Behavioral Design Patterns

Memento: Allows restoring an object to a previous state.

State: Allows an object to behave differently depending on the state it is in.

Iterator: Allows iterating over an object without having to expose the object's internal structure (which may change in the future).

Strategy: Allows passing different algorithms (behaviours) to an object. Allows defining a template (skeleton) for an operation. Specific steps will then be implemented in subclasses.

Command: Allows decouple a sender from a receiver. The sender will talk to the receive through a command. Commands can be undone and persisted.

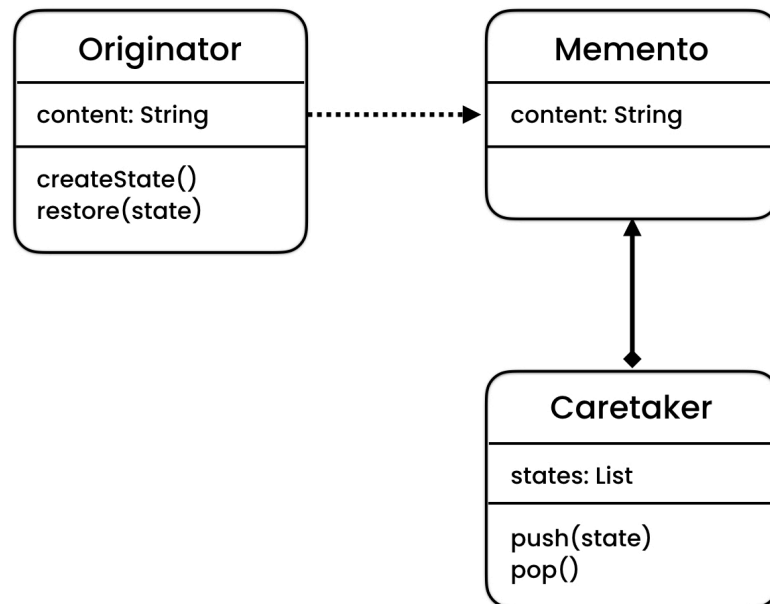**Observer**: Allows an object notify other objects when its state changes.

Mediator: Allows an object to encapsulate the communication between other objects.

Chain of Responsibility: Allows building a chain of objects to process a request.

Visitor: Allows adding new operations to an object structure without modifying it.

Memento Pattern

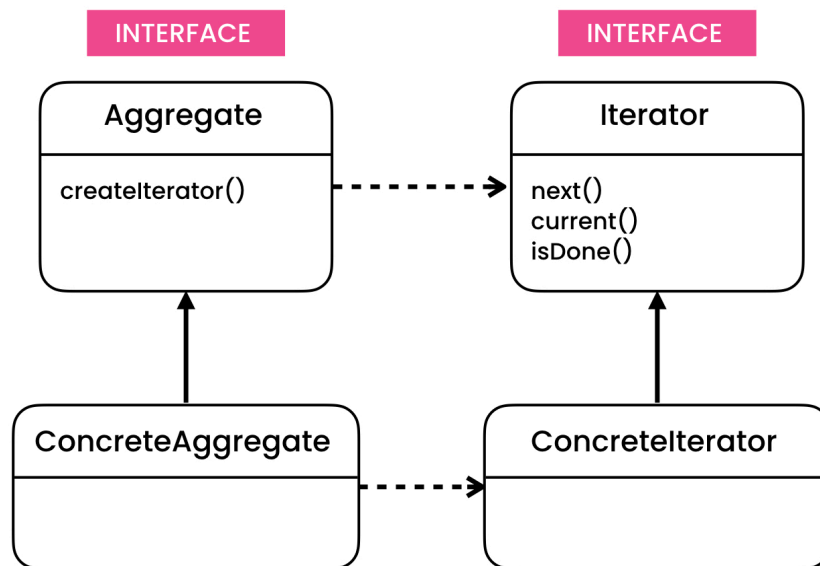Allows restoring an object to a previous state.

Originator	
content: String	
createState() restore(state)	

Memento	
content: String	

Caretaker	
states: List	
push(state) pop()	

State Pattern

Allows an object to behave differently depending on the state it is in.

```
┌─────────────────┐        ┌─────────────────┐
│     Context     │        │      State      │
├─────────────────┤        ├─────────────────┤
│                 │◆──────▶│                 │
├─────────────────┤        ├─────────────────┤
│  request()      │        │  handle()       │
└─────────────────┘        └─────────────────┘
                                    △
                           ┌────────┴────────┐
              ┌─────────────────┐   ┌─────────────────┐
              │  ConcreteStateA │   │  ConcreteStateB │
              ├─────────────────┤   ├─────────────────┤
              │                 │   │                 │
              ├─────────────────┤   ├─────────────────┤
              │  handle()       │   │  handle()       │
              └─────────────────┘   └─────────────────┘
```
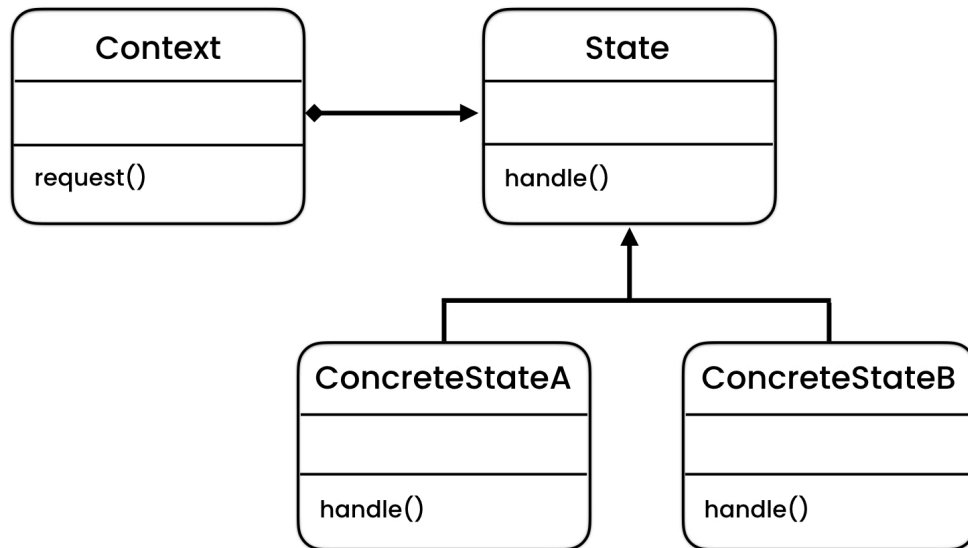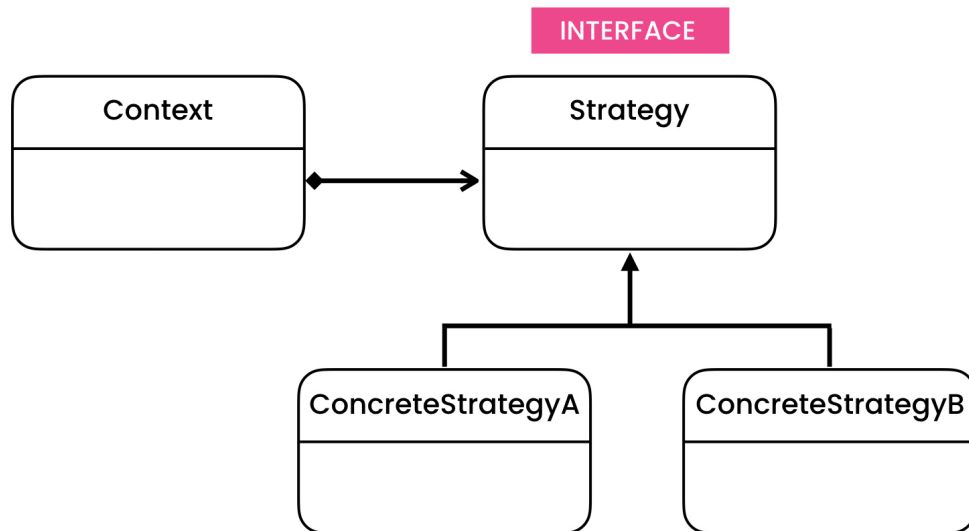
Iterator Pattern

Allows iterating over an object without having to expose the object's internal structure (which may change in the future).

INTERFACE

INTERFACE

Aggregate

createIterator()

Iterator

next()
current()
isDone()

ConcreteAggregate

ConcreteIterator

Strategy Pattern

Allows passing different algorithms (behaviours) to an object.

Context	Strategy

INTERFACE

ConcreteStrategyA   ConcreteStrategyB

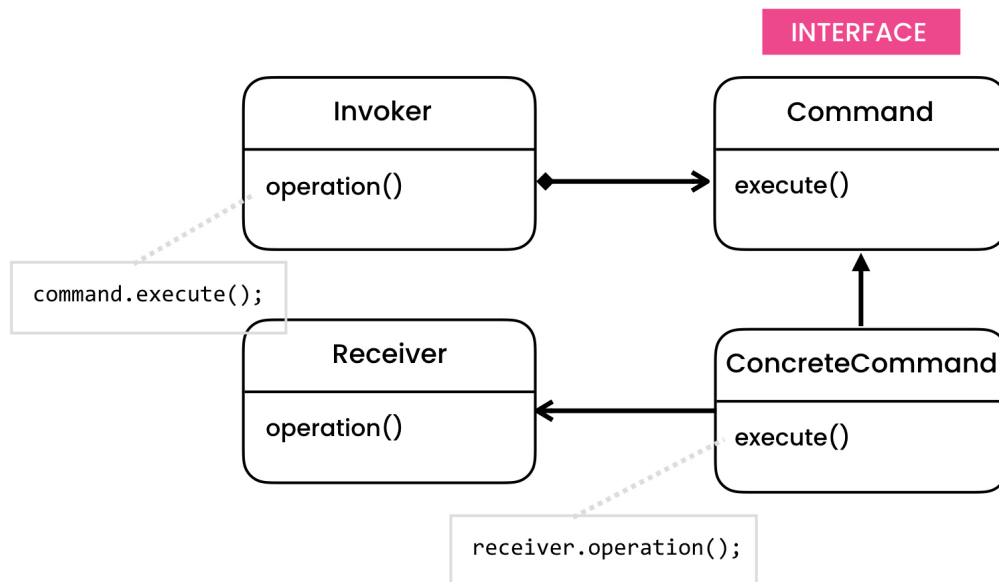# Template Method Pattern

Allows defining a template (skeleton) for an operation. Specific steps will then be implemented in subclasses.

```
AbstractClass
─────────────────────
templateMethod()
primitiveOperation1()
primitiveOperation2()
```

```
…
primitiveOperation1()
…
primitiveOperation2()
```

```
ConcreteClass
─────────────────────
primitiveOperation1()
primitiveOperation2()
```
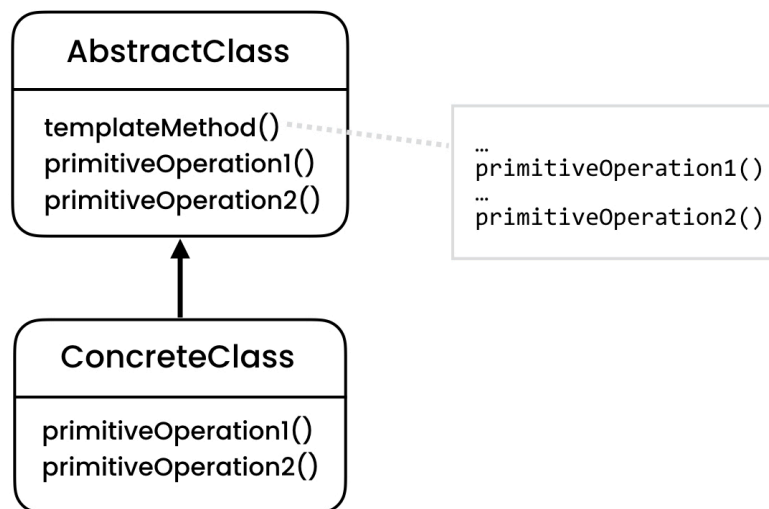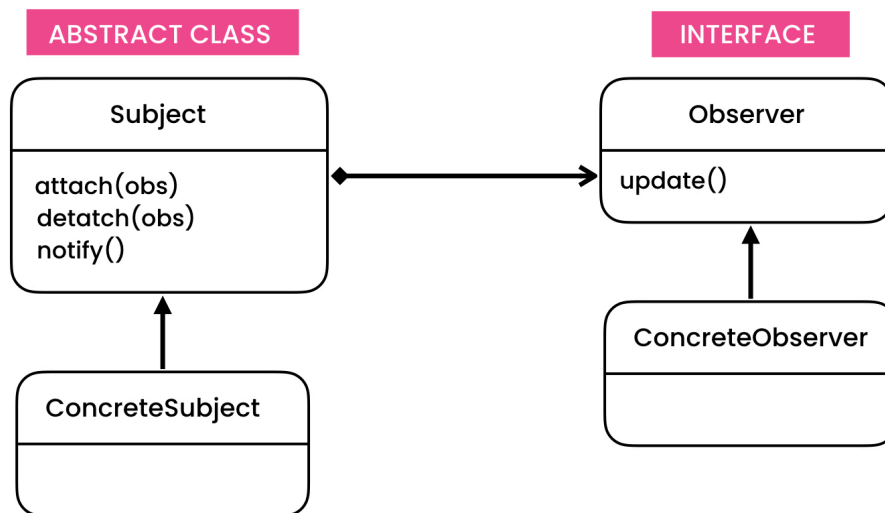
Command Pattern

Allows decouple a sender from a receiver. The sender will talk to the receive through a command. Commands can be undone and persisted.

```
                                          INTERFACE
┌─────────────────┐              ┌─────────────────┐
│     Invoker     │              │     Command     │
├─────────────────┤              ├─────────────────┤
│  operation()    │◆──────────►  │  execute()      │
└─────────────────┘              └─────────────────┘
   ┆                                      ▲
┌──────────────────┐                      │
│ command.execute();│                     │
└──────────────────┘                      │
┌─────────────────┐              ┌─────────────────┐
│    Receiver     │              │ ConcreteCommand │
├─────────────────┤              ├─────────────────┤
│  operation()    │◄─────────────│  execute()      │
└─────────────────┘              └─────────────────┘
                        ┆
              ┌──────────────────────┐
              │ receiver.operation(); │
              └──────────────────────┘
```
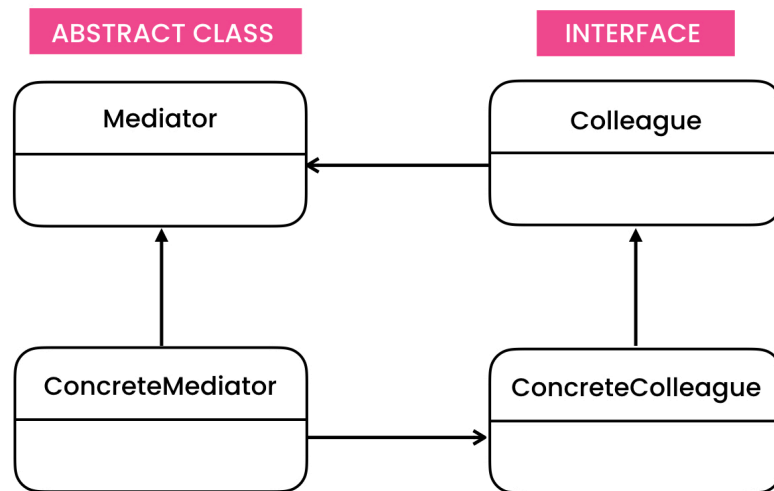
Observer Pattern

Allows an object notify other objects when its state changes.

Mediator Pattern

Allows an object to encapsulate the communication between other objects.

ABSTRACT CLASS	INTERFACE
Mediator	**Colleague**

```
ABSTRACT CLASS              INTERFACE

  ┌──────────────┐        ┌──────────────┐
  │   Mediator   │◄───────│   Colleague  │
  │              │        │              │
  └──────────────┘        └──────────────┘
         ▲                       ▲
         │                       │
  ┌──────────────┐        ┌──────────────┐
  │ConcreteMediator│─────►│ConcreteColleague│
  │              │        │              │
  └──────────────┘        └──────────────┘
```

Chain of Responsibility Pattern

Allows building a chain of objects to process a request.

ABSTRACT CLASS

Sender
handle(request)

Handler
handle()

next

`handler.handle(request);`

ConcreteHandler1

ConcreteHandler2

Visitor Pattern

Allows adding new operations to an object structure without modifying it.