University of Glasgow

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]

Solution: 64 bits

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

Solution: 16 bits

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

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

[1]

Solution: 101

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

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

[1]

Solution: 11

(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]

Solution:

- 1 mark for a for/each loop, or an appropriate other loop/iterator construct
- 1 mark for a String temporary variable for the current element
- 0.5 marks for calling String.length()
- 0.5 marks for println

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

Solution:

- 0.5 marks for `private`
- 0.5 marks for `static`
- 1 mark for three `int` params `maxOfThree(int a, int b, int c)`

- 1 mark for sensible logic - either relational comparators or Math.max() use
- 1 mark for use of conditionals - either if/else or ternary ?: / or for nested Math.max() use
- 1 mark for correct code

(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]

Solution:

- 1 mark: Use varargs maxOfN(int ...values) or a Collection (ArrayList?) or array of int values
- 1 mark: iterate over values
- 1 mark: keep a maxSoFar variable, compare against each value in array, return at end.

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]

Solution:

```
        public class Computer {
// 1 mark for class definition
          private static int nextId = 0;
// 0.5 marks for static field
          private int id;
// 0.5 mark for three int fields
          private int numCpus;
          private int ram;
          private double storage;
// 0.5 marks for double field
          private boolean available;
// 0.5 marks for boolean field

          public Computer(int numCpus, int ram, double storage) {
// 1 mark for signature
            this.id = Computer.nextId++;
// 1 mark for id assignment
            this.numCpus = numCpus;
// 1 mark for all other assignments
            this.ram = ram;
            this.storage = storage;
            this.available = false;
          }
        }
```

(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:

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

where RAM is measured in MB, storage in TB and 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]

Solution:

```
          public int getDailyPrice(Customer c) {
// 1 mark for signature
            int pence = 0;
// 0.5 marks for int local
            if (!c.isNew()) {
// 1 mark for test
              pence = (int)(this.numCPUs * this.ram * this.storage);
// 0.5 marks for formula, 0.5 for cast
            }
            return pence;
// 0.5 marks for return
          }
```

(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]

Solution: There are lots of possibilities—some of which are more elegant than others.

- If the student suggests using `instanceof` with if/then/else tests only award **1 mark**.
- If the student suggests adding a new `boolean` method to the `Customer` class called `shouldPay()` and this method can be overridden in subclasses and called in `getDailyPrice()` then award up to **4 marks**.
- If the student suggests using the *Visitor pattern* with the customer instance being the visitors and the computer instance being the elements. The `getDailyPrice()` method is analogous to `accept` and each customer should have a corresponding `visit` method that computes the price to pay and is a polymorphic call. Award up to **5 marks**.
- If the student suggests using *double dispatch* or *multimethods* and can explain coherently what these are then award up to **5 marks**.
- Other reasonable solutions, e.g. setting up a separate `BillingSystem` class that handles Computer and Customer instances as pairs, will be given reasonable credit too.

If the student suggests more than one of the above approaches, then sum credit up to the maximum of 5 marks.

3. This question concerns object oriented analysis and design.

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

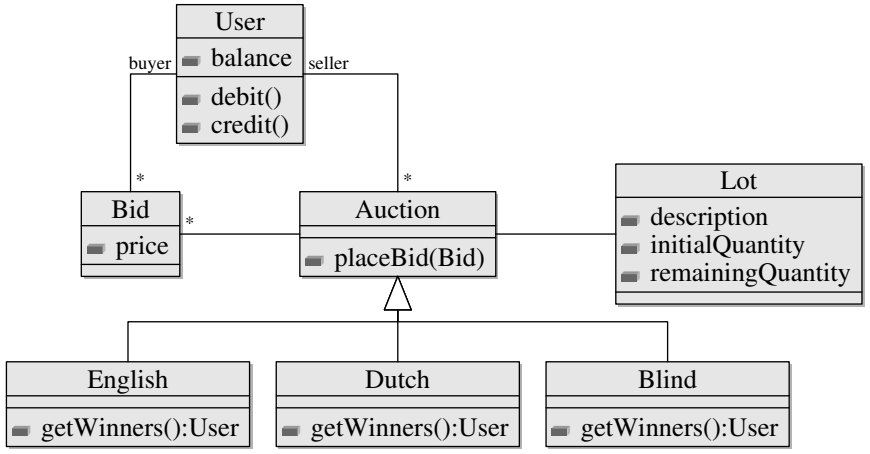
> An auction house wishes to develop a software system for running auctions online. An auction is the sale of one or more items according to particular rules. A user of the system who wishes to make a sale will create a *lot*, which will include an item description and a quantity available. For example, 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 at features.

(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]

Solution:



There is some variation possible here, but a sample solution is provided below.

4 Identification of `User`, `Bid`, `Auction` and `Lot` as domain model classes with appropriate attributes and behaviours.

1 Appropriate use of associations, labels and arities.

2 Appropriate use of inheritance with the `getWinners()` method as distinguishing behaviour for the different types of auction.

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]

Solution:

1 The observer-observable pattern.

1 A new sub-class of `Auction` is needed called `ObservableAuction`, which will have `DutchAuction` and `EnglishAuction` as subclasses.

1 There will also be a new interface called `AuctionObserver` and a new class called `BidEvent` to hold the details of a new bid..

2 `ObservableAuction` will have a single method called `addAuctionObserver(AuctionObserver)` and `AuctionObserver` will have a single method `newBid(BidEvent)`.

(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]

Solution:

1 The player role pattern.

1 Each item (the player) will be associated with many different categories (roles).

1 The item will iterate over the descriptions using a common method `getDescription():String` to produce the item description.

4. This question is about software testing.

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

[3]

Solution:
Book work:

1 Failures are observations of a system implementation's deviation from *expected* behaviour.

1 Defects (or bugs, faults) are the adjudged or deduced causes of failure in a system, an incorrect variable assignment, or an invalid pointer reference, for example.

1 Slips(or errors) are the actions by programmers that introduce defects into a software system.

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]

Solution:

Sample code for following answers.

```
public class FileSystemSearchSessionTest {

 private SearchSession searchSession;
 private URL file1;
 private URL file2;
 private URL file3;

 private Set<String> query1;

 @Before
 public void setUp() throws Exception {
  searchSession = new FileSystemSearchSession();
  file1 = new URL(
   "file://C:/Users/testuser/tim-software.txt");
  file2 = new URL(
   "file://C:/Users/testuser/joose-lecturers.txt");
  file3 = new URL(
   "file://C:/Users/testuser/joose-exam.txt");

  query1 = new HashSet<String>();
  query1.add("joose");
  query1.add("jeremy");
 }

 @Test
 public void testSearchWithBlocking() {
  searchSession.updateSearchWithBlocking(query1);
  List<SearchResultItem> searchResults = searchSession
   .getSearchResults();

  assertEquals(2, searchResults.size());
  assertEquals(file2, searchResults.get(0)
   .getDocumentURL());
  assertEquals(0.95, searchResults.get(0)
   .getRanking(), 0.0);
```

```
 assertEquals(file3, searchResults.get(1)
  .getDocumentURL());
 assertEquals(0.95, searchResults.get(1)
  .getRanking(), 0.0);
}

@Test
public void testSearchWithCallBack() throws Exception {

 StubSearchSessionCallBack callBack =
  new StubSearchSessionCallBack();

 searchSession.updateSearchWithCallBack(query1,callBack);

 while (!callBack.searchComplete) Thread.sleep(1);

 assertNotNull(searchSession.getSearchResults());
}

public class StubSearchSessionCallBack
 implements SearchSessionCallBack {

 public Boolean searchComplete = false;

 @Override
 public void notifySearchComplete() {
  searchComplete = true;

 }
}
```

1 private field with interface type.

2 a setup method that initialises the field with the concrete type.

(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]

Solution:

1 Extend the fixture with the specified files and query set.

1 Create a test method (with correct annotation).

1 Exercise the blocking query method.

1 Create assertEquals for the various features of the results.

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

[5]

Solution:

2 Create a stub callback class with a single boolean attribute. This attribute is set when the notify method is invoked.

1 Create a test method that invokes the updateSearchWithCallBack method (basically same as above), but with the extra callback instance argument.

1 Create a while loop over a thread interrupt until the boolean attribute in the stub callback is ture.

1 test that a subsequent call to getSearchResults is not null.