edX

Quiz: High Level Design
Question 1

1/1 point (graded)
Which of the following challenges is more likely to arise with a Bottom Up Composition approach to Abstraction, and is less likely to arise with a Top Down Decomposition approach?

- ○ making high level design decisions first can cause low level inconsistencies later on

- ◉ making low level implementation decisions first can cause high level inconsistencies later on ✔

- ○ the same challenges will be faced no matter what approach is taken

| Submit | You have used 1 of 1 attempt |

✔ Correct (1/1 point)

Question 2

1/1 point (graded)
What is the main difference between Information Hiding and Encapsulation?

- ○ information hiding: only applies to interfaces; encapsulation: only applies to concrete classes

○ information hiding: only applies to concrete classes; encapsulation: only applies to interfaces

○ information hiding: separating implementation from specification; encapsulation: separating what varies from what stays the same

◉ information hiding: separating what varies from what stays the same; encapsulation: separating implementation from specification ✔

Submit	You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 3

1/1 point (graded)
A high quality Technical Representation should:
Select all that apply.

☐ be ambiguous

☑ be unambiguous

☑ be accurate

☐ be imprecise

☐ provide equal value for all stakeholders

☑ facilitate concrete technical discussions between stakeholders

☐ be the only specification for a system

✔

Submit   You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 4

1/1 point (graded)
Which of the following techniques help contribute to better API usability?
Select all that apply.

☐ exposing internal details of your API

☐ having a large API that does many things

☑ having well-named methods in your API

☐ updating method names in your API frequently

☑ providing examples of how to use methods in your API

✔

Submit   You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 5

1/1 point (graded)
Which of the following are considered high level architectural constraints of REST based
systems?
Select all that apply.

☑ clients accessing a REST service must use the URI naming scheme

☐ clients accessing a REST service are exposed to internal implementation details

☑ REST services communicate using intermediate representations

☑ REST services are based on self-descriptive hypermedia documents

☐ REST services only include idempotent actions (i.e. GET, PUT, POST, DELETE)

✔

| Submit | You have used 1 of 1 attempt |

✔ Correct (1/1 point)

Question 6

1/1 point (graded)
Which of the following is NOT true about Coupling?

○ it negatively impacts the maintanability and evolvability of programs

◉ it minimizes interfaces between program elements ✔

○ it causes errors to propogate further when changes are made

○ it makes code harder to reuse

○ it makes code harder to understand

| Submit | You have used 1 of 1 attempt |

✔   Correct (1/1 point)

Question 7

1/1 point (graded)
Which of the following is NOT true about Cohesion?

○   it measures how well program elements belong together

○   it encourages a larger number of smaller classes

○   it helps to isolate changes

○   it impedes software maintenance and evolution due to a larger number of classes
    ✔

Submit   You have used 1 of 1 attempt

✔   Correct (1/1 point)

Question 8

1/1 point (graded)
Why is copying and pasting code problematic?
Please select all that apply.

☐   because it is an example of needless complexity

☑   because any errors will be propogated to all clones

☑   because changing one clone means changing all clones

☐ it isn't actually problematic, because it is done in practice

✔

Submit	You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 9

1/1 point (graded)
Which of the following are CORRECT definitions of the corresponding SOLID design principle?
Select all that apply.

☐ Open/Closed: open to modification, closed to extension

☑ Open/Closed: open to extension, closed to modification

☑ Interface Segregation: clients should not depend on unnecessary interfaces

☐ Interface Segregation: interfaces should do one thing, and do it well

☐ Dependency Inversion: clients should depend on implementations, not abstractions

☑ Dependency Inversion: clients should depend on abstractions, not implementations

✔

Submit	You have used 1 of 1 attempt

✔ Correct (1/1 point)