Observer and State pattern

Table of Contents

2018.09.04

VC DCV

1. Observer pattern – Problem

- You are the product manufacturer. You have about ten agencies (it's fixed).
- You want to develop an application that notify to your agencies when you made new product, when you change the price of other product and so on.
- A solution is very easy. When your company has a change, you will notify to ten agencies statically via using loop operation.
- One day, your company grow up very fast. You have vey big agencies, not ten. And it's not fixed. The old application can't handle this case and need modification.

1. Observer pattern – Problem

- If you keep old design for your application, once your agencies changes you need to update your application.
- Observer pattern will help you to develop your application meet with the above requirement.
- When you have new agency, you only need to register.
- When a agency no longer to get notify, you need to unregister it.

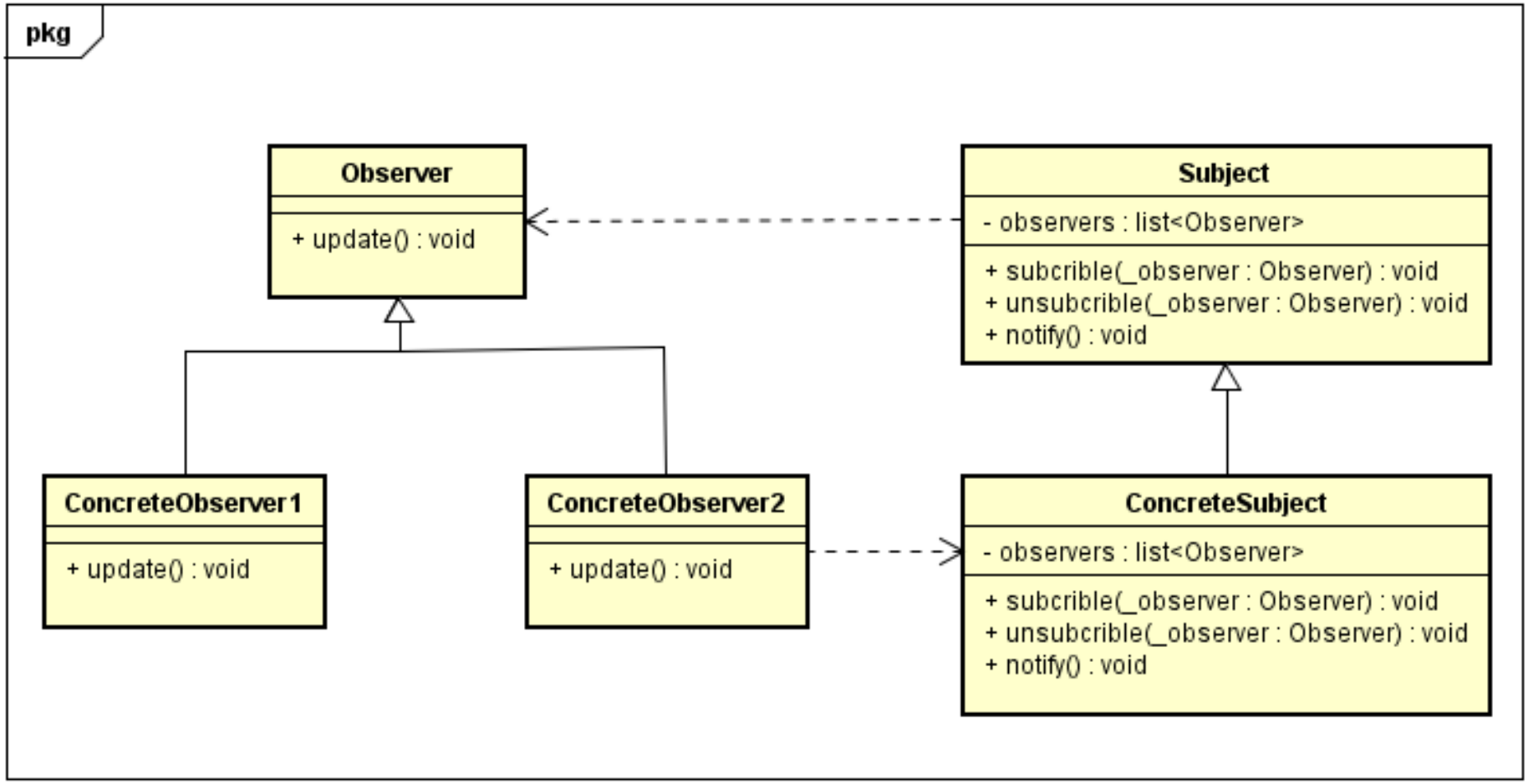
1. Observer pattern – Definition

- The **observer pattern** is a software design pattern in which an object, called the **subject**, maintains a list of its dependents, called **observers**, and notifies them automatically of any state changes, usually by calling one of their methods.

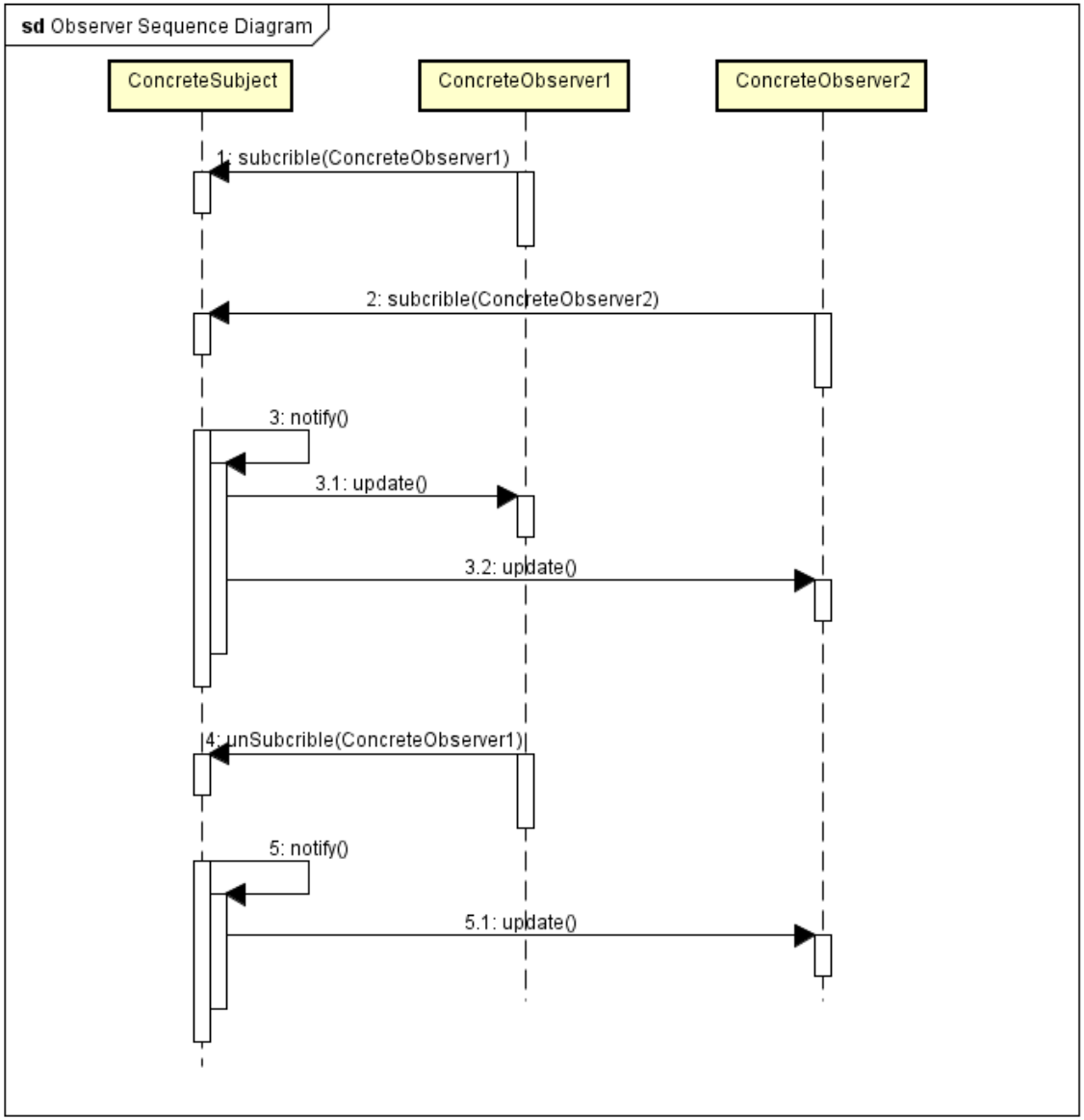
- The Observer [2] design pattern is one of the twenty-three well-known *"Gang of Four" design patterns* that describe how to solve recurring design problems to design flexible and reusable object-oriented software, that is, objects that are easier to implement, change, test, and reuse.

- Observer pattern falls under behavioral pattern category.

1. Observer pattern – Class diagram



1. Observer pattern – Sequence diagram



1. Observer pattern – Advantages

- Observer pattern provides a loose coupling as:
- Subject only knows that observer implement Observer interface. Nothing more.
- There is no need to modify Subject to add or remove observers.
- We can reuse subject and observer classes independently of each other.

1. Observer pattern – Disadvantages

- Memory leaks caused by Lapsed listener problem because of explicit register and unregistering of observers.

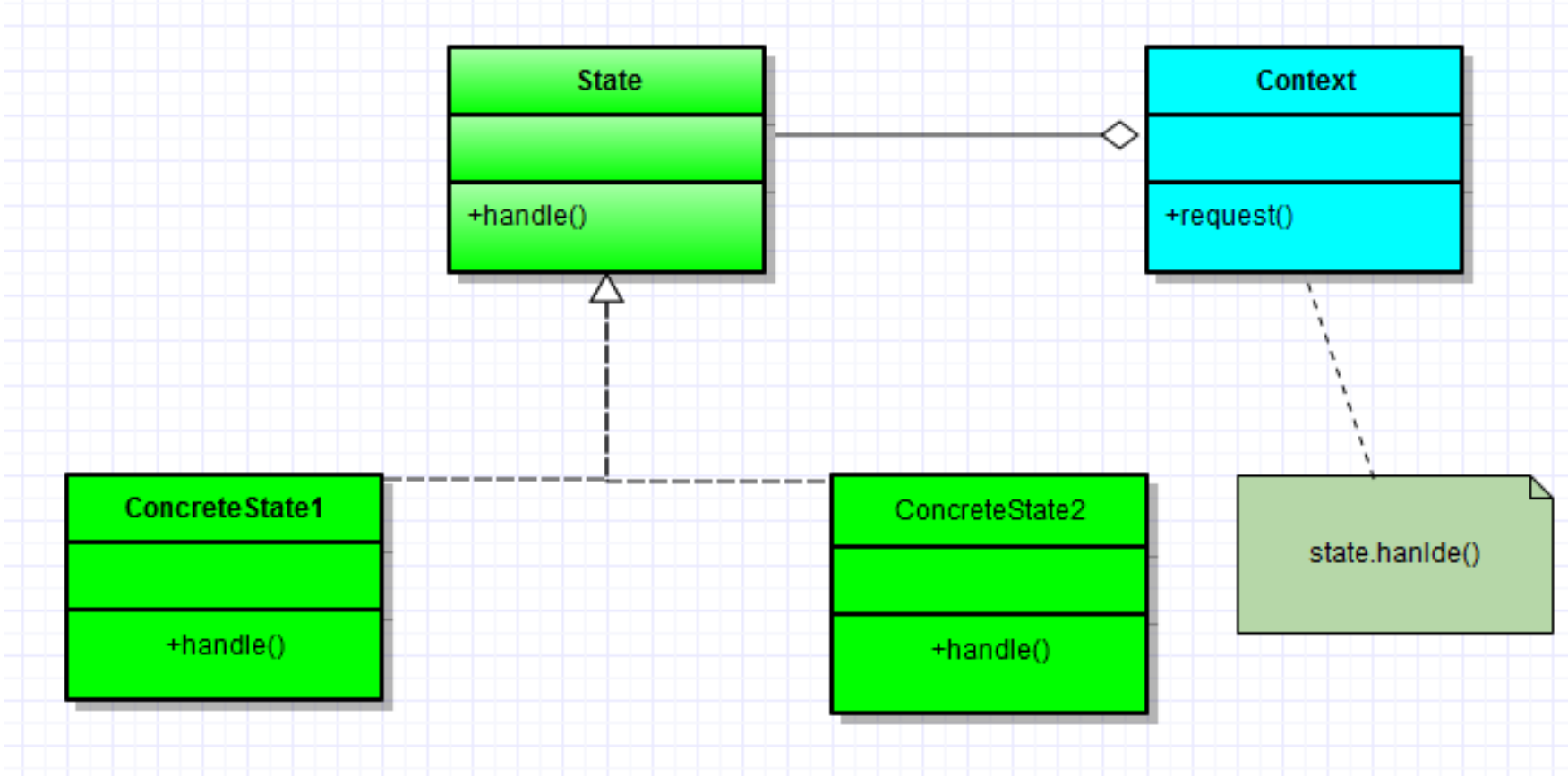
1. Observer pattern – Source code demo

2. State pattern – Definition

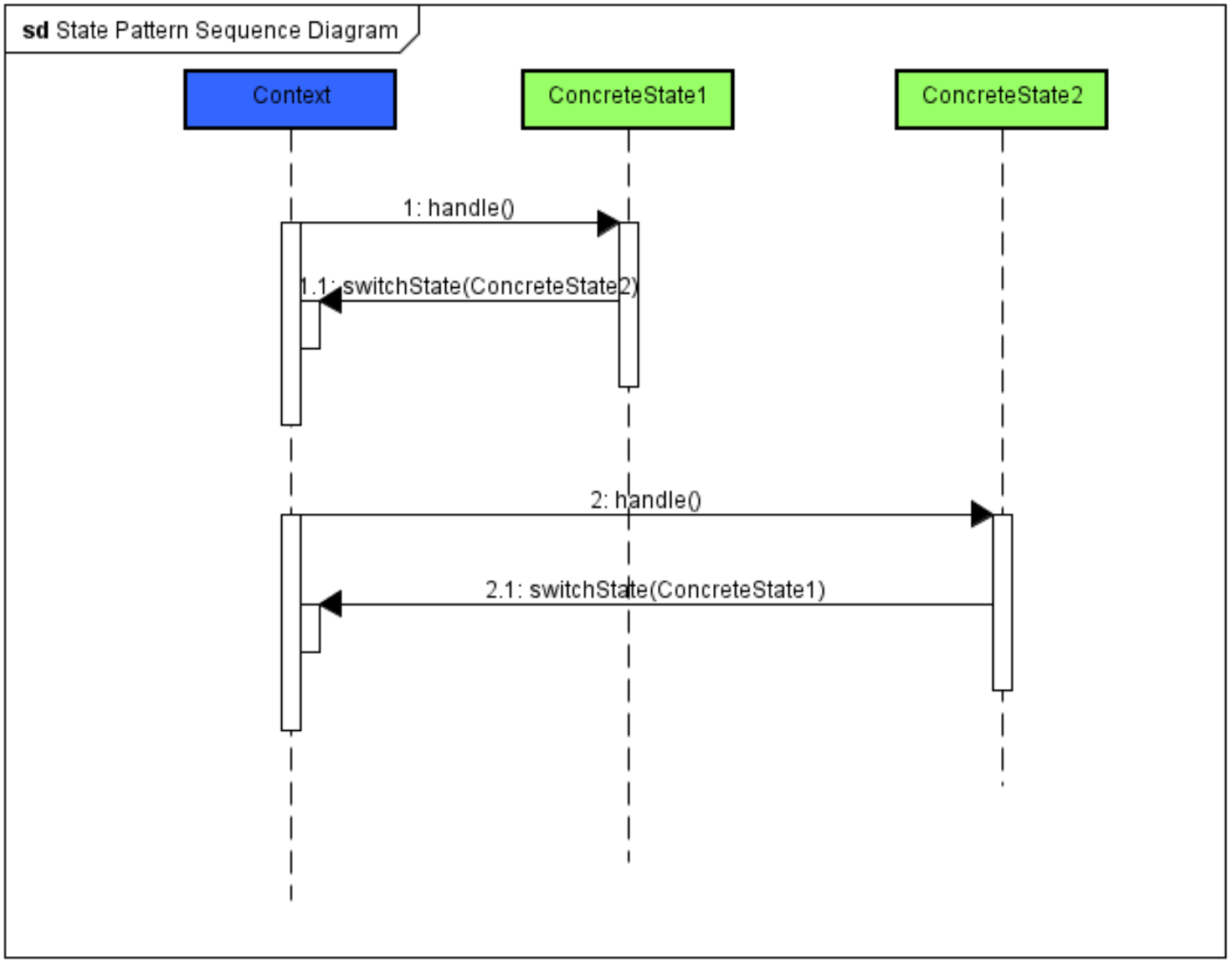
- State pattern is one of the behavioral design pattern.

- State design pattern is used when an Object changes its behavior based on its internal state.

2. State pattern – Class diagram



2. State pattern – Sequence diagram



2. State pattern – Advantages

- This makes a class independent of how state-specific behavior is implemented. New states can be added by defining new state classes.

- A class can change its behavior at run-time by changing its current state object.

2. State pattern – Disadvantages

- The number of classes grows up

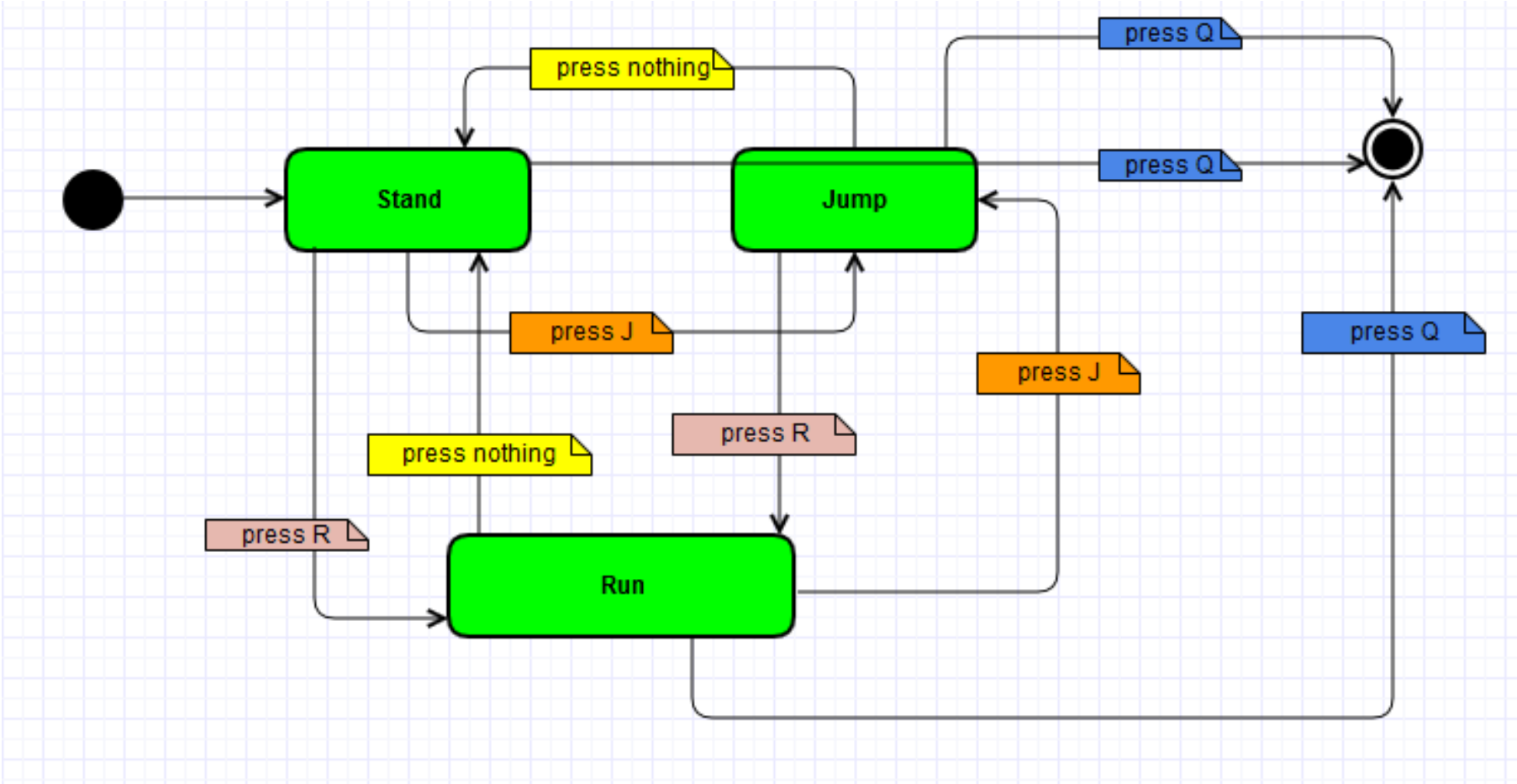
2. State pattern – Source code demo

3.*Finite State Machine - Definition*

- A *Finite State Machine* is a model of computation based on a hypothetical machine made of one or more states.

- Only one single state of this machine can be active at the same time. It means the machine has to transition from one state to another in to perform different actions.

3.*Finite State Machine - Diagram*



3.*Finite State Machine - Implementation*

- This pattern can implement by using switch case or using state pattern.
- **Where to use finite state machines**
- When your behavior can be defined by a finite set of states.
- When each state can be reached by an external or internal trigger/Action.
- Define behavior not too much complex.

4. Quiz

1. What kind of Observer parttern?
   a. *Creational* **b.** *Structural* **c.** *Behavioral* **d.** *Concurrency*
2. What kind of State parttern?
   a. *Creational* **b.** *Structural* **c.** *Behavioral* **d.** *Concurrency*
3. What is the benefit of Observer pattern?
4. What is the benefit of State pattern pattern?
5. Which are these methods that Subject class commonly have in Observer pattern?
6. How many class types need to implement in State pattern?
7. Do Observer pattern and State pattern conflict with the SOLID principles or not ? Please explain why?

End…

Thank you!