Final 1 Sample 2020

Due Dec 16 at 11:59pm **Points** 17 **Questions** 17
Time Limit 25 Minutes

Instructions

The questions on this quiz are based mainly on:

- Module 7 (especially chapter 9 and chapter 11 slides)
- Module 10 (especially the slides)
- Module 11 (especially the slides and recordings)

There may also be a couple questions from the first half of the course.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	12 minutes	13 out of 17

Submitted Dec 11 at 8:10pm

Question 1

0 / 1 pts

What Java mechanism or feature serves as an abstraction of a Java component by describing its behavior?

orrect Answer ○ the API of a class

○ the class itself

ou Answered ● the interface of a class

○ the superclass of a class

Question 2

1 / 1 pts

What should software developers strive for regarding coupling and cohesion of components?

○ high coupling and high cohesion

○ high coupling and low cohesion

Correct! ● low coupling and high cohesion

○ low coupling and low cohesion

Question 3

1 / 1 pts

What is one of the most important things to look for when refactoring?

○ Faulty implementations

○ Irrelevant comments

○ Poor performance

Correct! ● Redundant code

Question 4

1 / 1 pts

Which of the following is not one of the four principles used to guide component-level design?

○ Dependency Inversion Principle

○ Interface Segregation Principle

○ Open-Closed Principle

Correct!

● Parsimonious Complexity Principle

Question 5

0 / 1 pts

Consider a "get" method in a collection that returns the object at the specified index and throws an exception if the index is out of bounds. According to strict design-by-contract, who is responsible for checking that the index is in bounds?`

○ Client caller

orrect Answer

○ Class implementer

ou Answered

● Both the client and the implementer

○ Neither the client nor the implementer

Question 6

1 / 1 pts

Occurs when a higher layer in a system can access a lower layer, but not the other way around

○ Common coupling

○ Content coupling

○ Functional cohesion

Correct!

● Layer cohesion

Question 7

1 / 1 pts

With thorough testing it is possible to remove all defects from a program prior to delivery to the customer.

○ True

Correct!

● False

Question 8

1 / 1 pts

In software quality assurance work there is no difference between software verification and software validation.

○ True

Correct!

● False

Question 9

1 / 1 pts

Which of the following is NOT a kind of system testing?

Correct!

● Acceptance testing

○ Beta testing

○ Recovery testing

○ Stress testing

Question 10

1 / 1 pts

Stress testing examines the pressures placed on the user during system use in extreme environments such as temperature and humidity.

○ True

Correct!

● False

Question 11

1 / 1 pts

In white-box testing, a reasonable goal is to ensure that

○ each input is executed at least once

○ each method is executed at least once

○ each path is executed at least once

Correct!

● each statement and each branch is executed at least once

Question 12

1 / 1 pts

If there were some way to make exhaustive testing practical, we could be certain that our code was correct.

○ True

Correct!

● False

Question 13

1 / 1 pts

What is the cyclomatic complexity of the following code?

```
if (condition) {
    x++;
} else {
    x--;
}
```

○ 0

○ 1

Correct!

● 2

○ 3

Question 14

1 / 1 pts

The cyclomatic complexity metric provides the designer with information regarding the number of

○ cycles in the program

○ errors in the program

Correct!

● independent logic paths in the program

○ statements in the program

Question 15

0 / 1 pts

A Java API is a good example of an informal specification.

orrect Answer

○ True

ou Answered

● False

Question 16

0 / 1 pts

One challenge to developing a verifying compiler is that developers who use it will have to write a lot more test cases.

ou Answered

● True

orrect Answer

○ False

Question 17

1 / 1 pts

Which of the following is NOT a principle of agile development?

○ Build projects around motivated individuals

Correct!

● Build software that exhibits effective modularity

○ Business people and developers must work together daily throughout the project

○ Welcome changing requirements, even late in development