Started on	Wednesday, 24 April 2024, 10:33 AM
State	Finished
Completed on	Wednesday, 24 April 2024, 10:41 AM
Time taken	7 mins 19 secs
Grade	**81.67** out of 100.00

QUESTION 1

Correct

Mark 10.00 out of 10.00

SOLID are five basic principles that help to create good software architecture. SOLID is an acronym where:-

A parent class object should be able to refer child objects seamlessly during runtime polymorphism.	Liskov substitution principle ✔
A class should take care of only one responsibility.	Single responsibility principle ✔
The Client should not be forced to use an interface if it does not need it.	Interface segregation principle ✔
High-level modules should not depend on low-level modules but should depend on abstraction.	Dependency inversion principle ✔
The extension should be preferred over modification.	Open closed principle ✔

QUESTION 2

Correct

Mark 10.00 out of 10.00

This class satisfies the 'Single responsibility principle'?

```
public class Order
{
        public Guid ID { get { /*…*/ } }
        public decimal Total { get { /*…*/ } }
        public List<Item> Items { get { /*…*/ } }
        public User User { get { /*…*/ } }
        public void Load() { /*…*/ }
        public void Save() { /*…*/ }
        public void Update() { /*…*/ }
        public void Remove() { /*…*/ }
 }
```

Select one:

- True
- False ✔

QUESTION 3

Correct

Mark 10.00 out of 10.00

What is mean by “Module” and “Reason to change” in explanation?

Reason to change: Responsibility ✔

Module: May be class, function, namespace ✔

Every Modules should contain only one responsibility. Here the word responsibility depends on the context where we are talking like, when we say module as class responsibility indicates “database responsibility”, “reporting responsibility” etc., when we say module as function it indicates “Generating xml from list”, “using generated xml for inserting multiple records into sql”.

QUESTION 4

Correct

Mark 10.00 out of 10.00

If you have a library containing a set of classes there are many reasons for which you will prefer to extend it without changing the code that was already written (backward compatibility, regression testing). This is why we have to make sure our modules follow the ... .

Answer: Open/Closed Principle ✔

QUESTION 5

Correct

Mark 10.00 out of 10.00

According to Robert Martin there are 3 important characteristics of a bad design that should be avoided:

Select one or more:

- a. Mobility
- b. Safity
- c. Rigidity ✔
- d. Fragility ✔
- e. Immobility ✔

QUESTION 6

Correct

Mark 10.00 out of 10.00

SOLID is a Design Pattern

Select one:

- True
- False ✔

QUESTION 7

Correct

Mark 10.00 out of 10.00

This principle states that if we have two reasons to change for a class, we have to split the functionality into two classes.

Answer: Single Responsibility Principle ✔

QUESTION 8

Partially correct

Mark 6.67 out of 10.00

What cases can explain Liskov's Substitution Principle?

Select one or more:

- [x] a. Derived types must be completely substitutable for their base types. ✔
- [ ] b. The new derived classes should not be able to replace the base classes without any change in the code.
- [ ] c. We must make sure that new derived classes are extending the base classes without changing their behavior.
- [x] d. The new derived classes should be able to replace the base classes without any change in the code. ✔
- [ ] e. Base types must be completely substitutable for their derived types.

QUESTION 9

Incorrect

Mark 0.00 out of 10.00

Due to the Interface Segregation Principle 'Clients should be forced to depend upon interfaces that they don't use'.

Select one:

- (•) True ✖
- ( ) False

QUESTION 10

Partially correct

Mark 5.00 out of 10.00

Dependency Inversion Principle states that we should decouple high-level ✔ modules from low-level modules, introducing an abstraction layer between the high-level ✔ classes and low-level classes. Furthermore, it inverts the dependency: instead of writing our details ✖ based on abstractions ✖ , then we should write the details based on abstractions.

high-level | abstractions | details

◄ Tasks. SOLID principles (part 2)

Jump to...

DB Design ►