PAPER C220=MC220

# SOFTWARE ENGINEERING DESIGN

Friday 1 May 2020, 11:00
Duration: 120 minutes
NO post-processing time
*Answer TWO questions*

While this time-limited remote assessment has not been designed to be open book, in the present circumstances it is being run as an open-book examination. We have worked hard to create exams that assesses synthesis of knowledge rather than factual recall. Thus, access to the internet, notes or other sources of factual information in the time provided will not be helpful and may well limit your time to successfully synthesise the answers required.
Where individual questions rely more on factual recall and may therefore be less discriminatory in an open book context, we may compare the performance on these questions to similar style questions in previous years and we may scale or ignore the marks associated with such questions or parts of the questions. In all examinations we will analyse exam performance against previous performance and against data from previous years and use an evidence-based approach to maintain a fair and robust examination. As with all exams, the best strategy is to read the question carefully and answer as fully as possible, taking account of the time and number of marks available.

Paper contains 2 questions

## General Information

This is a remote exam involving programming. To do it you will need:

- a computer with internet connection
- a Java development environment installed (IntelliJ IDEA is recommended)

## AnswerBook

All your answers should be submitted electronically by accessing the AnswerBook website at https://co220.doc.ic.ac.uk/exam using a standard web browser.
Log in to AnswerBook using your standard College username and password.

All work that you want marked should be entered into AnswerBook.
You should not upload anything to CATE or push anything to GitLab.

## Skeleton Code

Download https://www.doc.ic.ac.uk/~rbc/exam-materials/CO220-exam.zip which contains code related to the exam questions. Work on this code on your computer using your IDE.

## Using IntelliJ IDEA

To try to simplify things as much as possible, we have included an IntelliJ project configuration (.idea directory) inside the zip file, and no Gradle files. That means that if you unzip the file, you should just be able to *Open* the IntelliJ project (not *Import Project*). You should see two directories: Q1 and Q2, each of which contains a blue src directory and a green test directory.

The project is configured to expect JDK 11, but if you have a different JDK on your machine, you will probably see a "Setup JDK" link at the top of the screen. Click this and select your JDK (it shouldn't really matter what version it is).

The lib folder contains all the testing libraries that we used during the course. These should be added into your classpath automatically if you open the project as above, but if you have problems you can add them manually.

There is no build.sh or suite of automated tests/checks for you to pass, but do aim to use good code style and clear formatting in your solutions.

## Answer the Questions

Answer the questions on the following pages using the downloaded code. When you are happy with your answer for each part, paste your code into the relevant boxes on the AnswerBook site. Use the button top-right to save your work. You can update your answer and save again as often as you want. We recommend saving after you have answered each part.

## When You Have Finished

At the end of the exam, or when you have finished, check that you are happy with your work and that all your answers are saved in AnswerBook.

1 **Test Driven Design**

For this question, look at the Sequence Diagram "Book Sales" on the next page and the code given in the Q1 directory (there is not much code there).

a Using test-driven development and mock objects, iteratively develop an implementation of OrderProcessor according to the scenarios illustrated.

Hint: you do not need to implement Client.

i) Show your first test and the implementation to make it pass.

ii) Show your second test and the implementation to make it pass.

iii) Show your third test and the implementation to make it pass.

b Considering the code that you have written for OrderProcessor and its collaborators, and particularly the methods that you have defined, answer the following questions. Give examples from your code.

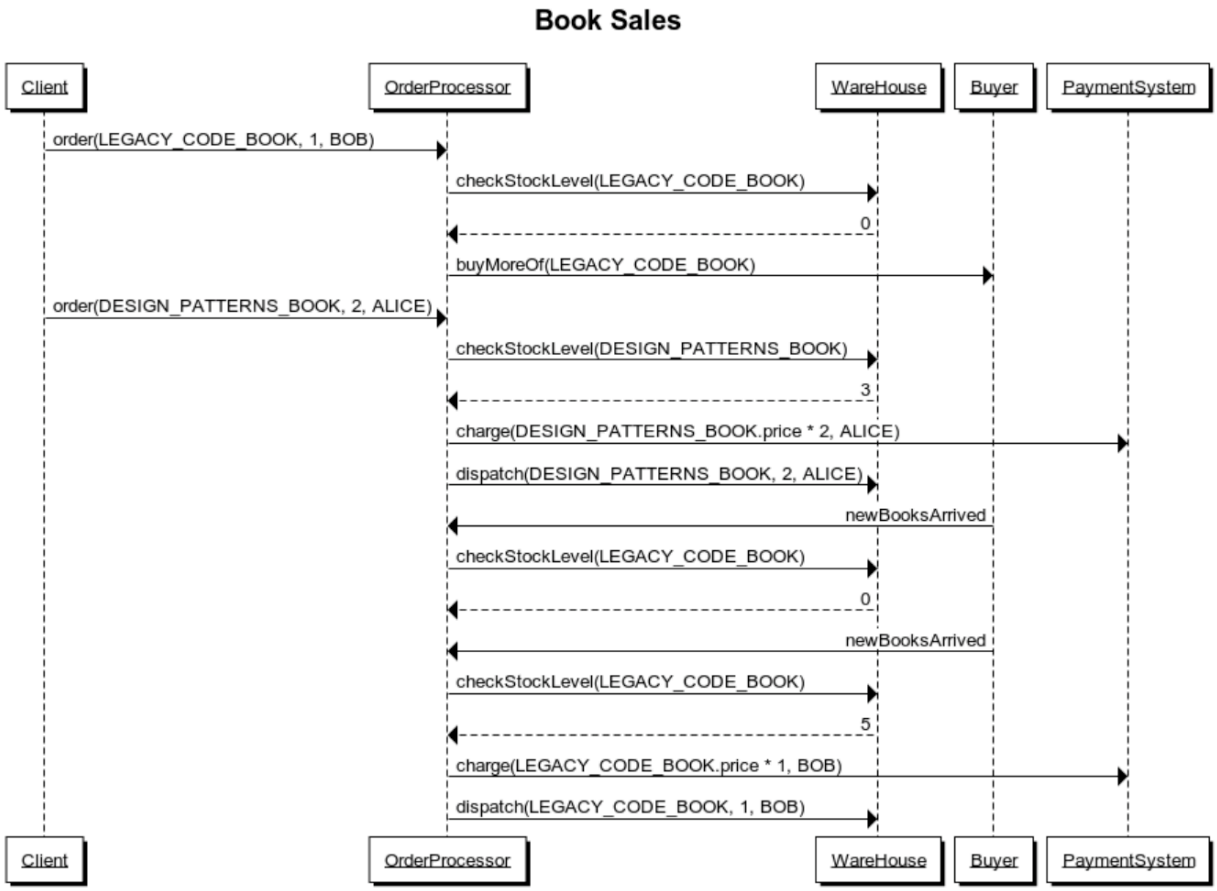
i) What is the difference between a *command* and a *query*?

ii) From your code in part a) identify one command and one query.

iii) What is the benefit of designing code based largely on commands?

iv) What is the general name for this style of design?

*The two parts carry, respectively, 70% and 30% of the marks.*




Sequence Diagram for Question 1

2 **Working with Legacy Code**

For this question, look at the code provided to you in the directory Q2.

a Looking at the code for AlgoTrader, what design pattern can you see in use that makes unit testing of this code difficult? Give the location in the code.

b To improve the future maintainability of this code, we might want to isolate the code in package ic.doc from the code in com.londonstockexchange.

i) Name a design pattern that can be used for this purpose, and show how you would implement it (show your code).

ii) Name the architectural style that this relates to.

c By applying the principle of Dependency Inversion, refactor the code in package ic.doc to make it more flexible, and to enable effective unit testing.

d Once the code is in a better state, write two unit tests for its core behaviour.

*The four parts carry, respectively, 10%, 40%, 20% and 30% of the marks.*