learn.solent.ac.uk/onlinesubmission