# Qualidade do Software

**Mestrado Integrado em Eng$^{a}$ Informática – Ano Lectivo de 2019/2020, 1$^{o}$ semestre**

**duration: 80 minutes**

**Question 1.**
Over time, changes need to be made on a software system, which is likely to increase its complexity. Which of the options below contributes to controlling that complexity?
[A] Re-engineering
[B] Replacement
[C] Re-documentation
[D] Measurement

**Question 2.**
What is the study of a software system, that sometimes yields a wider range of information than the system itself can provide?
[A] Re-engineering
[B] Measurement
[C] Design recovery
[D] Covering the system with unit tests

**Question 3.**
One category of activities covered in lectures is carried out after having been delivered to at least one client. What it that category of activities?
[A] Reverse Engineering
[B] Re-documentation
[C] Restructuring
[D] Maintenance

**Question 4.**
A Legacy system is a system of value to its owner that
[A] is implemented in an obsolete technological platform
[B] lacks up-to-date documentation
[C] significantly resists modifications and changes
[D] requires modifications

**Question 5.**
What is the desirable combination of coupling and cohesion in a software system?
[A] High cohesion and high coupling
[B] High cohesion and low coupling
[C] Low cohesion and high coupling
[D] Low cohesion and low coupling

**Question 6.**
Consider the following chain of operation calls: `obj.op1().op2().op3().op4()`
In what circumstances are such chains undesirable from the point of view the overall level of connectivity of a system?
[A] When the return type of the various operations are all different developer classes
[B] When the return type of the various operations originate from the language's standard APIs
[C] When the return type of the various operations are all the same type
[D] When the original calling object (obj) is an instance of small and simple class

**Question 7.**
One example from the lectures uses an image of a few solid cylinders and uses height, diameter and weight as quantities to be interpreted. What is the problem with one the use of the quantities used?

[A] The height quantity gets confused with the weight quantity

[B] The levels of gray quantity gets confused with the weight quantity

[C] The human cognitive system does not map relations between different values of weight, in the way it does with the other quantities

[D] The number of different quantities being mapped in a single entity is excessive

**Question 8.**
Which of the following statements about the *Cyclomatic Number* is true?

[A] It is a size metric

[B] It is a inheritance metric

[C] It is a process metric

[D] It is a direct metric

**Question 9.**
When is a very high complexity value for a method not a cause of worry?

[A] When the method is a constructor

[B] When the method is an instance of the *Brain Method* symptom

[C] When the method does not call any other methods

[D] When the code of the method is very regular and looks like a table

**Question 10.**
Does the *Overview Pyramid* comprise an example of software visualization?

[A] Yes, because the *Overview Pyramid* enables the compact visualization of severalmetrics

[B] Yes, because the *Overview Pyramid* enables the visualization of multiple metrics

[C] No, because the *Overview Pyramid* uses colours instead of levels of gray

[D] No, because the *Overview Pyramid* shows the metrics as numerical values

**Question 11.**
A ratio of metrics studied is FANOUT/CALLS, i.e., the sum of the FANOUT metric (*efferent coupling*) for all user-defined operations, divided by the total number of distinct operation calls (invocations) in the project. What aspect of the system does this ratio provide information about?

[A] The overall characterization of the usage of inheritance in the system

[B] The intrinsic functional complexity of the system

[C] The intensity of collaboration between the system's operations

[D] How much the coupling involves many different modules

**Question 12.**
When is a *God Class* not a cause of worry?

[A] When it resides in a stable part of the system

[B] When it stops attracting new functionalities during the long term evolution of the system

[C] When there are even worse instances of *God Class* in the system

[D] When the *God Class* is covered by unit tests

**Question 13.**
What is the primary purpose of a *software visualization*?

[A] To identify design patterns in source code

[B] To assist in the understanding of the software system

[C] To derive UML diagrams from the source code

[D] To detect instances of duplicated code

**Question 14.**
The *SourceMiner* tool provides several views of specific aspects of the system under analysis. However, one of the Views does not match the definition of software visualization given in classes (and in the book by Lanza et al), due to the nature of the information displayed. Which one?

[A] Grid

[B] Treemap

[C] Graph

[D] Polymetric

**Question 15.**
In a system that contains a significant number of instances of *Data Class*, what other undesirable symptom it is reasonable to expect to be detected?

[A] Duplication of functionalities concerning the class, located outside the class

[B] Several classes whose functionality is not required in the system

[C] Very heavy use of class inheritance

[D] Packages with too many classes

**Question 16.**
What is the symptom that occasionally appears associated with the existence, in the class, of a single constructor with private visibility?

[A] *God Class*

[B] *Null Object*

[C] *Brain Class*

[D] *Data Class*

**Question 17.**
Perfective maintenance is:

[A] the reactive modification of a software product performed after delivery to correct discovered problems or faults

[B] any modification of a software product after delivery to improve performance or maintainability

[C] the modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment

[D] modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults

**Question 18.**
The lectures covered “recipes” about the creating a layer between two different groups of classes, so that one group can be changed without impact on the other group. What are those “recipes”?

[A] hooks

[B] design patterns

[C] refactorings

[D] idioms

**Question 19.**
What is the relationship between refactoring and code smells?
[A] Refactoring aims to remove bad smells from the system
[B] Refactoring aims to further develop the smells in the places where they are detected
[C] Code smells and refactoring can both be used to derive information from a software system
[D] Code smells and refactoring are intended to document bad symptoms in a software system

**Question 20.**
Why is system optimization not considered refactoring?
[A] Because optimization tasks change the observable behavior of the system
[B] Because optimization tasks often cut across the system's module boundaries
[C] Because optimization tasks require the prior coverage of the system with unit tests
[D] Because optimization tasks involve re-developing parts of the system

**Question 21.**
Which of the following groups of refactorings is appropriate for dealing with the 'Feature Envy' symptom?
[A] *Pull Up Field, Pull Up Method*
[B] *Push Down Field, Push Down Method*
[C] *Move Field, Move Method, Extract Method*
[D] *Extract Class*

**Question 22.**
Which kinds of metrics can be used to detect code clones?
[A] Size and complexity metrics
[B] Inheritance metrics
[C] Coupling metrics
[D] None of the above

**Question 23.**
When analysing a software system, you find a section of heavily commented code. What should you try to find out about that comment text, that it will help you to decide whether there is indeed something wrong close to the commented code?
[A] See if the comments cite some non-obvious algorithm or solution that someonemay not perceive when looking at the code
[B] See if it explains the programmer's doubts, pointing aspects of the problem about which he/she isn't sure
[C] See if it explains what the code does
[D] See if the comments tell why something is done a particular way, or why it wasn't

**Question 24.**
The description of a design pattern in a specific language is called

[A] *hook*

[B] *template*

[C] *clone*

[D] *idiom*

**Question 25.**
What is the pattern that documents the use of inheritance to eliminate duplication in some specific situations?

[A] *Null Object*

[B] *Strategy*

[C] *Singleton*

[D] *Template Method*