1. When is the best time to use a design pattern? Choose **two** answers. 1 / 1 point

- ☐ For a problem that is unique to your program.
- ☑ 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
- ☑ When explaining a solution to your fellow developers

> **Correct**
> Correct! Design patterns are coding solutions!

2. What is the purpose of the Singleton pattern? Select the **two correct** answers. 1 / 1 point

- ☐ to enforce collaboration of a class with only one other class
- ☑ 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 global access to an object

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

- ☐ to provide simple classes with only one method

3. What does it mean to "let the subclass decide" in the Factory Method Pattern? 1 / 1 point

- ○ the subclass decides which object to create, but calls a method that is defined in the superclass to instantiate the class
- ○ the subclass will pass a parameter into a factory that determines which object is instantiated.
- ● 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.

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

1 / 1 point

○ class creation

● concrete instantiation.

○ object realization

○ manifestation

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).

5.

1 / 1 point

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

☑ The complexity of the subsystem is hidden

Correct
Correct! The Facade presents a simplified interface to clients.

☑ The client and the subsystem are more loosely coupled

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

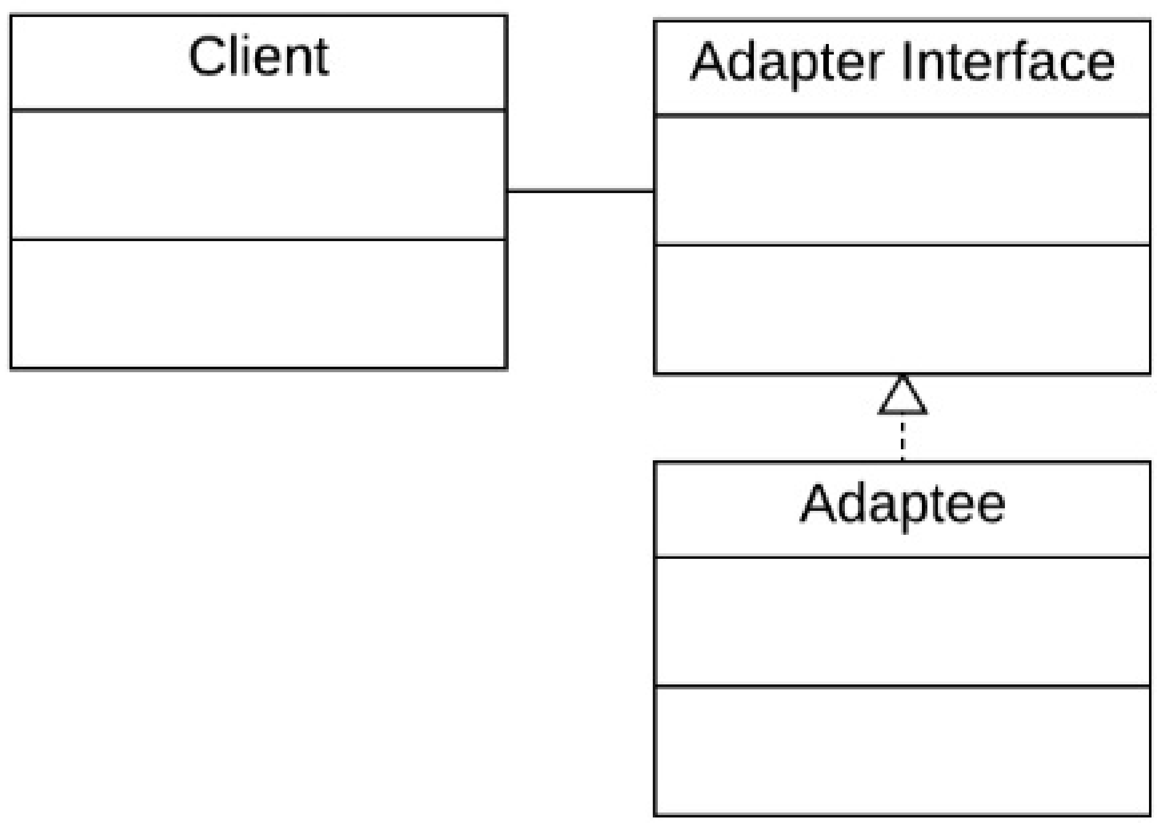
☑ The Facade class redirects requests as needed

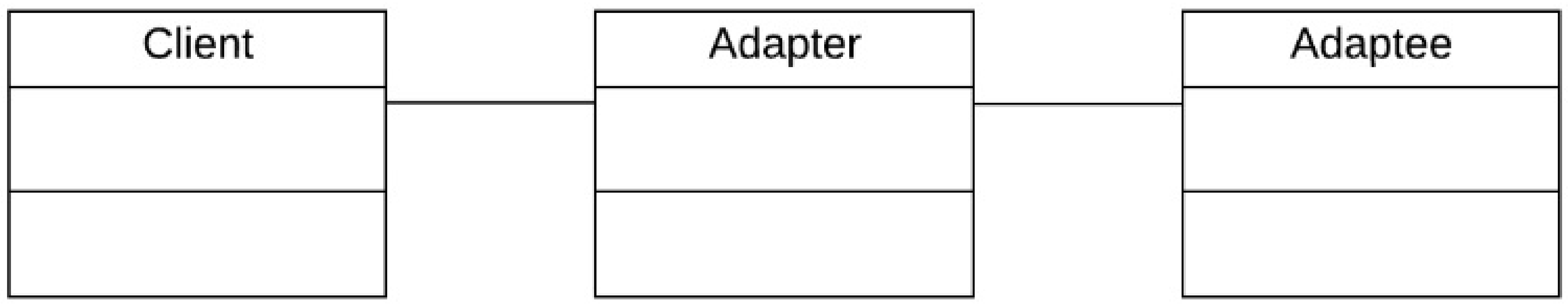
Correct
Correct! This is one of the ways that the Facade can simplify for the client.

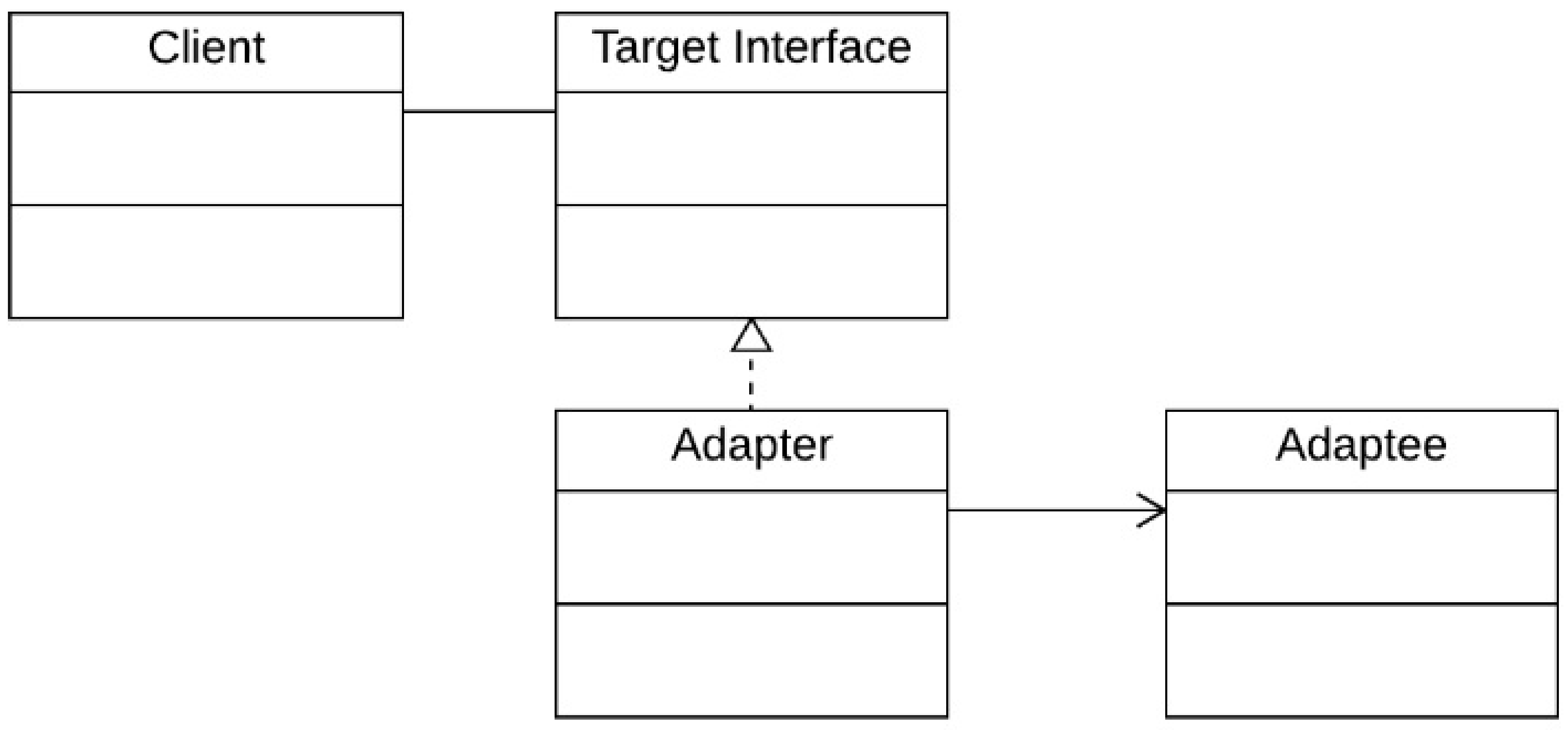
☐ The subsystem can handle more clients

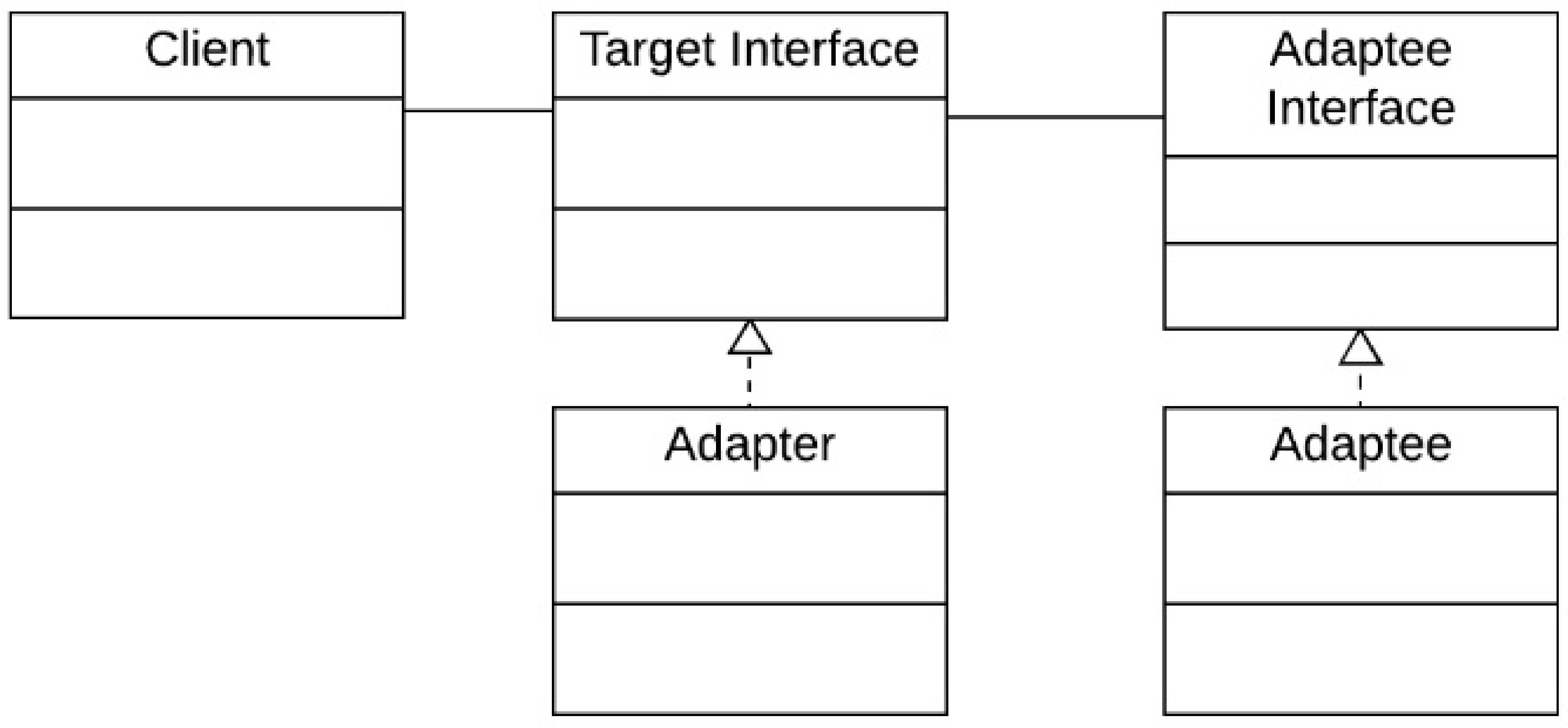
6. 1 / 1 point

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

a)


b)


c)


d)


○ a)

○ b)

◉ c)

○ d)

Correct

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

7. Which of these are the best applications for a **Composite** Pattern? Choose the **three correct** answers. 1 / 1 point

- [✓] 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)

- [✓] Music in a playlist

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

- [✓] Files and folders

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

- [ ] Students in a class

8. Which of these is **NOT** a common application of the **Proxy** Pattern? 1 / 1 point

- remote proxy
- [✓ information proxy]
- virtual proxy
- protection proxy

  Correct
  You got it! Information proxy is not a common application of the Proxy pattern.

9. How does a **Decorator** Pattern work? Choose one. 1 / 1 point

- adding features to a class with a new class
- encapsulates a class to give it a different interface
- [✓ builds a behaviour by stacking objects]
- expands the methods of a class with inheritance

  Correct
  Correct! This accurately describes a Decorator pattern.

10. What are the object types that are used in the **Composite** Pattern? Select the **two correct** answers. 1 / 1 point

- [ ] branch
- [x] leaf

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

- [x] composite

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

- [ ] root
- [ ] trunk

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? 1 / 1 point

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.

12. How do you enforce the creation of only one Singleton object? Select the **two correct** answers. 1 / 1 point

- [x] 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.
- [ ] Throw an exception if a Singleton object is already instantiated
- [x] 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!