quiz solution w1

Question 1

When is the best time to use a design pattern? Choose **two** answers.

☐ **For a problem that is unique to your program.**

This should not be selected

Incorrect. Design Patterns are for addressing common design issues.

☑ ~~**For a commonly-encountered issue.**~~

Correct

Correct! Design patterns will help you with issues that developers have encountered often. Remember that sometimes they need some adapting.

☐ **When fixing spaghetti code**

This should not be selected

Incorrect. While design patterns could be used to fix spaghetti code, it is best to use them earlier.

☑ ~~**When explaining a solution to your fellow developers**~~

Correct

Correct! Design patterns are coding solutions!

Question 2

What is the purpose of the Singleton pattern? Select the **two correct** answers.

☑ ~~**to provide global access to an object**~~

Correct

Correct. The Singleton pattern makes the one instance of the Singleton class globally accessible.

☐ **to enforce collaboration of a class with only one other class**

This should not be selected

Incorrect.The Singleton class is about how many instances of it, not how many collaborations

- [x] ~~to enforce instantiation of only one object of a class~~

Correct

Correct. The Singleton pattern enforces one and only one instantiation of the Singleton class.

- [ ] **to provide simple classes with only one method**

This should not be selected

Incorrect. The "single" in Singleton refers to enforcing one instance of a class, not only one method.

Question 3

What does it mean to "let the subclass decide" in the Factory Method Pattern?

- [x] ~~the subclass defines the methods for concrete instantiation. As such, the type of object is determined by which subclass is instantiated.~~

Correct

Correct! This is how the subclass "decides." By selecting a subclass you are limited to its concrete instantiation method.

- [ ] **the subclass decides which object to create, but calls a method that is defined in the superclass to instantiate the class**

Incorrect

Incorrect. Think of which method is responsible for object creation in the Factory Method pattern

- [ ] **the subclass will pass a parameter into a factory that determines which object is instantiated.**

Question 4

What do we call the creation of an object, for example, with the 'new' operator in Java?

- [x] ~~concrete instantiation.~~

Correct

Correct! Instantiation is the act of creating an instance of a class, while concrete means the actual act of doing it (rather than speaking about it in general terms, like some interface for creating objects).

☐ **manifestation**

Incorrect

Incorrect. This is not a term that is used.

☐ **object realization**

☐ **class creation**

Question 5

What are the advantages of the **Facade** pattern? Select the **three correct** answers.

☑ ~~The client and the subsystem are more loosely coupled~~

Correct

Correct! If the subsystem or client are changed, there are fewer connections to manage.

☑ ~~The Facade class redirects requests as needed~~

Correct

Correct! This is one of the ways that the Facade can simplify for the client.

☑ ~~The complexity of the subsystem is hidden~~

Correct

Correct! The Facade presents a simplified interface to clients.

☐ **The subsystem can handle more clients**

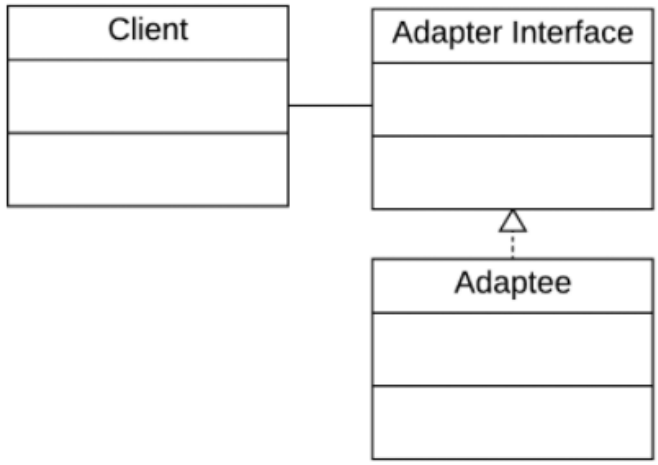
This should not be selected

Incorrect. There is nothing about the Facade pattern that allows you to add more clients. It may be easier to add clients, however, since the Facade is a simpler interface

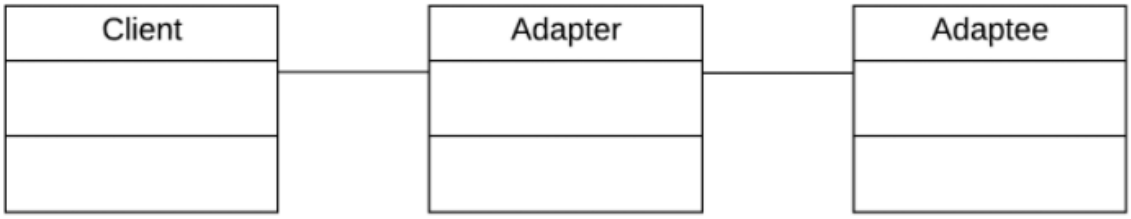
Question 6

Which of the following diagrams shows the **Adapter** pattern?

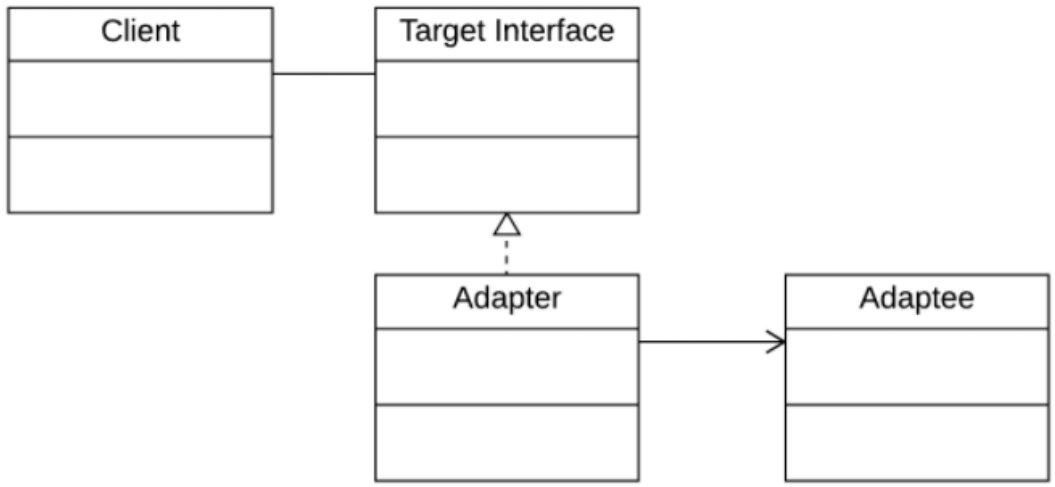
a)



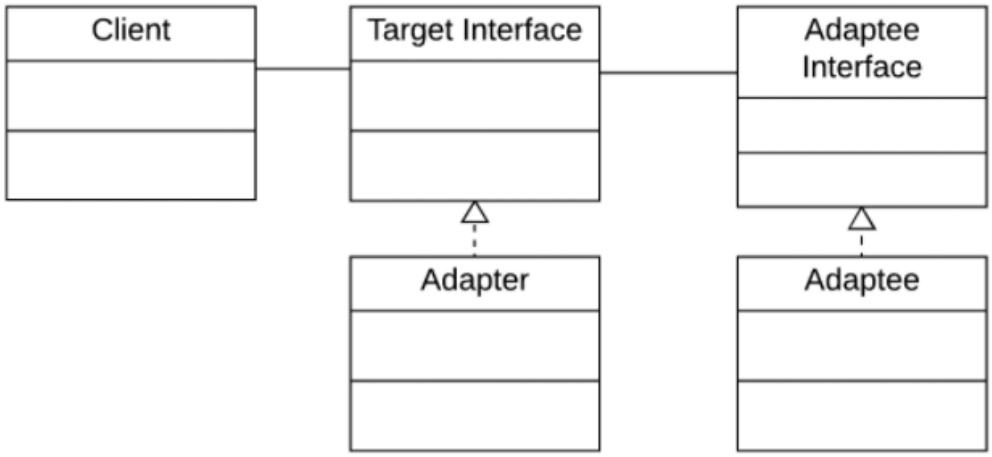
b)



c)



d)



- [ ] **a)**
- [ ] **b)**
- [ ] **d)**
- [x] ~~c)~~

Correct

Correct! The adapter wraps the adaptee and provides its functionality as a target interface that the client can connect with.

Question 7

Which of these are the best applications for a **Composite** Pattern? Choose the **three correct** answers.

- [x] ~~Music in a playlist~~

Correct

Correct! Each playlist can be composed of songs or other playlists -- or a combination of both.

☐ **Students in a class**

This should not be selected

Incorrect. Students in a class cannot contain other students so there is no need for Composite.

☑ ~~**Elements in a user-interface dialog**~~

Correct

Correct! Elements in a dialog may contain other elements (a composite class) or they may not (leaf class)

☑ ~~**Files and folders**~~

Correct

Correct! Folders (composite class) can contain other folders, or files (leaf class)

Question 8

Which of these is **NOT** a common application of the **Proxy** Pattern?

☐ **remote proxy**

Incorrect

Incorrect. Remote proxies have a local representation of a remote resource. Try again!

☐ **protection proxy**

Incorrect

Incorrect. Protection proxies are used for access restriction or gatekeeping on a component. Try again!

☐ **virtual proxy**

Incorrect

Incorrect. Virtual proxies are used when the subject is resource intensive. Try again!

☑ ~~information proxy~~

Question 9

How does a **Decorator** Pattern work? Choose one.

☐ **encapsulates a class to give it a different interface**

Incorrect

Incorrect. This is adapter pattern.

☐ **expands the methods of a class with inheritance**

Incorrect

Incorrect. This is not one of the patterns but is commonly done in object-oriented code.

☑ ~~builds a behaviour by stacking objects~~

Correct

Correct! This accurately describes a Decorator pattern.

☐ **adding features to a class with a new class**

Incorrect

Incorrect. This could be a wrapper or an example of composition.

Question 10

What are the object types that are used in the **Composite** Pattern? Select the **two correct** answers.

☑ ~~leaf~~

Correct

Correct! A leaf is the term for a composite subclass that cannot contain another component

☐ **branch**

This should not be selected

Incorrect. The Composite pattern does involve a tree metaphor though...

☐ **root**

This should not be selected

Incorrect. There's no root object that fits into this pattern.

☑ ~~composite~~

Correct

Correct. A composite object is a component object that can contain other components, instances of either other composites, or leaf classes.

☐ **trunk**

This should not be selected

Incorrect. There's no trunk object that fits into this pattern.

Question 11

Many different clients need to create a similar object. You would like to outsource this concrete instantiation to a dedicated class. Which technique will you use, in one word?

Factory

Correct

The correct answer is factory. Factories of different types are used to instantiate objects. This could be a simple factory, which is an object which is tasked with concrete instantiation. Factory Methods move concrete instantiation is achieved by a method- that is abstract in the superclass and specified in the subclass.

Question 12

How do you enforce the creation of only one Singleton object? Select the **two correct** answers.

☑ ~~Give the Singleton class a private constructor~~

Correct

Correct. This essentially only allows the Singleton to construct itself, which it will not do if it is already instantiated once.

☐ **Specify in the comments that only one Singleton object is to be instantiated.**

This should not be selected

Incorrect. You may do this but the goal of Singleton is to codify the design intent.

☐ **Throw an exception if a Singleton object is already instantiated**

This should not be selected

Incorrect. A Singleton object simply returns itself if you try to create a new one!

☑ ~~**Write a method that can create a new Singleton object or return the existing one.**~~

Correct

Correct! If the Singleton class is already instantiated, simply return that object. If it doesn't, make it!