

Wednesday, 13th May 2015  
09.30am – 11.00am  
(1 hour 30 minutes)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

# **COMPUTING SCIENCE 2P: JAVA AND OBJECT ORIENTED SOFTWARE ENGINEERING 2**

**Answer all 4 questions**

**This examination paper is worth a total of 60 marks.**

**For examinations of at least 2 hours duration, no candidate shall be allowed to leave the examination room within the first hour or the last half-hour of the examination.**

**INSTRUCTIONS TO INVIGILATORS: Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.**

**1.** This question is about the Java programming language.

(a) How long, in bits, is a long value? [1]

(b) How long, in bits, is a short value? [1]

(c) What is the output of the following statement?

```
System.out.println("10"+1);
```

[1]

(d) What is the output of the following statement?

```
System.out.println(10+1);
```

[1]

(e) Given a String array named args, write a code fragment that iterates over each element of args and outputs the number of characters in that element. [3]

(f) Define a private class method named maxOfThree that takes three int arguments and returns the largest of the three values. [5]

(g) Describe briefly how you would change your code from the previous question to find the maximum value from an arbitrary number of int values. [3]

2. This question is about modelling computers in a data centre. The computing resources may be rented by customers for set periods of time. Each computer in the data centre has a fixed number of CPUs, an amount of RAM and an amount of persistent storage. RAM is measured in integer values of MB. Storage is measured in real values of TB. Each computer is uniquely identified by a non-negative integer value. Each computer may be available for use or unavailable (down for maintenance).

- (a) Write a class definition for a `Computer`, incorporating all the attributes outlined above. You should also define a public constructor for `Computer` which should initialize all instance fields to sensible values. The constructor should assign a unique identifier value to each instance. You do not need to define any instance methods. [6]
- (b) Consider a `Customer` class, with an instance method `isNew()` that returns `true` if the customer is a recently signed up customer and `false` if the customer has been using the data centre for more than 12 months. Recent customers can rent a computer for free, whereas non-new customers have to pay for daily use, according to the following formula:

$$\text{dailyPriceInPence} = \text{floor}(\text{numCPUs} \times \text{RAM} \times \text{storage})$$

where RAM is measured in MB, storage in TB and  $\text{floor}(x)$  returns the largest integer that is not greater than  $x$ . Note that the minimum rental period is a single day.

Define a public instance method for the `Computer` class called `getDailyPrice` that takes a single `Customer` argument and returns the daily price in pence for that customer to rent that computer. [4]

- (c) Now suppose that there are two subclasses of `Customer`. The `Subscriber` subclass represents individuals who pay a flat monthly subscription and can rent computers without paying a daily price. The `Staff` subclass represents individuals who work for the data centre and can rent computers without paying a daily price or a subscription. Describe how you would modify the `Computer.getDailyPrice` method to handle `Customer`, `Subscriber` and `Staff` instances. You may illustrate your answer with fragments of Java source code, but this is not essential. [5]

3. This question concerns object oriented analysis and design.

Consider the following problem description carefully and then answer the questions below.

An auction house wishes to develop a software system for running auctions online. An auction is the sale of one or more items according to particular rules. A user of the system who wishes to make a sale will create a *lot*, which will include an item description and a quantity available. For example, a seller might create a lot described as “Leather wallet, new, 10x12cm.”, with a quantity of 5. Users who wish to purchase items (buyers) place *bids* on the auction.

There are several different types of auction run by the house, which have different rules for placing bids:

- *English* auctions require the seller to set a starting price for the item. The first bid must be at or over this price. Subsequent bidders must offer over the highest currently recorded bid. Bidders are permitted to view the entire bid history for an item. The winners of the auction are the highest bidders according to the quantity of items in the lot.
- A *Dutch* auction starts with a high price that is periodically lowered (to a schedule set by the seller) until a buyer makes a bid. In this case, a bidder automatically wins an item when they place a bid. Subsequent bids can be at the same price or wait for the price to drop. The auction continues with the price descending until all items in the lot are sold.
- A *Blind* auction starts with a guide price. Buyers submit bids at, above or below this price as they think best, without knowing the value (or quantity) of other bids being made. When the auction ends, the highest bid is the winner.

A seller can set a reserve price for any type of auction, below which an item cannot be sold.

An auction can run until no further bids are received for a period of time, until some fixed deadline, or when the seller decides to close the auction.

When an auction ends, the winning buyers’ accounts should be debited by the amount they bid and the seller should be credited the same amount.

The auction house has already determined that they will need a class called Auction in their system, but have not yet given it any attributes.

- (a) Develop a class diagram of the auction problem domain that includes an Auction class. You should identify the key classes in the domain and their attributes, behaviours and relationships. In addition, you should make use of any opportunities to refine the diagram using inheritance and/or design patterns as appropriate.

[7]

The following two questions concern design patterns. You may use diagrams or Java code fragments (these do *not* need to be in perfect syntax) if it helps your explanation.

- (b) Some users have expressed a desire to be notified when a bid occurs. There are several different reasons for this. For example, some bidding users wish to implement

automated software that will automatically increase their bids up to a limit if they are outbid. Some sellers want to analyse the bidding activity across their lots in real time to improve sales.

Propose a design pattern to address this problem and explain how it would be tailored to the problem context. Bear in mind that some auctions do not permit users to see bids until the auction ends.

[5]

- (c) The auction house has developed a hierarchical category taxonomy to help improve the structure of item descriptions. For example, if a motorbike is listed under the category “transport vehicle”, the seller can add a “top speed” and “miles per gallon” value to the description that can subsequently be formatted in a consistent manner for display. Unfortunately, many items fit into more than one category in the taxonomy. Motorbike, for example, also fits in the category “motorsport”, which allows the user to add the field “sport type” to the description.

Propose a design pattern to address this problem and explain how it would be tailored to the problem context.

[3]

4. This question is about software testing.

(a) Explain the terms *slip*, *defect* and *failure* in the context of defect testing.

[3]

The following collection of Java interfaces defines the behaviour for an interactive search engine. Examine the interfaces and the accompanying description and then answer the question below.

```
public interface SearchSession {

    public void updateSearchWithBlocking(
        Set<String> keyWords);

    public void updateSearchWithCallBack(
        Set<String> keyWords, SearchSessionCallBack callBack);

    public List<SearchResultItem> getSearchResults();
}

public interface SearchResultItem {

    public Double getRanking();

    public URL getDocumentURL();
}

public interface SearchSessionCallBack {
    public void notifySearchComplete();
}
```

The main interface is `SearchSession` which allows a user to update a search using the `updateSearchWithBlocking()` method. The result of the current query can be accessed using the `getSearchResults()` method. Each item in the result has a ranking, indicating how well the document matches the query, and a URL for accessing the actual document from the search. Items in the results are comparable and sorted by ranking. Searching can take some time, which may mean that the `getSearchResults()` method will return a null value.

Rather than continuously polling this method, a user can instead use the `updateSearchWithCallBack()` to register an object that implements the `SearchSessionCallBack` interface. This callback object will be notified when the results of a search are complete.

In the following questions, you may use Java code if it helps to explain your answer. Any code you provide does not need to be syntax perfect, but the intention should be clear.

(b) Assume that you are creating a test class (in a test framework such as JUnit) for an implementation class of the `SearchSession` interface called `FileSystemSearchSession` that is used to search files on a local file system. Explain how you would create a *fixture* for the `FileSystemSearchSession` class. You may assume that `FileSystemSearchSession` has a zero-parameter constructor.

[3]

- (c) For testing purposes, you also have a file system which, when the query set {joose, jeremy} is used, should return search results with the following properties:

URL	ranking
file://C:/Users/testuser/joose-lecturers.txt	0.95
file://C:/Users/testuser/joose-exam.txt	0.50

Explain how you would create a test for this result.

[4]

- (d) Explain how you would create a test for the callback mechanism to guarantee that the `getSearchResults()` method does not return null.

[5]