IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2023

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BEng Honours Degree in Mathematics and Computer Science Part II
MEng Honours Degree in Mathematics and Computer Science Part II
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER COMP50002

SOFTWARE ENGINEERING DESIGN

Wednesday 3rd May 2023, 10:00
Duration: 120 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

Question 1 - Design Patterns

Look at the code in the `flights` package, under Q1. The code in class `Example` in the `bookings` package shows how these classes can be used.

a i) Name the design pattern used for constructing new `FlightNumber` objects.

ii) Name the design pattern implemented by the `SeatManager` class.

b Look at the classes `EconomyFlight` and `BusinessClassFlight`. There is quite a lot of duplication between these two classes. Reduce this duplication by applying the Template Method Pattern.

Show the relevant code.

c Imagine you work on the Booking team, so you can change any of the code in the `bookings` package, and add new code there, but you cannot change the code in the other packages provided.

We want to write some tests for the logic in `BusinessClassFlight`, but the `SeatManager` is not deterministic — it gives different patterns of available seats each time it is used.

Refactor the code to give us the option to test the code in `BusinessClassFlight` without invoking the `SeatManager` class, which is owned by the Availability team.

Show the relevant code. Remember that you may not edit `SeatManager`.

d Given your solution to part c) above, write three tests showing different aspects of the behaviour of the `seatingOptions()` method in `BusinessClassFlight`. Make sure your tests are deterministic.

Show the relevant code.

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

Question 2 - Test-Driven Development

For this question, look at the Sequence Diagram on the next page and the code given in the Q2 directory (there is not much code there). In the illustrated scenario, Charlie is an Author, and Alice and Bob are Subscribers.

Using test-driven development and mock objects, iteratively develop an implementation of an article repository that supports notifications to subscribers when articles related to their interests are published.

For each of the following behaviours, write a test and the implementation to make it pass. Show the code you add/change at each step.

a On publication, articles are summarised to extract keywords.

b Subscribers with matching interests are notified on publication of a new article.

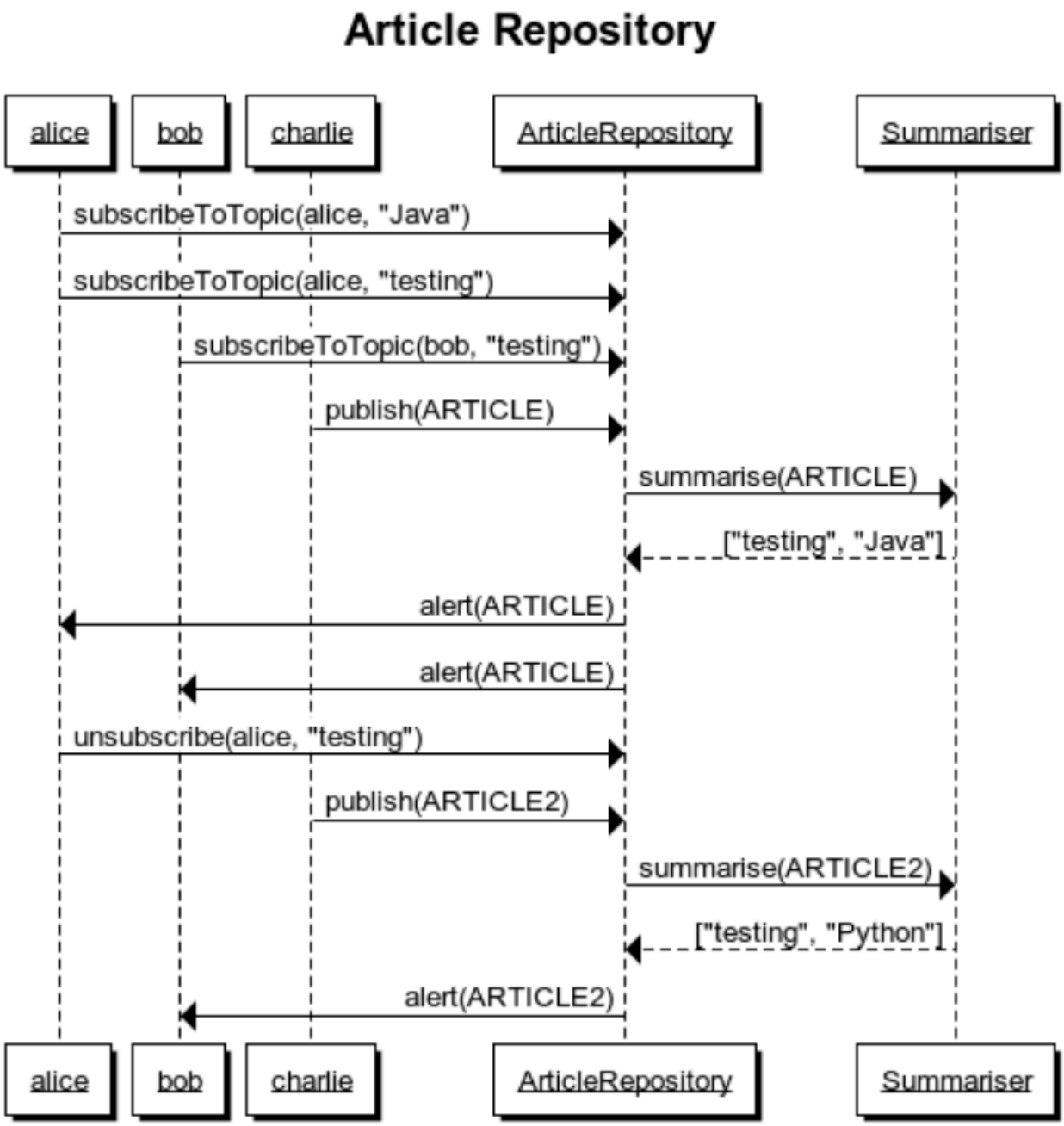
c Subscribers without matching interests are not notified on publication.

d Subscribers with multiple matching interests are only notified once per article.

e If a subscriber removes an interest, they no longer receive alerts for that topic.

Hint: *If you create multiple mocks of the same type, you may see the message "a mock with name … already exists". If this happens you need to give each mock of the same type a distinct name (a string). Pass the name in as a second parameter to the mock() method.*

Each of the five parts carries 20% of the marks.



Sequence Diagram relating to Question 2