Qualidade do Software

Mestrado Integrado em Engª Informática – Ano Lectivo de 2019/2020, 1º semestre

Duration: 1h30

Question 1. Which of the following statements about *reengineering* (of a software system) is true?
[A] Reengineering is the same as reimplementing the software system from scratch.
[B] Reengineering generally includes some form of reverse engineering followed by some form of forward engineering or restructuring.
[C] Reengineering is another name for reverse engineering the software system.
[D] Reengineering is another name for forward engineering the software system.

Question 2. Which of the following statements about *legacy system* is true?
[A] A legacy system is a software system developed a long time ago.
[B] A legacy system is a system offered by another organization.
[C] A legacy system is a software system developed using an obsolete programming language.
[D] A legacy system is valuable to the organization using it.

Question 3. Which of the following statements about *reverse engineering* is true?
[A] Reverse engineering involves changing the subject system.
[B] Reverse engineering involves replicating the subject system.
[C] Reverse engineering involves examining the subject system.
[D] Reverse engineering is the traditional process of moving from high-level abstractions to the physical implementation of a system.

Question 4. Reverse engineering involves the creation of alternative representations of the subject system. Which of the following sentences is correct?

[A] The alternative representations usually reside at the same level of abstraction as the system studied.
[B] The alternative representations usually reside at a lower level of abstraction than the system studied.
[C] The alternative representations usually reside at a higher level of abstraction than the system studied.
[D] The alternative representations usually serve as a blueprint to (re)build the system anew.

Question 5. One thing covered in the lectures was characterized as serving to "*understand applications, to get an overview of a large system and identify potential design problems.*". Which of the options below fully serves to this purpose?

[A] Design patterns
[B] Code Smells
[C] Software Metrics
[D] Design Disharmonies

Question 6. One of the key aspects of software *maintenance* is that:

[A] It is carried out after delivery.
[B] The software system is deteriorating.
[C] It is indistinguishable from traditional software development.
[D] It assumes that all possible requirements are known before it begins.

Question 7. A *metric* is the mapping of a particular characteristic of a measured entity to:
[A] a graphical artefact.
[B] a area with a given colour.
[C] a numerical value.
[D] some shape or an arrow.

Question 8. One of the metrics covered in the lectures and discussed in the book by Lanza et al., is CYCLO – the *Cyclomatic Number*. Which of the following statements is true?
[A] CYCLO is a direct metric, just like *Weighted Method Count* or *Average Hierarchy Height*.
[B] CYCLO is a direct metric, just like *Number of Packages*.
[C] CYCLO is a direct metric and as such, it is ideal to represent proportions.
[D] CYCLO is not a direct metric.

Question 9. According to the notion of *software visualization* given in this course, those visually accessing metrics information by means of some visualization approach, should expect NOT to see:
[A] rectangles and arrows
[B] videos
[C] graphs
[D] numbers

Question 10. Examples of *software visualization* used in this course include:
[A] The *Polymetric* and *Treemap* views of SourceMiner
[B] The *Grid* view of SourceMiner.
[C] The *Overview Pyramid* of inFusion.
[D] The Design Disharmonies reported by inFusion.

Question 11. What is the reason why colours such as green, red or blue are usually not used to represent quantities, with levels of gray being preferred instead?
[A] Colours proved to be excessively distracting in practice.
[B] Colours reflect more than one dimension simultaneously.
[C] Humans do not derive a precise ordering between colours so as to reflect relative magnitudes.
[D] Only light colours are suitable for the background of areas and rectangles.

Question 12. The *Treemap* view of SourceMiner is suitable to:
[A] view and analyse three or more metrics simultaneously.
[B] analyse hierarchically organized data.
[C] order the parts of the analysed system in ascending or descending order of a given metric.
[D] represent the structure of a graph.

Question 13. If the *representation condition* is applied to a metric M, what does it state?
[A] It states that if one software entity is less than another entity in terms of a selected attribute, then M must associate a smaller number to the first entity than it does to the second entity.
[B] It states that M must define a maximum and minimum values.
[C] It states that M must be defined in such a way, as to enbale comparisons between systems of different sizes.
[D] It states that M must be a direct metric.

Question 14. What is *software refactoring*?

[A] The process of improving the structure of the software code, possibly affecting its behaviour.

[B] The process of changing the behaviour of the software code.

[C] The process of improving the structure of the software code without affecting its behaviour.

[D] The process of improving the runtime performance of the software code by without changing its behaviour.

Question 15. What *software refactoring* is NOT?

[A] Adding new functionality.

[B] Rewriting from scratch.

[C] Optimizing.

[D] All the above.

Question 16. Consider the following code fragments involving two classes – `Phone` and `Customer`:

```
public class Phone {
   ...
   public String getAreaCode() { ... }
   public String getPrefix() { ... }
   public String getNumber() { ... }


public class Customer ...
   private Phone mobilePhone;

   public String getMobilePhoneNumber() {
       return "(" +
          mobilePhone.getAreaCode() + ") " +
          mobilePhone.getPrefix() + "-" +
          mobilePhone.getNumber();
   } ...
```

Which code smell can be detected in this fragment?

[A] *Alternative Classes with Different Interfaces*

[B] *Feature Envy*

[C] *Data Clumps*

[D] *Message Chains*

Question 17. A given class was identified as being a case of *God Class*. In what circustances is that class not likely to cause problems to the evolution of the system?

[A] When the class is covered with tests.

[B] When only one instance of the class is created by the program.

[C] When the class is duplicated into several clones.

[D] When it resides in a stable part of the system that is not likely to change.

Question 18. Which group of indicators is suitable to detect cases of *Feature Envy*?

[A] The conjunction of a high value for *Weighted Method Count* (WMC) with a low value for *Tight Class Cohesion* (TCC).

[B] The sum of the complexity of class methods has a high value.

[C] The class's method count is low and count of fields is high.

[D] Number of methods used from a few other classes is higher than the number of methods used from the class itself.

Question 19. Among other things, cases of *Data Class* may be indicators of
[A] Occurrence of instances of *Brain Method* in the system.
[B] Excessive method complexity.
[C] Excessive coupling between classes.
[D] Duplication of the logic that uses data from the *Data Class*.

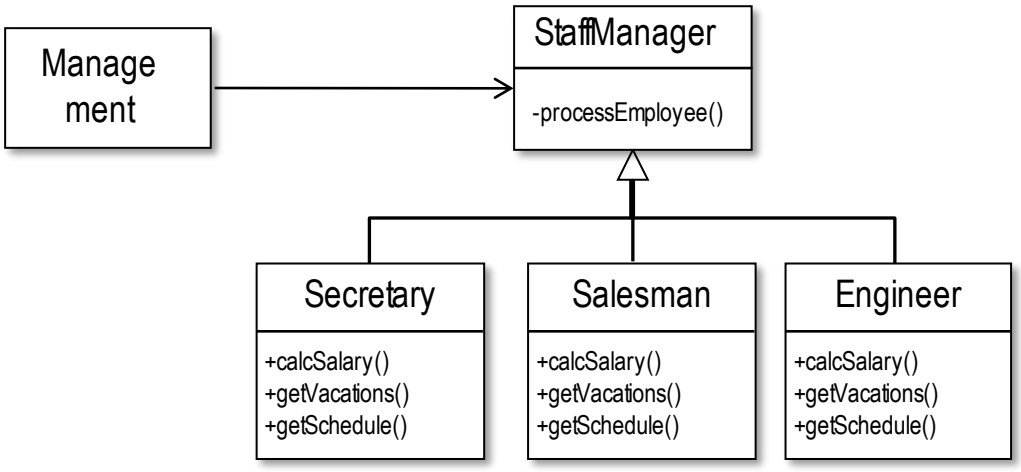
Question 20. In *Clone Detection* tasks, one of the cares taken is to:
[A] Detect very fine-grained chunks of code.
[B] Detect violations of encapsulation between several classes.
[C] Detect chunks of code that are scattered across many different classes.
[D] Aggregate code chunks near to each other, considering them a single chunk of code.

Question 21. Consider the following line of code. The type of each called method is a class from the system, not a library class. What is the potential problem with this code?

```
invoice.getOrder().getCostumer().getOrderHistory()
```

[A] Code gives rise to dependencies of several other classes and to the navigation structure.
[B] Excessively complex code that is difficult to understand.
[C] Lines of code of this kind tend to be excessively long, going beyond 80 characters.
[D] This kind of code is unsuitable to be used by subclasses.

Question 22. The diagram below shows a class structure in which `Managent` uses `StaffManager` to process data on employees. The processing includes a private operation comprising several steps, e.g., calculating the salary, computing vacation entitlements and work schedule, etc. Each step is implemented in a way specific to a given kind of employee. What pattern is used in this example?



[A] *Template Method*
[B] *Observer*
[C] *Null Object*
[D] *Composite*

Question 23. Sometimes, *God Classes* have a private constructor. What pattern is related to that trick?

[A] *Singleton*

[B] *Template Method*

[C] *Façade*

[D] *Composite*

Question 24. The design patterns covered in the lecture and sessions aim to achieve:

[A] Separation of concerns

[B] Better performance

[C] Lower memory footprint

[D] Simpler code

Question 25. The *hooks* used in most pattern examples given in the lecture and sessions consist of:

[A] piece of code changed by a pre-processor such as that in C.

[B] an abstract method (in a superclass or possibly in an interface) and several concrete implementations, each provided in a different (sub)class.

[C] a method that receives an argument and which returns an instance of the concrete class specified through that argument.

[D] Java interfaces of abstract classes.