CS 310 - Software Engineering
Quiz3 Solution

Question 1: Fill in the blanks. Please make sure you use the correct spellings.

1. An **interaction** ***(or you can also write use-case)*** model is a dynamic model that shows how the system interacts with its environment as it is used.

2. **Interfaces** have to be specified so that different components can be designed in parallel.

3. **Problem tracking** is an activity of configuration management where support is provided to allow users to report bugs and to allow all developers to see who is working on these problems and when they are fixed.

4. If an open-source software has a **GPL (or LGPL)** licence, you must publish any changes or modifications made to this software.

5. The **logical** view of the software architecture shows the key abstractions in the system as objects or object classes.
6. A **context** model is a structural model that demonstrates the other systems in the environment of the system being developed.
7. **Subsystem** models show logical groupings of objects into coherent subsystems.
8. If an open-source software has a **BSD** licence, you are not obliged to re-publish any changes or modifications made to this software.
9. **Implementation** is the process of converting a design into a program code.
10. **Architectural Design** is the process of identifying the major components that make up the system and their interactions.
11. A **Design/Architectural pattern** is a way of reusing abstract knowledge about a problem and its solution.
12. **Configuration Management** is the name given to the general process of managing a changing software system.

Question 2: Answer the following questions. **[9 marks]**

1. What type of software architecture should you choose if maintainability is the most important non-functional requirement for the system? **[1]**

   Use fine-grain, replaceable components that can be readily changed if needed

2. What are the **4 approaches** used to identify object classes in a system? **[2]**

Use a grammatical approach based on a natural language description of the system

Base the identification on tangible things in the application domain.

Use a behavioural approach and identify objects based on what participates in what behaviour.

Use a scenario-based analysis. The objects, attributes and methods in each scenario are identified.

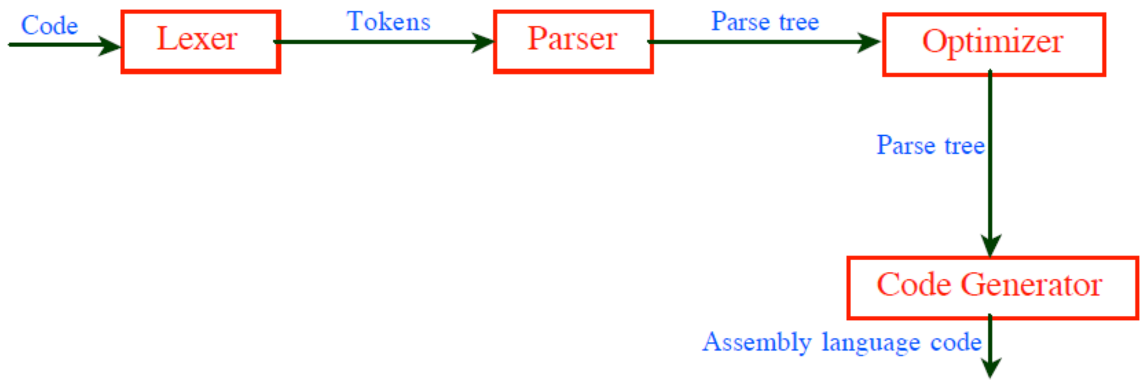
3. Consider the MVC pattern and answer the following questions. **[2]**

   When should it be used? What are its disadvantages?

   See slide 20 in lecture 10

4. Consider a software system that compiles programs written in a high-level programming language (e.g., Java) into assembly language programs. The compiler consists of four components: *Lexer*, *Parser*, *Optimizer* and *Code Generator*. The Lexer reads the input program (an ascii file) and produces a stream of word tokens which are fed into the Parser. The Parser reads in tokens and produces a stream of parse trees, one for each statement of the input program. The Optimizer processes parse trees as they become available by transforming each to another parse tree using optimizing transformations. Finally, the Code Generator takes in optimized parse trees as they become available, and produces assembly language code which is written on an output file.
   Draw the software architecture for such a software system, using one of {pipes-and-filters, repository-based, client-server, MVC}. Make sure to draw a fully labeled diagram. Explain and justify your choice. **[4]**

Use pipe-and-filter architecture because all transformations are applied in a sequence one after the other



5. What type of software architecture should you use if safety is the most important non-functional requirement for the system? **[1.5]**
   Localise safety-critical features in a small number of sub-systems

6. What is meant by reusing software at an abstraction level? Give two examples. **[1.5]**

   At this level, you don't reuse software directly but use knowledge of successful abstractions in the design of your software. Examples include Design patterns and architectural patterns

7. Which UML diagrams can be used to represent the process view of software's architecture? **[1]**

   State, Activity, Sequence and Collaboration diagrams

8. Consider the client-server architectural pattern and answer the following questions: **[3]**

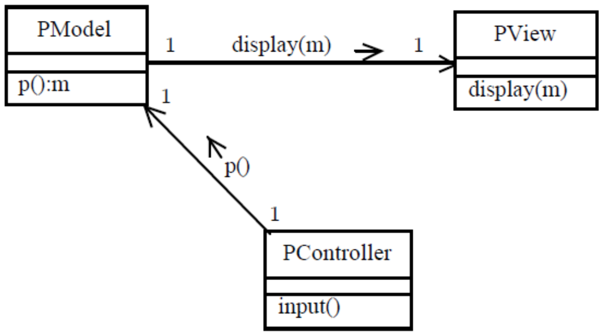
What is it? Explain 2 advantages. Explain 2 disadvantages.

See slide 34 in lecture 10

9. Mention 2 advantages and 2 disadvantages of the repository pattern.
    See slide 31 in lecture 10
10. What are the two ways in which an architectural model of a system may be used? **[1]**
    As a way of facilitating discussion about the system design

    As a way of documenting an architecture that has been designed

11. Consider a software component that takes a point-and-click **input**, calls an operation **p()** and **displays** the message generated by the operation. Define a Model-View-Controller architecture for this component using a class diagram.
    This is just a model answer. Several variations from students were also accepted.



12. Your company has decided to develop a new word processor called BeyondWords (BW for short). In its desktop version BW will compete directly with Microsoft Word and similar products from other companies. BW is also intended for use on engineering workstations (running Solaris or Linux), and versions of BW will be required for PalmPilots and Compaq IPAQ's. You have been named project manager. Your responsibility is to establish the detailed process by which BW will be created and lead the development team. Naturally your management wants BW on the market as soon as possible and they have a limited development budget. What software architecture would you use for this application? Give two reasons for your selection.

    Since the software needs to be compatible with multiple platforms (Solaris, Linux, Windows, PalmPilot, IPAQ's etc.), this requirement suggests that we use a layered architecture so that we can design the upper layers (User Interface, Application Logic etc.) without being concerned about the platform while the lower layer that directly deals with the platform can be changed for each different platform without having to make changes to the higher layers.

    Secondly, the requirement states that the product should be released in the market as soon as possible. Again, it suggests using the layered architecture because the layered pattern supports incremental delivery. You can develop the core product with the most urgently required facilities first, deliver that version and keep working on the added functionality for the later versions.