Instructions

Please, read these instructions carefully. Failure to comply with *any* of the following instructions means *invalidation of your exam*:

1. This exam consists of 32 multiple-choice questions. For each question one answer is correct.
2. Not answering a question is considered an incorrect answer.
3. In principle, given P correct answers in one set of questions, the grade for that part G will be determined by the formula: $G = max(1, 10 - (32 - P)/2)$
4. Fill in your answers on the provided separate answer sheet and hand in the answer sheet at the end of the exam. The set of sheets with the questions (the one you are reading) is for you to bring home.
5. The use of books, papers, computers, or other material is not permitted.

Page left empty on purpose.

Part I
Software Security

1. Defensive programming is:
   A. always to validate assumptions on inputs
   B. has been invented by Phil Zimmermann
   C. not needed for strongly typed languages
   D. useless because hackers will get in anyway
2. The problem that is least likely to be solvable with defensive programming is:
   A. buffer overflow
   B. integer overflow
   C. jailbreaking
   D. stack overflow
3. The OWASP top 10 is:
   A. a list of common issues with web sites
   B. a web site owned by Microsoft
   C. a list of issues that do not concern Java programmers
   D. a conspiracy by the NSA to be able to access our data
4. Exceptions are:
   A. a structuring technique for defensive programming
   B. equivalent to conditionals
   C. available in all programming languages
   D. a technique for program obfuscation
5. Responsible disclosure is:
   A. a bounty program by the NSA
   B. a technique for defensive programming
   C. a way to make money
   D. an opportunity for those responsible for issues to fix them in time
6. Defensive programming in Java is
   A. not needed because Java is a strongly typed language
   B. harder than defensive programming in C because you have more control in C
   C. very inefficient because exceptions are so inefficient
   D. Particularly useful if you use libraries that someone else has provided

Part II

Design Patterns

7. Fill in the blank, with the correct design pattern:
The _____ pattern limits the numbers of instances that can be created for a class. These instances are globally accessible.

   A. state B. adapter C. iterator D. none of the others

8. Fill in the blank, with the correct design pattern:
The _____ pattern isolates a data collection from the means that's used to store the collection.

   A. state B. adapter C. iterator D. none of the others

9. Fill in the blank, with the correct design pattern:
The _____ pattern allows an object to alter its behavior when its internal condition changes. The object will appear to change its class.

   A. state B. adapter C. iterator D. none of the others

10. Fill in the blank, with the correct design pattern:
The _____ pattern decouples operations from the object that actually performs the operation.

    A. state B. adapter C. iterator D. none of the others

11. What is a *limitation* of the *Singleton* pattern when implemented in a programming language with a static type system (such as Java)?

    A. the Singleton object is created and destroyed only once

    B. the global namespace is cluttered with hard-to-find objects

    C. at most one instance of a Singleton class can be created

    D. none of the above

12. When implementing the *Adapter* pattern in a language with a dynamic type system (such as Python),

    A. the client needs to know the class of the Adaptee

    B. the Adapter class does not need to reference to the Adaptee class

    C. the adapter does not need to implement the target interface

    D. none of the above

13. What is a *limitation* of the *Adapter* pattern when implemented in a programming language with a static type system (such as Java)?

    A. Adapters are passive, passing messages to single Adaptees

    B. an Adapter makes it hard to add classes without changing the Adaptee class

    C. the Adapter makes the Adaptee appear to support the Target interface

    D. none of the above

14. The *Command* pattern is suited for implementing the 'undo' and 'redo' operations. This is because the `Command` object can encapsulate:

    A. a method for doing a specific task for the invoker

    B. the state information necessary to perform these operations

    C. how to adapt to the changes in the caller

    D. all of the above

15. When implementing the *Singleton* pattern, it is necessary to:
    A. store the instance as a private static variable
    B. providing a static method returning a reference to the instance
    C. declaring all constructors of the class to be private
    D. all of the above

16. When implementing the *Singleton* pattern, it is important to consider:
    A. concurrency problems
    B. the number of invokers
    C. the last invoker
    D. none of the above

17. With the *Iterator* pattern it is possible to:
    A. traverse all the objects in a collection
    B. obtain an object in a collection by its index
    C. obtain the last object in a collection, directly
    D. obtain the second object in a collection, directly

18. As discussed in class, it is possible to create a programming language without `true` and `false` primitives: A good solution would consist in designing a language that defines an abstract class `Boolean` with two concrete subclasses `True` and `False`. In this design, where should the method '`public Boolean and(Boolean aBoolean)`' be implemented?
    A. in the class `Boolean`
    B. in the classes `True` and `False`
    C. in an external class `Operations`
    D. none of the above

19. Consider the design for non-primitive booleans explained in question 18. The implementation of the method '`public Boolean and(Boolean aBoolean)`', as explained in class, uses a technique called:
    A. nominal subtyping
    B. operator overloading
    C. double dispatch
    D. ad hoc polymorphism

20. Consider the design for non-primitive booleans explained in question 18. What would be a reasonable design pattern to use for the classes `True` and `False`, as also discussed in class?
    A. singleton
    B. command
    C. state
    D. none of the above

Part III

Software Metrics

21. A metric represents:
    A. Quantitative indication of the amount of some attributes of a product or process.
    B. A quantitative measure of the degree to which a system, component, or process has a certain attribute.
    C. The act of determining a measure.
    D. None of the above.

22. Fill in the blank: Traceability links can can be used as a metric for _____.
    A. customer satisfaction.
    B. estimating the complexity of source code.
    C. requirement implementation.
    D. estimating the error rate of developers.

23. Which of the following sentences about 'size metrics' is *true*?
    A. They can determine the number of white box test cases to be developed.
    B. They are language and programmer dependent.
    C. They cannot be used to identify the presence of bad code.
    D. None of the above.

24. Fill in the blank: Coupling metrics measure _____.
    A. how good is the modularity of the system.
    B. whether a class implements a single responsibility.
    C. the developers productivity.
    D. the independent linear paths of a software method.

25. Which kind of metrics are more suitable to detect Blob classes?
    A. Only size metrics.
    B. Complexity metrics.
    C. Size, cohesion, and coupling metrics.
    D. None of the above.

26. Which of the following sentences about thresholds selection for bad code identification is *true*?
    A. They should be defined depending on the project and/or the context.
    B. They can be defined once and used for any new project.
    C. They must be specified only for cohesion metrics.
    D. None of the above.

Part IV

Software Architectures

27. Fill in the blank: The architecture of a software system _____.
    A. defines the requirements of the system.
    B. defines only the interactions between production classes and the corresponding test code.
    C. defines the system in terms of computational components and interactions among those components.
    D. does not define the interactions among the computational components of the system.

28. Which of the following sentences about an architectural style is *false*?
    A. It collects a set of architectural decisions that are applicable in a given development context.
    B. It collects a set of architectural decisions that constrain architectural design decisions that are specific to a particular system within that context.
    C. It collects a set of architectural decisions that elicit beneficial qualities in each resulting system.
    D. None of the above.

29. Fill in the blank: A layered architecture _____.
    A. is universally applicable.
    B. increases the abstraction level incrementally.
    C. does not ensure evolvability and reuse.
    D. ensures high performance.

30. Which of the following sentences about the service-oriented architecture (SOA) is *true*?
    A. Services are assembled dynamically, incorporating new capabilities on-the-fly for greater adaptability.
    B. They are not designed for integration into enterprise-wide environment.
    C. The way services are assembled is dependent from the way they are implemented.
    D. None of the above.

31. Fill in the blank: The primary property to evaluate and maintain in cloud architectures is _____.
    A. usability.
    B. reliability.
    C. software quality.
    D. scalability.

32. Which of the following software architecture strategies explicitly aims at reducing the time between committing a change to a system and the change being placed into normal production, while ensuring high quality?
    A. Client-Server.
    B. A layered one.
    C. DevOps.
    D. None of the above.