Workshop seminar 1

Comp201 Software Engineering

Software Engineering Introduction

Review Questions

1. What is the major problem when dealing with large scale software systems?
   The issue is with complexity, this can be expressed in terms of lines of code, but also computational or flow complexity.
   Here is a link explaining the DES encryption standard, for example,
   https://en.wikipedia.org/wiki/Data_Encryption_Standard
   Or check out the algorithm for Huffman compression coding
   https://en.wikipedia.org/wiki/Huffman_coding
   or MD5
   https://en.wikipedia.org/wiki/MD5

   These are relatively simple algorithms but will involve a fair amount of skill to correctly Implement and test. To simplify them, breaking the problem into small sub=problems helps to deal with the complexity issue.

2. If a program has N lines of code, how many different interactions between lines of code is it possible to have?
   In theory the maximum number of interactions is N x (N-1)/2
   This is assuming interactions in one direction are the same as the other. So each interaction is uni-directional.

3. When using a structured programming language (not assembler), what feature of the language helps you reduce the number of interactions between lines of code?
   The use of functions, procedures or methods limits the different flow that are possible between different parts of the code.
   So if I have a function:

```
public int factorial(int x) {
        int answer=1;
        for (int index=1;index<=x;index++) {
              answer=answer*index;
        }
        return(answer);
}
```

   You cannot jump into the middle of the function from outside, you have to enter from the start. In assembly language this is not so, you can jump to any point in the code you like.
   In a language such as Java you can return at any point in the method. You can call another method but at least you know where you will return to (the line after the call).
   You can move about in the method using control statements such as if/then/else, while, for, case statements. Finally you can terminate the method due to an exception, can jump to a caught exception or call System.exit(1), which will terminate the program and all its threads.
   So shorter methods gives you limited complexity and makes it easier to test the code. If the method is private to the class, again its interaction is limited to other methods in the class.

4. How does the OO programming, scope and accessibility features of the Java™ language help you to reduce complexity?

   There are a number of features which reduce complexity, as follows:

   Data can be encapsulated within the class, like follows

```
public class Person {
    private String surname;
}
```

   In this case notice that ONLY code inside class can access this data, but also notice that adding a public setter, will break this encapsulation. Sometime is makes more sense to mot use setters and only set data in class constructors.

   Only public or protected methods can be called from outside a class, this reduces the scope of interaction between classes and makes testing a lot easier.

   In summary you have private, only accessible within class.

   Package private only accessible by code in classes that are in the same package as this class. To have package private, just don't put anything in front of the class name or attribute

```
class Person {
    String surname;
}
```

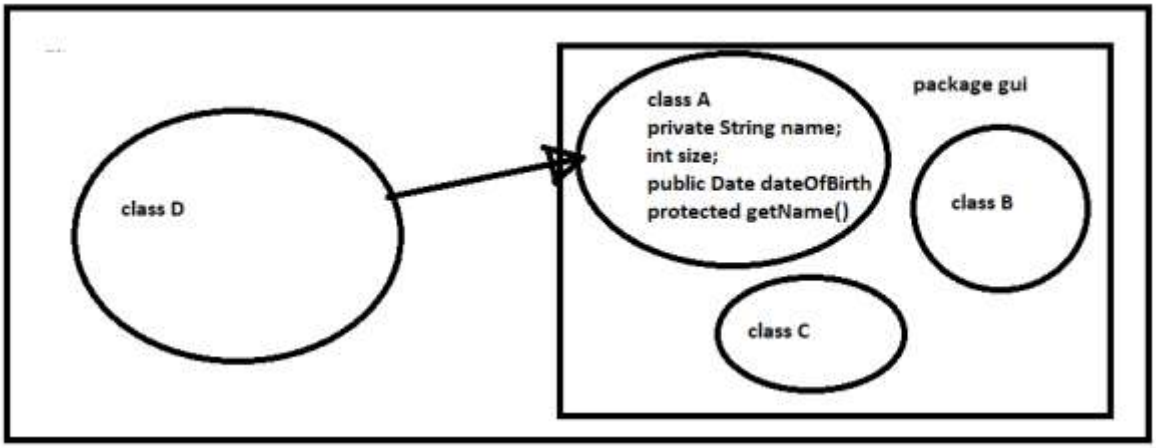
   This is useful for classes which you want to use internally inside a package but don't want to allow public access to, this keeps the code inside the package safe from being messed around with for example by extension.

   Then we have protected, this is particularly useful if you've defined a superclass that you want to allow overriding of its methods in a subclass. Perhaps the superclass is abstract and defines a method, but the sub-class will override it to provide another version.

```
public class Shape {
    protected double calculateArea() {
        return(0.0);
    }
}

public class Circle extends Shape {
     return(Math.PI*radius*radious);
}
```

   We can see we there is visually as follows:



Only class A can access name.
Classes A B and C can access size, same package, you can use one class as façade class into a package, to allow access from the outside.
Classes A, B, C (same package) and D (as a sub-class) can access get name.
All classes in the whole application can access dateOfBirth.
The more you use constrained access like private the better you control complexity.

5. What are the main features of a software process?
   Software process is defined as sets of tasks required to produce the software.
   The tasks are executed according to a defined order.
   Tasks often consume and/or produce resources, requirement engineering produces the user and system specification. The design process will consume the software specification and produce the detailed software design.
   For some software processes there is iteration in which tasks are repeated, iterative approaches are better suited to software development as the product can be grown gradually and then validated as it is produced. This also leads to better testing.
6. What is the relationship between the
7. software specification and design?
   The software specification is the basis for the design, but often the software specification shapes some of the factors that go into design. So you might make basic architectural decisions at the software specification phase. Examples of decisions which might be made at the software specification stage which could affect the design, might be: which language are we going to use, what type of database will be storing the data (document database e.g. MonoDB or relational e.g. MySQL) or what type of application will it be (mobile app or web app), are we going to use a pattern like MVC.

8. **Why is the specification important when doing testing?**
   All testing is essentially based on the specification, if the specification does not define each interaction in detail, then the testing itself will be purely objective. One of the ways of defining the specification correctly, is to make the specification a series of pass or fail tests for given scenarios. So when we define a series of use cases, we can construct tests which exercise each use case in turn including all the alternative cases and case extensions.
9. **Look up the following SCRUM terms (or guess what they might mean)**
   a. Product back log List of all remaining features that have not been completed and tested
   b. Daily SCRUM Morning 15-30 minute meeting, 3 questions, what did I complete yesterday, what do I hope to complete today, what will stop me completing what I hope to complete today
   c. SCRUM master Project manager, job is to plan and remove all obstacles to development progress
   d. Sprint One iteration of development in which team, produce and test new version of product with added features
   e. Time box Time limit for particular activity
   f. Stakeholders Individuals who are affected by production of software, can be users, developers, customers, managers, for clinic system can be doctors, nurses patients etc.
   g. Sprint burndown Shows progress of development within a sprint, so we can see if we are ahead or behind schedule.
   h. Sprint Backlog Features that are committed to sprint, but not yet completed and tested yet
   i. Sprint Planning Meeting before Sprint to determine what features to complete in next sprint, be careful not to overload Sprint.
10. There is a term “inventory loss” in software engineering, which describes features which were added to the specification and product but were not considered useful to the customer. Consider how prioritizing functional elements and developing high priority elements first will result reduced inventory loss.
    If development is done on features regardless of their importance and time starts to run out, this can lead a major problem. For example if features which were “nice to have” are developed first, but some critical core elements are left till the end the product might not work at all when we have run out of time/money. This means we may have an outcome which is total project failure.
    For this reason you should always start development with the most core critical features, examples of this could be the facility to login, to create a simple booking for a hotel system, to delete a booking. The project development plan should as early as possible develop a simple working system, features are then added to this simple system in each iteration.