COMP 3111: Software Engineering

Review 1

Q1. Having quality design goals is of most help in reducing the complexity of:

1. designing the system.
2. building the system.
3. maintaining the system.
4. cost and time estimates for developing the system.
5. understanding the system.

Q2. An interface *abstracts* a module. Abstraction helps most in reducing the complexity of:

1. designing the system.
2. building the system.
3. maintaining the system.
4. cost and time estimates for developing the system.
5. understanding the system.

Q3. Which of the following is not an issue when considering "programming-in-thelarge"?

1. Understanding user requirements
2. Applying appropriate software development processes
3. Building models of a system
4. Validating user input
5. Making design trade-offs

Q4. Which statement is not true about software engineering?

1. It requires a team effort.
2. It should build a quality system.
3. It should solve a real user problem.
4. It is an ad-hoc development effort.
5. It deals with multiple versions of the software.

Q5. Which statement about the UML (Unified Modeling Language) is true?

1. The UML makes us think about the world in a certain way.
2. The UML can be used to model only software systems.
3. The UML can be used for only object-oriented software systems.
4. The UML is a software development process.
5. The UML provides a fixed set of modeling elements.

Q6. Which of the following UML concepts is not a classifier?

1. class
2. operation
3. association
4. method
5. attribute

Q7. In software engineering, we build models of a software system to:

1. reduce the workload of the project team.
2. help us deal with the complexity of a problem.
3. reduce the amount of communication with users.
4. know which people to hire into the project team.
5. know which language to use for implementation.

Q8. Which of the following is not a property of an attribute in the UML?

1. data type
2. multiplicity
3. visibility
4. signature
5. changeability

Q9. Is it possible for there to be more than one association between any two classes?

1. Yes, always.
2. Yes, but only for binary and higher order associations. 3. Yes, if the multiplicity is many to many (N:M)
4. No, no way.
5. Gee, I don't know!

Q10. In the UML the multiplicity of an association specifies

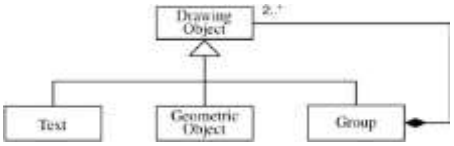
1. which classes can be related to each other.
2. how many classes participate in the association.
3. the navigability of the association.
4. the number of objects that must/can be related.
5. whether role names are required.

Q11. Considering what is true **in the real world**, what is the most likely multiplicity of the IssuedTo association?

1. CreditCard (1..*) -----IssuedTo----- (1..1) Person
2. CreditCard (0..*) -----IssuedTo----- (1..1) Person
3. CreditCard (1..1)-----IssuedTo----- (0..*) Person
4. CreditCard (1..1)-----IssuedTo----- (1..*) Person
5. CreditCard (0..*) -----IssuedTo----- (1..*) Person

Q12. A drawing object can be either text, a geometric object or a group of text and geometric objects. Which kind or kind(s) of relationship(s) are needed to correctly model this situation?

1. association only
2. generalization only
3. composition and generalization only
4. association and generalization only
5. all of association, composition and generalization



Q13. When is an association class needed?

1. When the multiplicity of the relationship is many-to-many (N:M).
2. When the relationship is unidirectional.
3. When the relationship has properties.
4. When the relationship is mandatory.
5. When the relationship is optional.

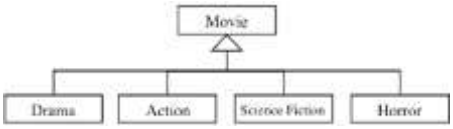
Q14. In the UML, the concept of generalization

1. relates two different classes by an association relationship.
2. allows a class to remove attributes and operations from its subclasses.
3. supports the concept of stereotype.
4. allows a class to specialize its attributes and operations.
5. links instances of different classes together.

Q15. Which of the following is a generalization coverage constraint?

1. complex
2. disunion
3. overloaded
4. common
5. disjoint

Q16. Considering what is true **in the real world**, the generalization shown in the figure is:



1. overlapping and complete.
2. disjoint and complete.
3. overlapping and incomplete.
4. disjoint and incomplete.
5. none of the above.

Q17. Which of the following statements is <u>true</u> about ***custom*** software?

1. The number of copies in use is high.
2. The requirements come from market research.
3. The development effort is high.
4. The requirements come from hardware needs.
5. The development effort is low.

Q18. A milestone in a software development project is

1. a management decision point.
2. a problem that delays the project.
3. the time at which a project starts.
4. the time at which a project completes.
5. a meeting with the client.

Q19. When developing a software development plan, the first task to do is to:

1. decide on the implementation environment.
2. define the scope of the project.
3. identify the deliverables.
4. develop a schedule for the project.
5. identify the project risks.

Q20. Which of the following is not a way to deal with risks in software development?

1. avoid (replan or change requirements)
2. ignore (act as if it won't happen)
3. mitigate (devise tests to see if it occurs)
4. confine (restrict the scope of its effect)
5. monitor (constantly be on the lookout for it)

Q21. Which of the following is not an emphasis of an Agile development process?

1. individuals and interactions
2. comprehensive test plan
3. client involvement/collaboration
4. working software
5. responsiveness to change

Q22. In domain modeling we capture the system's most important

1. user interfaces.
2. functional requirements.
3. use cases and scenarios.
4. classes and associations.
5. acceptance tests.

Q23. What do we capture in use-case modeling?

1. user interface requirements
2. hardware requirements
3. acceptance tests
4. system behaviour
5. data requirements

Q24. "The system should register a student in less than a second" is an example of what type of requirement?

1. functional
2. pseudo
3. data
4. nonfunctional
5. user interface