State
Design Pattern

Paul Linca
30432

Introduction

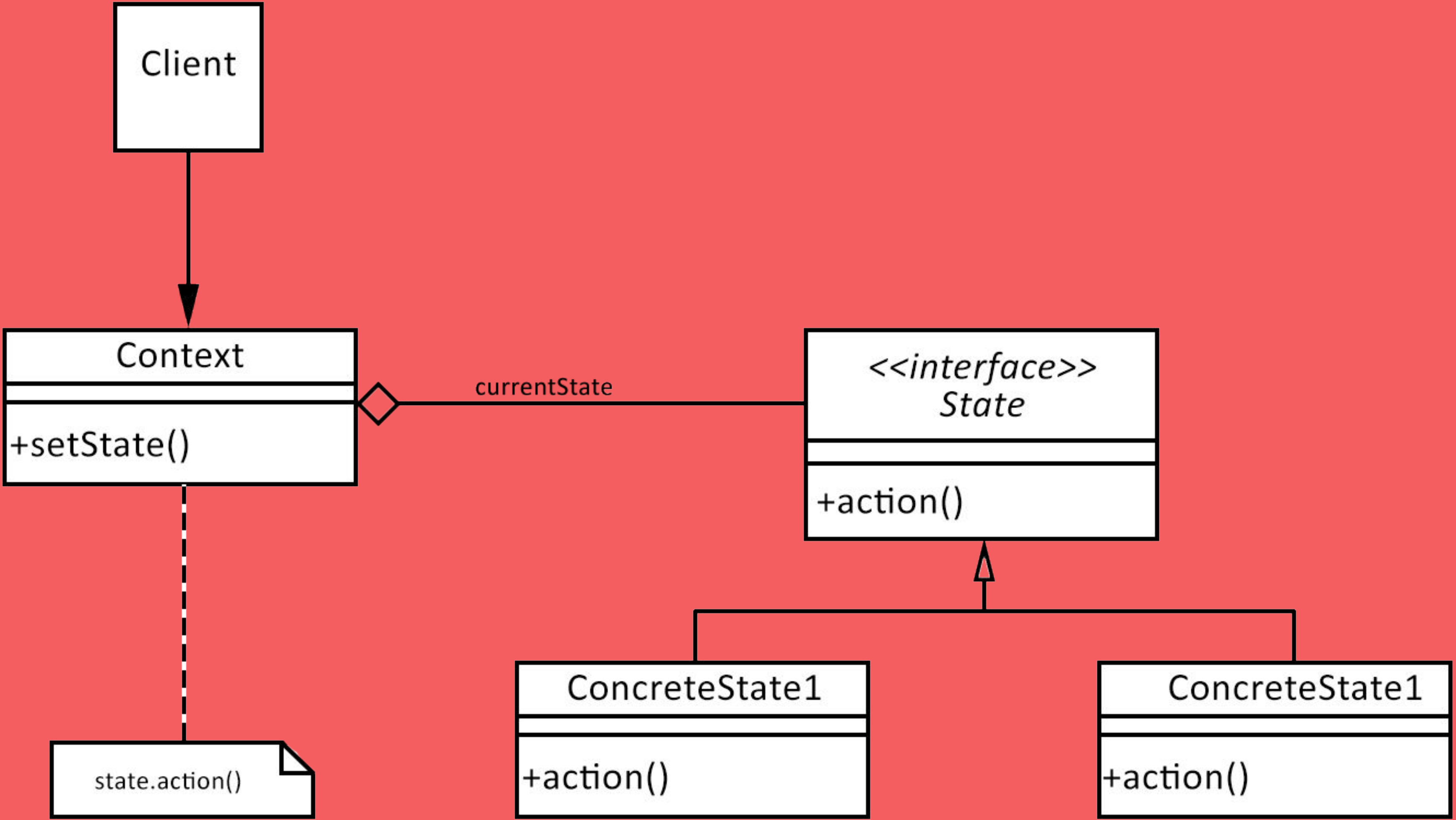
- The state design pattern is a behavioral design pattern:
  - Interaction and responsibility of objects.
- State pattern is used when an object changes its behavior based on its internal state.
  - The object will appear to change its class.

When to use

- When an object has a relatively complex set of possible states and multiple business rules.
- When the states and behavior have to be updated at any time.

Components

- **Context** - defines an interface for the client to interact with. It maintains an instance of a ConcreteState used to define the current state of the object.
- **State** - defines an interface for encapsulating the behavior of each concrete state - declares what each state should do.
- **ConcreteState** - provides the implementation for the methods in state.


Client
Context
+setState()
currentState
<<interface>>
State
+action()
ConcreteState1
+action()
ConcreteState1
+action()
state.action()

Example

```
class MobileStateContext
{
	private AlertState currentState;
	public void setState(AlertState state)
	{
		currentState  = state;
	}
	public void alert()
	{
		currentState.alert(this);
	}
}
```

```
interface AlertState
{
	public void alert(MobileStateContext ctx);
}
class Silent implements AlertState
{	@Override
	public void alert(MobileStateContext ctx)
	{
		sout(“silence...”);
	}
}
class Vibration implements AlertState
{	@Override
	public void alert(MobileStateContext ctx)
	{
		sout(“vibration...”);
	}
}
```

Example

```
class Main
{
        Public static void main()
        {
                MobileStateContext context =
                        new MobileStateContext();

                context.setState(new Silent());
                context.alert();

                context.setState(new Vibration());
                context.alert();
        }
}
```

```
> silence...

>vibration...
```

Advantages and Disadvantages

Advantages

- Polymorphic behavior.
- Minimizes conditional complexity.
- Easy to add new states/behavior.
- Improves cohesion.

Disadvantages

- Large amount of code.