Mentimeter

Instructions

How is it???

0 responses

Mentimeter

Tell me if you have problems with practictical tasks

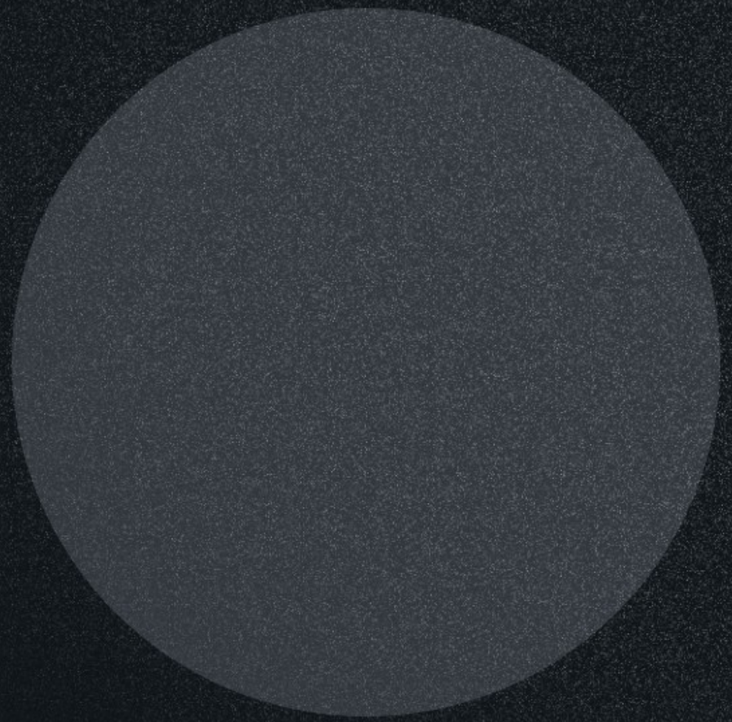
Agenda

- Recall what was learned
- Look at current topic practice
- Check current topic knowledge

Total Recall

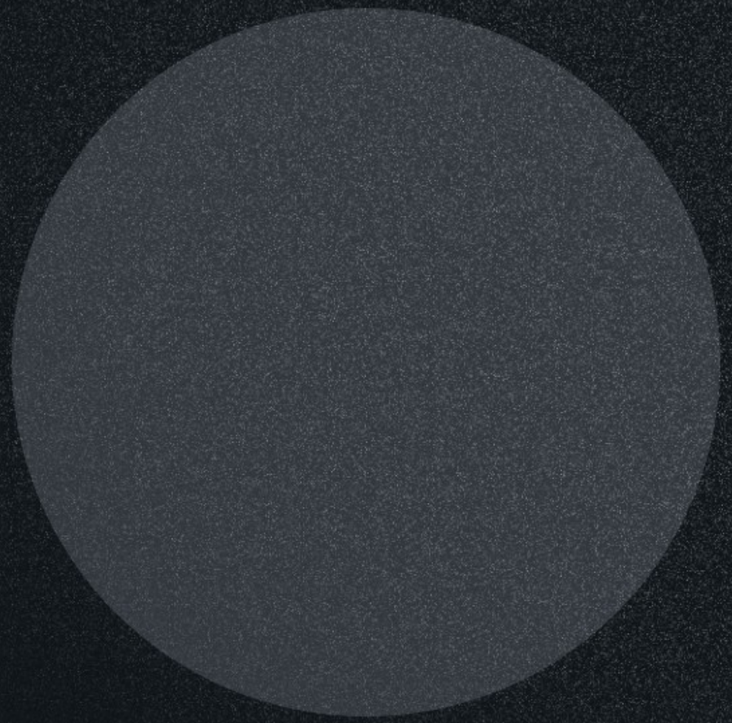
Is shallow copy allowed for immutable object?

What is the primary purpose of the visitor pattern?



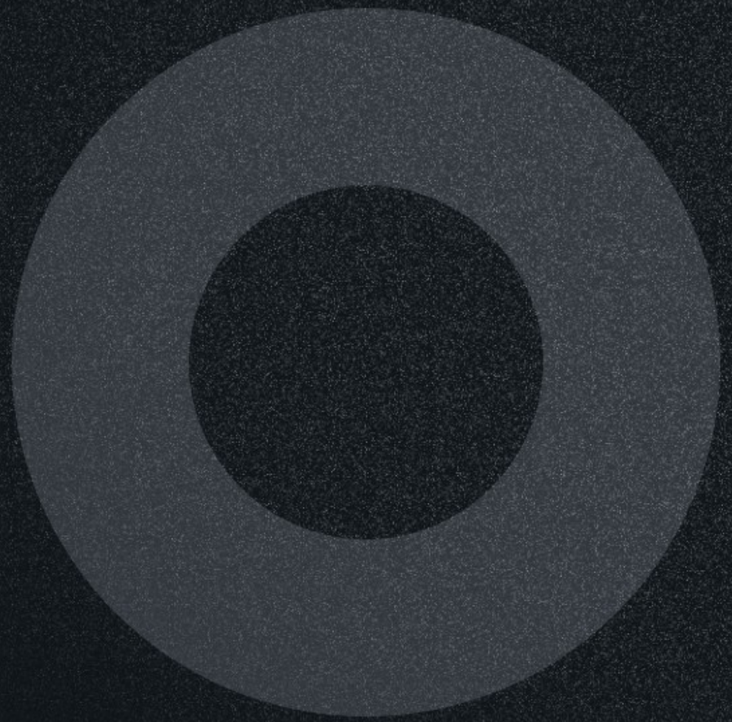
- To enforce a singleton pattern ✗
- To define a one-to-many dependency between objects ✗
- To create families of related or dependent objects without specifying their concrete classes ✗
- To define a new operation to be performed on each element in an object structure ✓
- To provide a flexible way to create objects of different types ✗

What are the benefits of Visitor pattern?



- Open/Closed Principle. You can introduce a new behavior that can work with objects of different classes without changing these classes.
- Single Responsibility Principle. You can move multiple versions of the same behavior into the same class.
- Visitor can store some data while working with various objects. This can be handy when you want to traverse some complex object structure.
- All mentioned

What are the drawbacks of the Visitor?



- You need to update all visitors each time a class gets added to or removed from the element hierarchy. ✓
- Visitors might lack the necessary access to the private fields and methods of the elements that they're supposed to work with. ✓
- You can separate processing logic from data transfer objects. ✗
- All mentioned ✗


Mercury
Discuss current topic

A year on Venus is shorter than a day!

It takes 243 Earth days for Venus to spin around just once
And only 225 Earth days to orbit the Sun completely

Have you looked at todays' materials?

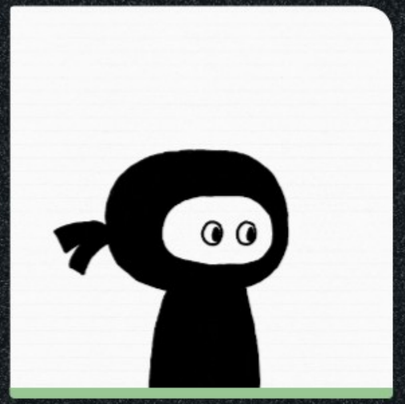
0

yes

0

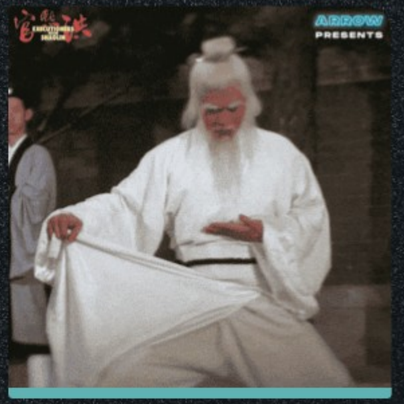
no

0



partially

0



i'm the author

Advanced Patterns: Proxy

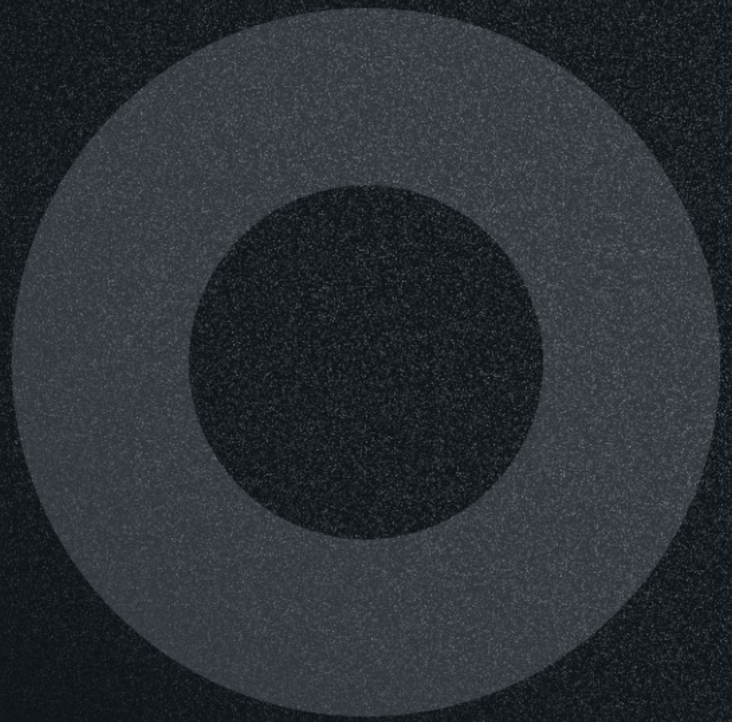
Pros

- You can control the service object without clients knowing about it.
- You can manage the lifecycle of the service object when clients don’t care about it.
- The proxy works even if the service object isn’t ready or is not available.
- Open/Closed Principle. You can introduce new proxies without changing the service or clients.

Cons

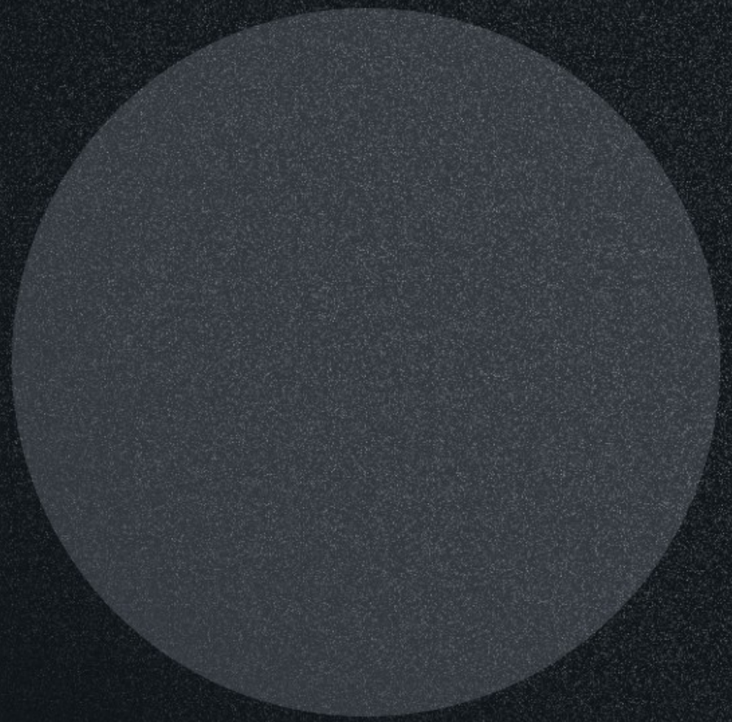
- The code may become more complicated since you need to introduce a lot of new classes.
- The response from the service might get delayed.

What is the primary function of the Proxy Pattern?

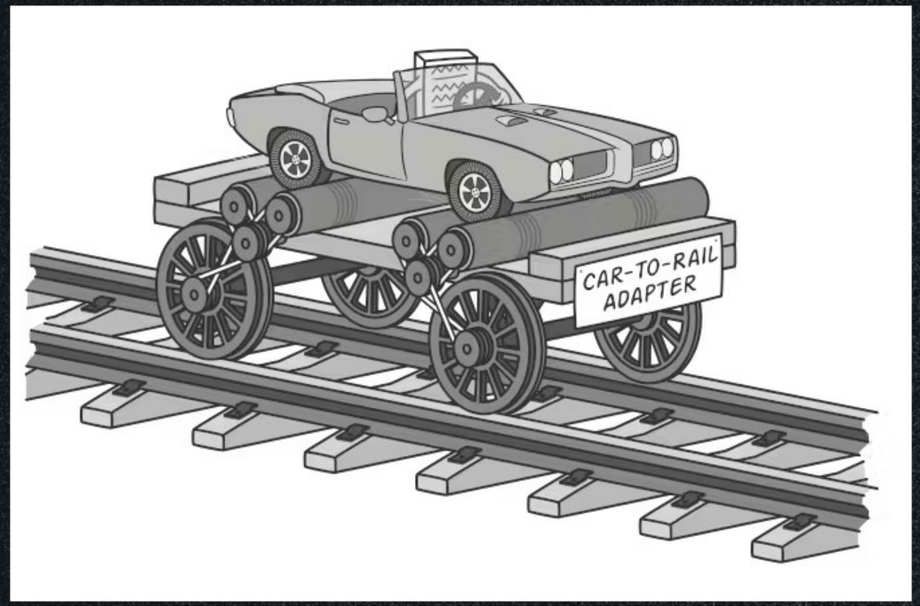


- To store data for quick access later
- To simplify the interface of an existing object
- To add an intermediary layer between a client and an actual object
- To replace the original object with a different one

What is the primary function of the Proxy Pattern?



- Providing logging and caching functionalities ✓
- Implementing an interface for multiple objects ×
- Simplifying complex object interactions ×
- Managing object creation and destruction ×



Advanced Patterns: Adapter

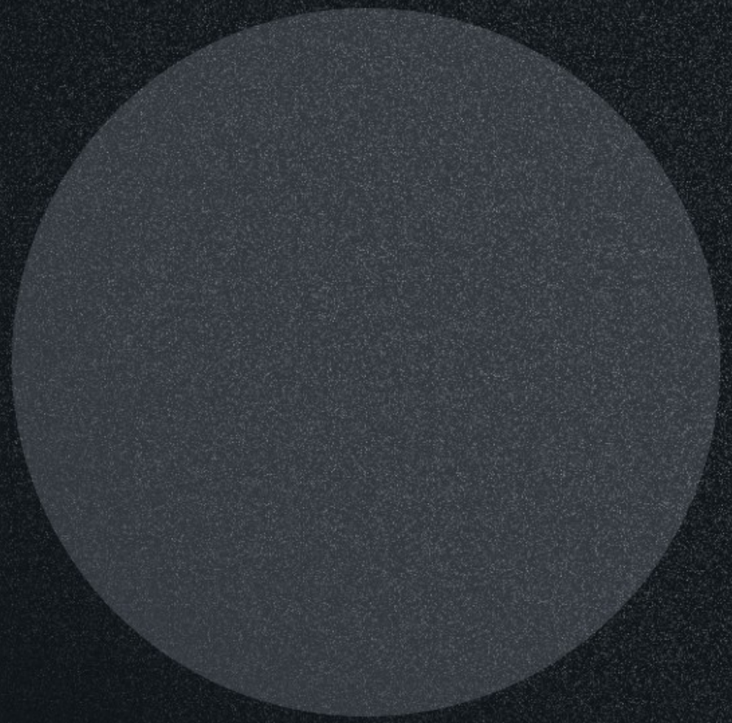
Pros

- Single Responsibility Principle. You can separate the interface or data conversion code from the primary business logic
- Open/Closed Principle. You can introduce new types of adapters into the program without breaking the existing client co

Cons

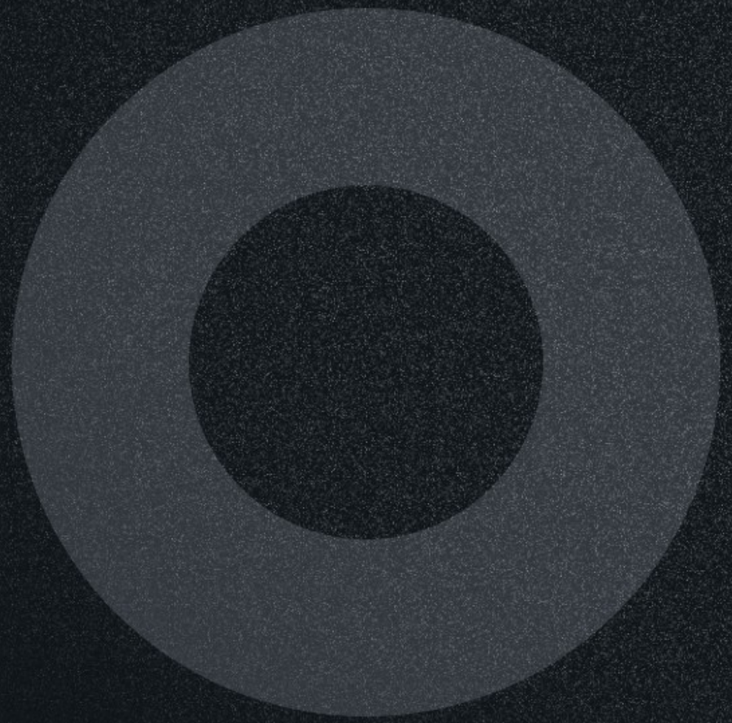
- The overall complexity of the code increases because you need to introduce a set of new interfaces and classes.
- Sometimes it’s simpler just to change the service class so that it matches the rest of your code.

What is the primary function of the Adapter design pattern in software development?



- A) It allows classes with incompatible interfaces to work together by wrapping its own interface around that of an already existing class. ✓
- B) It ensures that a class only has one instance and provides a global point of access to it. ×
- C) It provides an interface for creating families of related or dependent objects without specifying their concrete classes. ×
- D) It allows for the dynamic addition of new operations to objects without modifying the classes. ×

Which of the following is true for Adapter pattern?



- a) An adapter or wrapper is a component that provides a new interface for an existing component
- b) An Adapter or Wrapper pattern is a broker pattern that provides a new interface for existing software so that it can be reused
- c) Adaptation for reuse is an old technique that has been used since the beginning of software development
- d) All of the mentioned

Mentimeter

10 mins break

Advanced Patterns: Decorator

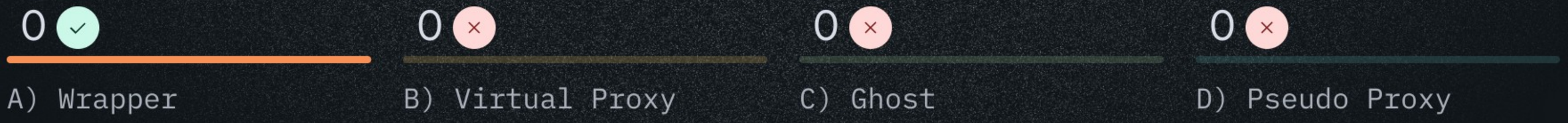
Pros

- You can extend an object’s behavior without making a new subclass.
- You can add or remove responsibilities from an object at runtime.
- You can combine several behaviors by wrapping an object into multiple decorators.
- SRP. You can divide a monolithic class that implements many possible variants of behavior into several smaller classes.

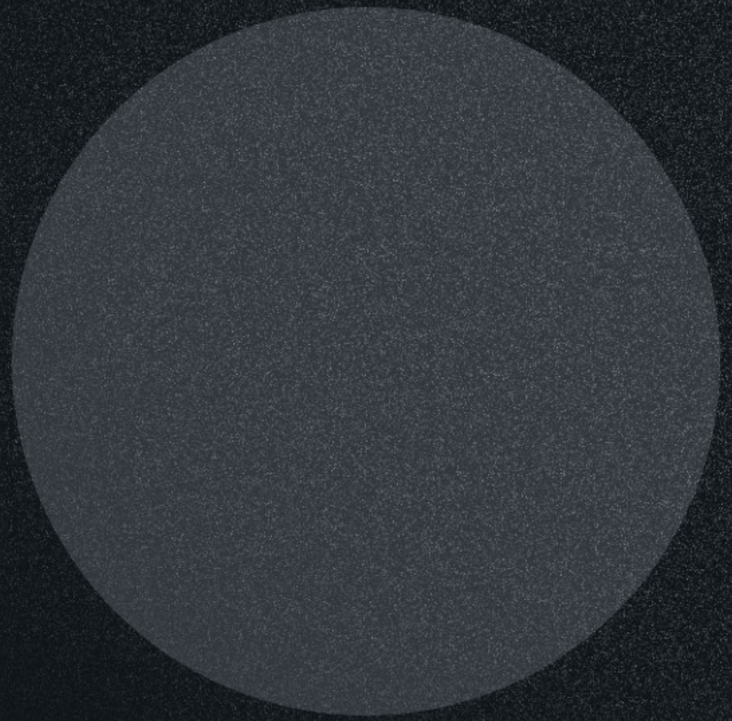
Cons

- It’s hard to remove a specific wrapper from the wrappers stack.
- It’s hard to implement a decorator in such a way that its behavior doesn’t depend on the order in the decorators stack.
- The initial configuration code of layers might look pretty ugly.

The Decorator design pattern is also known as:



What is the main purpose of using the decorator design pattern in object-oriented programming?



- a) To add new methods to an object dynamically ✗
- b) To modify existing behavior of an object dynamically ✓
- c) To enhance the performance of an application ✗
- d) None of the above ✗

Which of the following statements is true about the decorator design pattern?

a) A decorator and the object it wraps must implement the same interface.

b) A decorator can only wrap a single instance of an object.

c) Decorators are mostly used for improving the speed of the system.

d) None of the above

Advanced Patterns: Bridge

When to use

Use the Bridge pattern when you want to divide and organize a monolithic class that has several variants of some functionality (for example, if the class can work with various database servers).

When to use

Use the pattern when you need to extend a class in several orthogonal (independent) dimensions.

When to use

Use the Bridge if you need to be able to switch implementations at runtime.

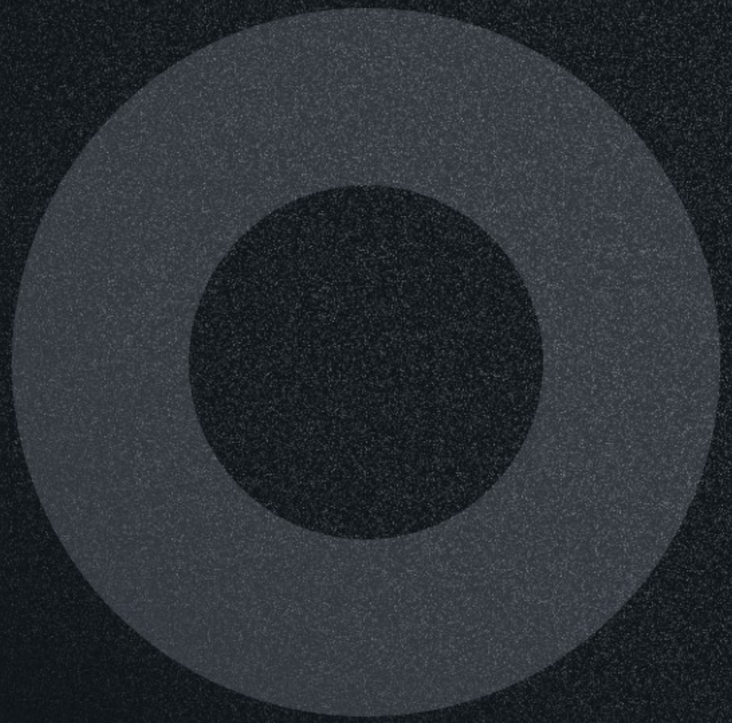
Pros

- You can create platform-independent classes and apps.
- The client code works with high-level abstractions. It isn’t exposed to the platform details.
- Open/Closed Principle. You can introduce new abstractions and implementations independently from each other.
- SRP. You can focus on high-level logic in the abstraction and on platform details in the implementation.

Cons

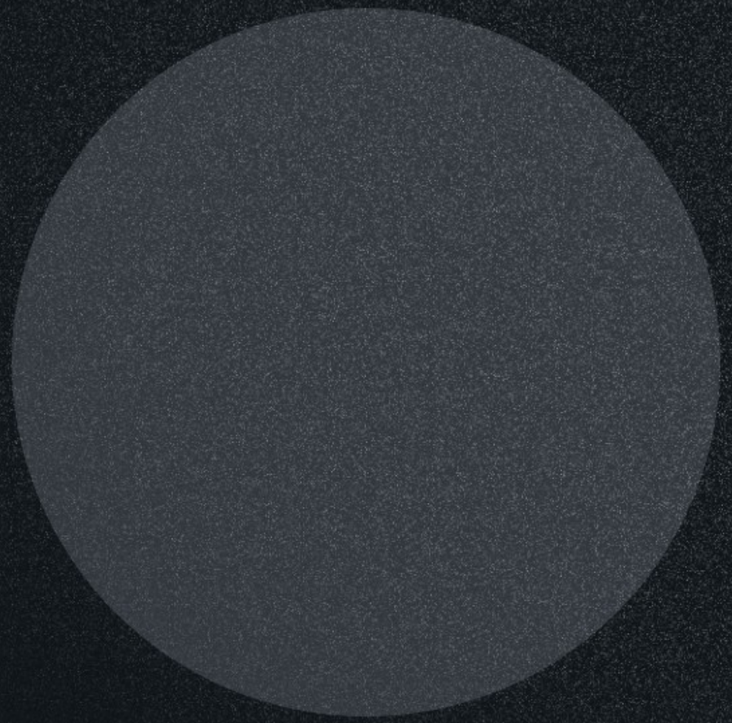
- You might make the code more complicated by applying the pattern to a highly cohesive class.

What is the main purpose of the Bridge design pattern in object-oriented programming?



- a) To allow the implementation details to be changed at runtime. ✗
- b) To separate the interface hierarchy from the implementation hierarchy. ✓
- c) To ensure that classes are closed for modification but open for extension. ✗
- d) To provide a way to configure classes with multiple functionalities. ✗

Which of the following scenarios best suits the use of the Bridge design pattern?



- a) When you need to generate multiple versions of a component without affecting its clients. ✗
- b) When the implementation of a component needs to be chosen or switched at runtime. ✓
- c) When you have several classes that differ only in their implementation details. ✗
- d) All of the above. ✗

Relations with Other Patterns

- With **Adapter** you access an existing object via different interface. With **Proxy**, the interface stays the same. With **Decorator** you access the object via an enhanced interface.
- **Facade** is similar to **Proxy** in that both buffer a complex entity and initialize it on its own. Unlike *Facade*, *Proxy* has the same interface as its service object, which makes them interchangeable.
- **Decorator** and **Proxy** have similar structures, but very different intents. Both patterns are built on the composition principle, where one object is supposed to delegate some of the work to another. The difference is that a *Proxy* usually manages the life cycle of its service object on its own, whereas the composition of *Decorators* is always controlled by the client.

Q&A part

0 questions
0 upvotes