

IMPERIAL COLLEGE LONDON

TIMED REMOTE ASSESSMENTS 2021-2022

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 assessments for the
Associateship of the City and Guilds of London Institute*

PAPER COMP50002

SOFTWARE ENGINEERING DESIGN

Thursday 12 May 2022, 10:00

Writing time: 120 minutes

Upload time: 15 minutes

Answer ALL TWO questions

Open book assessment

This time-limited remote assessment has been designed to be open book. You may use resources which have been identified by the examiner to complete the assessment and are included in the instructions for the examination. You must not use any additional resources when completing this assessment.

The use of the work of another student, past or present, constitutes plagiarism. Giving your work to another student to use constitutes an offence. Collusion is a form of plagiarism and will be treated in a similar manner. This is an individual assessment and thus should be completed solely by you. The College will investigate all instances where an examination or assessment offence is reported or suspected, using plagiarism software, vivas and other tools, and apply appropriate penalties to students. 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 assessment 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 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. Paste your code into the boxes on AnswerBook. Do not try to upload PDFs or images. You should not upload anything to CATE or push anything to GitLab.

Skeleton Code

Download <https://www.doc.ic.ac.uk/~rbc/2122-exams/SED-exam.zip> which contains code related to the exam questions. Unzip this file and work on this code on your computer using your IDE.

Using IntelliJ IDEA

If you open the project in IntelliJ IDEA, you should see two directories, Q1 and Q2, each of which should contain a blue src directory and a green test directory. **You will probably have to set the JDK.** See <https://www.doc.ic.ac.uk/~rbc/SED/> for details.

The project should already be configured with all the testing libraries that we used during the course, in case you need to use them in answering the questions.

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. The code will not be auto-tested, it will be read by a human.

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

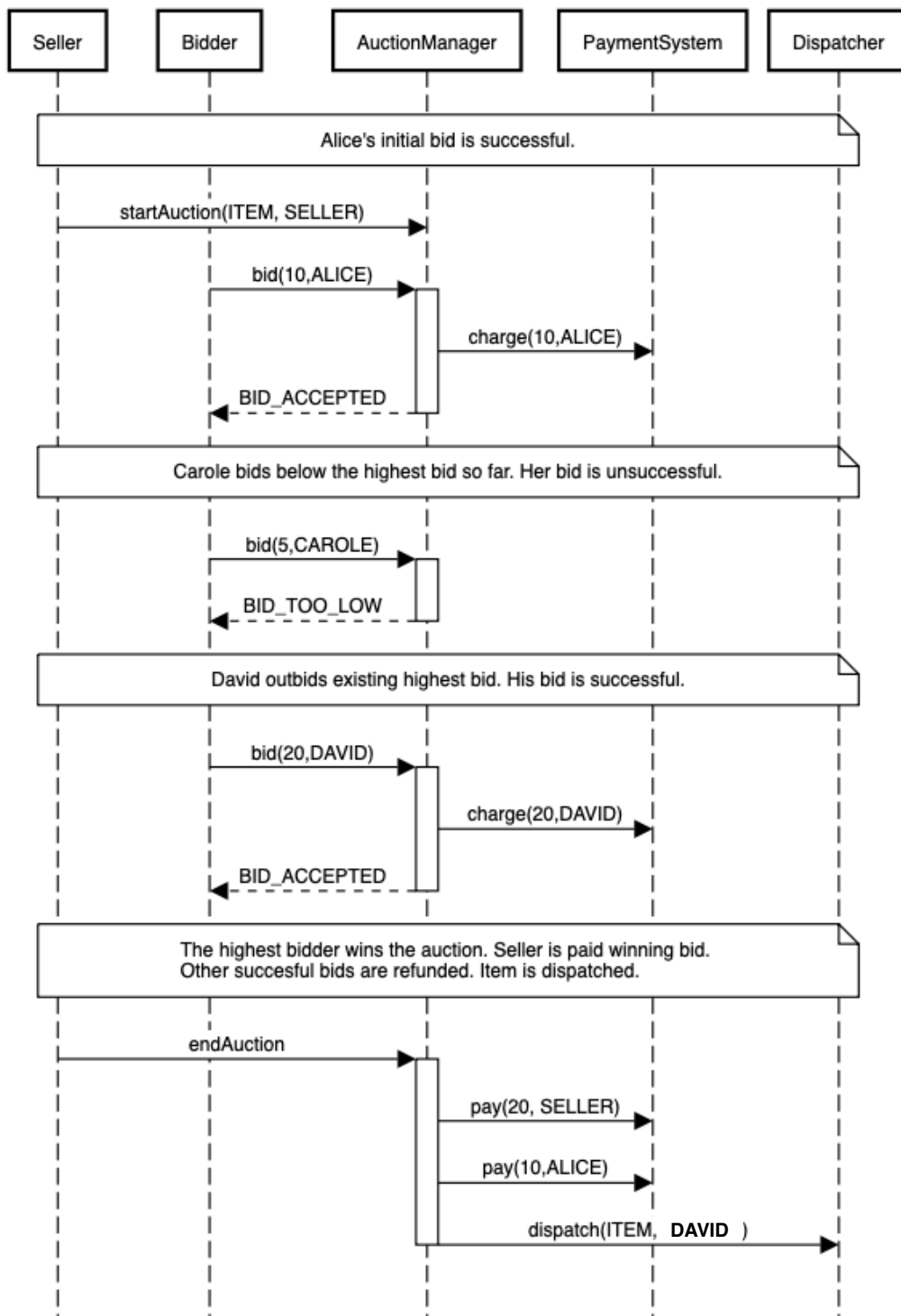
For this question, look at the Sequence Diagram on the next page and the code given in the Q1 directory (there is not much code there).

Using test-driven development and mock objects, iteratively develop an implementation of an auction manager according to the illustrated scenario.

Hint: you do not need to implement Seller and Bidder.

- a) Show your first test and the implementation to make it pass.
- b) Show your second test and the implementation to make it pass.
- c) Show your third test and the implementation to make it pass.
- d) Show your fourth test and the implementation to make it pass.

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



2 Video Streaming

Look at the code under the folder Q2, which contains a few different classes forming part of a system for streaming movies, and a couple of simple tests.

- a The MediaLibrary could be very large if it contains a large index of movie recommendations. Update the code using an appropriate design pattern to prevent more than one instance of this class being created.

Name the pattern you used and show the relevant parts of your code.

- b Now look at class VideoStreamer. VideoStreamerTest shows a simple example of its usage. Apply the dependency inversion principle and hence write a test for the behaviour of getSuggestedMovies() without invoking the (potentially large) MediaLibrary object.
- c If a user watches a particular stream for more than 15 minutes, the system logs that they watched the movie (to enhance future suggestions) using a PlaybackEventLog. Make appropriate changes to the code to allow you to write tests that verify this behaviour - then write a test. Do not change the signatures of the existing public methods in VideoStreamer.

Show the relevant parts of the code that you have changed.

The three parts carry, respectively, 20%, 30% and 50% of the marks.

APPENDIX *Provided Code*

In the exam this will be provided as a download for candidates to work on. These pages should not appear in the final exam paper, they are for examiners' reference.

Q1

BidResult.java

```
package ic.doc.auction;

public enum BidResult {BID_ACCEPTED, BID_TOO_LOW};
```

AuctionManager.java

```
package ic.doc.auction;

public class AuctionManager {

}
```

AuctionManagerTest.java

```
package ic.doc.auction;

public class AuctionManagerTest {

}
```

Q2

```
public class VideoStreamer {
    private final Map<VideoStream, StreamTracker> currentStreams = new HashMap<>();
    private final PlaybackEventLog playbackEvents = new ConsolePlaybackEventLog();

    public List<Movie> getSuggestedMovies(User user) {
        List<Movie> recommendations = new MediaLibrary().recommendedMoviesFor(user);

        // sort the list of suggestions in descending order of number of views
        List<Movie> suggestions = new ArrayList<>(recommendations);
        suggestions.sort(Comparator.comparing(Movie::numberOfViews).reversed());
        return suggestions;
    }

    public VideoStream startStreaming(Movie movie, User user) {
        VideoStream stream = new VideoStream(movie);
        currentStreams.put(stream, new StreamTracker(user));
        return stream;
    }

    public void stopStreaming(VideoStream stream) {
```

```

        StreamTracker streamTracker = currentStreams.remove(stream);
        LocalTime endTime = LocalTime.now();
        long minutesWatched = ChronoUnit.MINUTES.between(streamTracker.startTime(), end-
Time);
        if (minutesWatched > 15) {
            playbackEvents.logWatched(streamTracker.user(), stream.movie());
        }
    }
}

```

```

public class MediaLibrary {

    private final List<Movie> topMovies;

    public MediaLibrary() {
        topMovies = List.of(
            new Movie("Jurassic Park", ... etc etc ));
    }

    public List<Movie> recommendedMoviesFor(User user) {

        // A sophisticated ML algorithm runs and then recommends...

        if (user.isChild()) {
            return topMovies.stream().filter(Movie::isSuitableForChildren).toList();
        } else {
            return topMovies;
        }
    }
}

```

```

public class VideoStreamerTest {

    @Test
    public void allowsUserToStreamSuggestedMovies() {

        VideoStreamer streamer = new VideoStreamer();
        User user = new User("Adam", 9);

        List<Movie> movies = streamer.getSuggestedMovies(user);
        VideoStream stream = streamer.startStreaming(movies.get(0), user);

        // adam watches the movie

        streamer.stopStreaming(stream);
    }
}

```