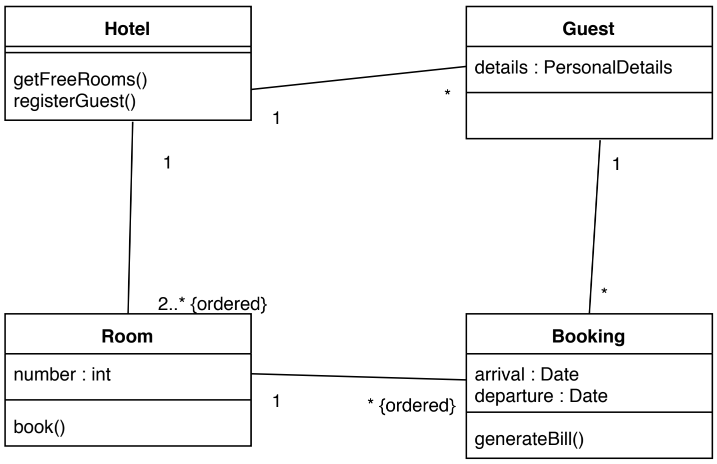
Module Title: Inf2C-SE: Introduction to Software Engineering
Exam Diet: Dec 2018
Brief notes on answers:

1. (a) *(Application of knowledge)* Answer expected similar to



Marking:
2 marks: Correct UML class diagram notation 2 marks: reasonable associations
2 mark: reasonable multiplicities
1 mark: reasonable operations
1 mark: Judgement that Date is best shown simply as an attribute type. (PersonalDetails could be shown as an attribute or connected to by an association. Either is fine.)
1 mark: at least one instance of "{ordering}" property.

(b) *(Bookwork)*

(i). When a program is refactored, its overall functionality is not changed, but its understandability and maintainability are improved.
Marking:
1 mark for functionality not changed.
1 mark for improvements

(ii). Software development involves repeated increments being realised in the software which can quickly destroy structure and modularity if refactoring is not regularly performed.
Marking:
1 mark for relevant aspect of Agile processes
1 mark for why this causes problems that refactoring can address

(c) *(Application of knowledge) Precondition*: `this` argument is non null and the `index` argument is in the range 0 . . . `size() - 1`.

Postcondition: Returned element is a reference to the object in the `index` position of the list referenced by `this` and the list referenced by `this` is unchanged.

Marking:
2 marks for the precondition
3 marks for the postcondition

(d) *(Application of knowledge)*

- Small team
- Experienced software engineers
- Good access to customer
- Not safety critical

This is just rote memorisation. Much better would be fishing for whys...

(i). It is clear that the software needs to be innovative so it is unlikely that it will be right first time. Through the evaluation opportunities it should be possible to get high quality feedback that can help direct the successive iterations.
Marking:
3 marks total. For full marks, looking for two good points such as those above, driven by information in the description.

(ii). XP would be better. There are no strong dependability or safety requirements that might make the extra planning and documentation activities of UP worthwhile. Several project factors (small software development team, experienced software engineers, ready access to customer) all are desirable factors for XP.
Marking:
1 mark for XP
3 marks for justification. For full 3 marks, it would be good to have both reasons why UP is not so good and why XP is appropriate. If all reasons are one or the other, award only up to 2 marks.

2. (a) *(Bookwork and application of knowledge)* The other 4 principles discussed in class are:

   - **Minimal suprise.** Avoid situations where the user might expect the system to behave one way, but it actually behaves otherwise.
   - **Recoverability.** Allow the user to recover from errors. E.g. provide an undo facility.
   - **User guidance.** Provide context-sensitive help. Ensure error messages make sense to users.
   - **User diversity.** Remember users will have different levels of expertise and might have physical impairments (e.g. colour-blindness, poor vision, hearing deficit).

   Icons could involve containers, representations of ships and the dockside, and arrows to indicate loading or unloading.

   marking:
   6 marks for 3 principles: 1 mark for each name, 1 mark for each description.
   2 marks for suggestions of how the operations could be depicted graphically.

   (b) *(Problem solving)*

   (i).

```
@Test
testAdd() {
    double arg1InFeet = 2.0;
    double arg2InMetres = 3.0;
    Distance d1 = Distance.makeDistanceInFeet(arg1InFeet);
    Distance d2 = Distance.makeDistanceInMetres(arg2InMetres);
    double actualResultInMetres = d1.add(d2).getDistanceInMetres();
    double expectedResultInMetres =
        arg1InFeet * ONE_FOOT_IN_METRES + arg2InMetres;
    assertEquals(
        expectedResultInMetres,
        actualResultInMetres,
        DELTA);
}
```

   Marking:
   1 mark: Test annotation
   4 marks: Sensible Java. Correct test
   2 marks: Code structuring that makes units clear. Good use of variable names rather than just comments.

   (ii). Computations inside the Distance class implementation will involve floating-point arithmetic and round off errors. There is no guarantee that the computations will be done in exactly the same way in the implementation code as in the test method. The 3 argument version checks the expected and actual values are within the delta value. Its value should be chosen as an upper bound on the sum of the maximum round off errors for the implementation and test calculations.

   Marking:
   1 mark for mention of what the 3 argument version is doing.

1 mark for some explanation of round-off error.
Actual answers are not expected to be as detailed as this one provided above.

(c) *(Bookwork and Application of Knowledge)*

(i). Use cases will often be easier for staff to relate to and understand. The hope then is that they will be more accurate and comprehensive.
marking:
1 mark for each of two points similar to these.

(ii). Use cases directly suggest sets of scenarios for testing the software and the whole system.
marking:
1 mark for similar argument.

(iii). In class, generic issues raised included:

- Interactions might be too detailed and could needlessly constrain design.
- Operational nature may result in less attention to software architecture and static object structure.
- May miss requirements not naturally associated with actors

For an autonomous car, while use cases might be good at capturing functional requirements at a high level, capturing a range of driving situations and events that need to be reacted to, it is not clear that they would be good for phrasing requirements on tasks such as sensor data integration and interpretation. The second and third listed issues are probably the more relevant ones to hightlight.
marking:
5 marks in total:
Answers will vary.
Award at most 3 marks if only reasonable generic issues are listed.
For full 5 marks, 2 reasonable issues must be described and their relevance must be explained.

3. (a) *(Problem solving)* A sample answer is:
Primary actor: User
Supporting actors: Bike, BankingServer (for buying extension passes)
Precondition: User has downloaded app and bought at least one pass. User is within 20m of a location with a bike.
Trigger: User signals for bike to be unlocked.
Guarantee: Selected bike is locked up at some bike dock. System has recorded that one hire pass and, if relevant, one or more extension passes, are used up.
Main Success Scenario:
1. System tells bike to unlock and to flash lights. 2. User removes bike from dock. (BUC)
3. User rides bike for up to 1 hour. (BUC)
Next two steps might be iterated 0 or more times. 4. User asks system for locations with empty docks
5. System reports locations
6. User returns bike to some dock. (BUC)
7. User signals system that hire is over.
8. System tells bike to lock itself.

Extension:
3a. User wishes to keep bike for more than 1 hour
.1 User buys one or more extension passes
.2 User rides bike for additional time
.3 Scenario continues with MSS step 4.

Answers will vary. Looking for plausible entries for each field, satisfying requests in question. BUC above indicates business use case steps.

Marking:
1 mark: Actors, including bike.
1.5 mark: Precondition. 0.5 mark for each of three.
1 mark: Trigger
1.5 mark: Guarantees. 1 mark for bike locked. 0.5 marks for pass used up.
3 marks: Main success scenario
1 mark. Extension. Looking for both enabling condition and steps.

(b) *(Bookwork)* Cohesion is a measure of the strength of the relationship between pieces of functionality within a component. Benefits of high cohesion include increased understandability, maintainability and reliability.
Marking:
2 marks for definition
2 marks for benefits – full marks if two given.

(c) *(Bookwork)*

(i). From class, advantages listed include:

- reduces dependence on a single node,
- allows people to work while disconnected,
- faster VC operations,

and the main disadvantage is that they are much more complicated and harder to understand.

Marking:
1.5 marks for advantage
1.5 marks for disadvantage.

(ii). Workflows provide teams with protocols for managing the pushes and pulls between the multiple repositories. Workflows are important for organising how a team works with a distributed VC system. They enable teams to work effectively and productively, avoiding getting confused and making errors.
Marking:
1.5 marks for definition
1.5 marks for explanation of importance.

(d) *(Application of knowledge)* Relevant other areas include:

- **Public**. Users could be frustrated or confused if the UI is buggy. Worse, given the medical application, there are risks of harm if the UI malfunctions.
- **Product**. Software engineers have a responsibility to strive to deliver high quality products. If you deliver a buggy UI, you are not doing so.
- **Judgement**. Software engineers have a responsibility to maintain integrity and independence. On this project, you have an obligation to make clear your technical concerns about the risks and therefore your possible reluctance to use the library. Whether you actually refuse to use the library somewhat depends on the level of risk. Maybe you have an obligation to be a whistleblower, disclosing the issue to other parties such as your boss's boss or the client if your boss insists on the library being used.

marking:
6 marks total.
For each of 3 areas: 0.5 marks for name, 1.5 marks for some argument as to its relevance.