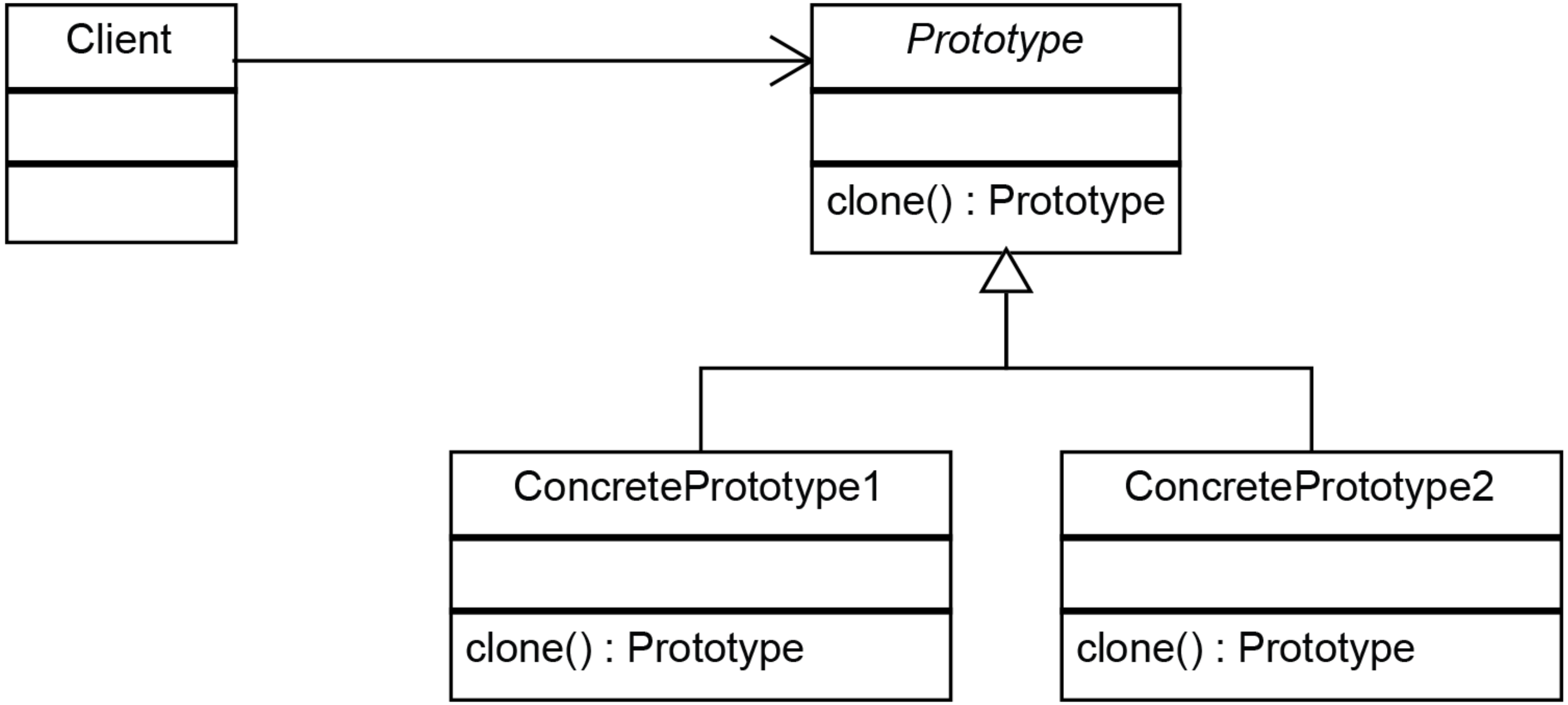
Prototype

Prototype is a creational design pattern that allows cloning objects, even complex ones, without coupling to their specific classes.

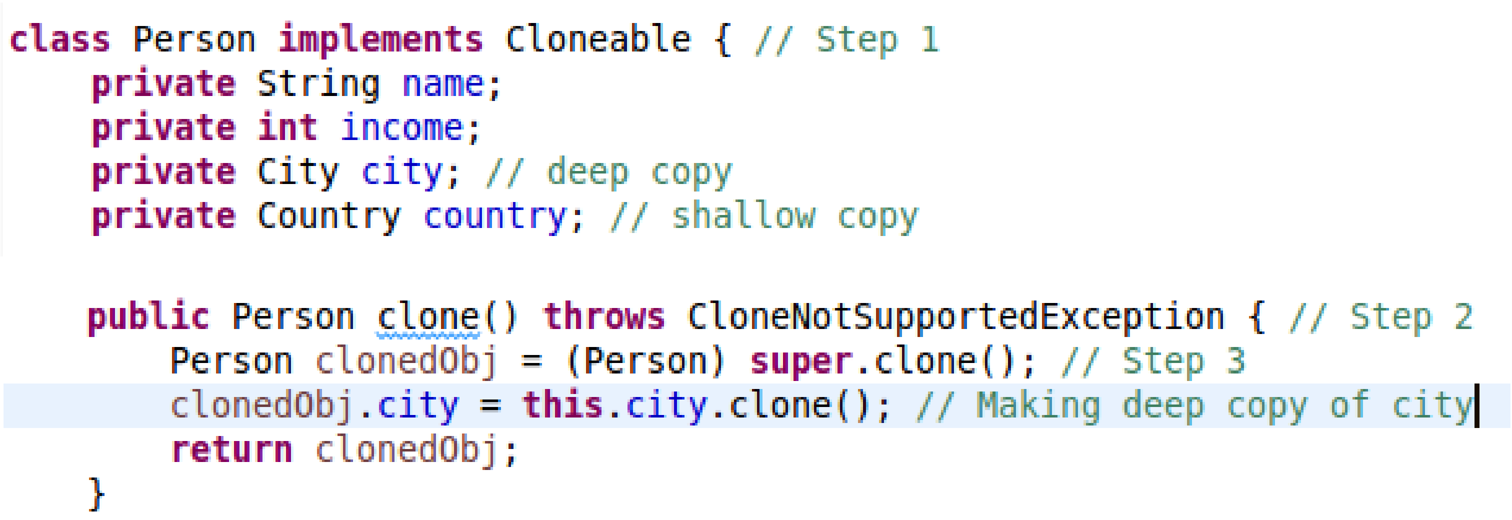
All prototype classes should have a common interface that makes it possible to copy objects even if their concrete classes are unknown.

Prototype


Client
Prototype
clone() : Prototype
ConcretePrototype1
clone() : Prototype
ConcretePrototype2
clone() : Prototype

Cloneable

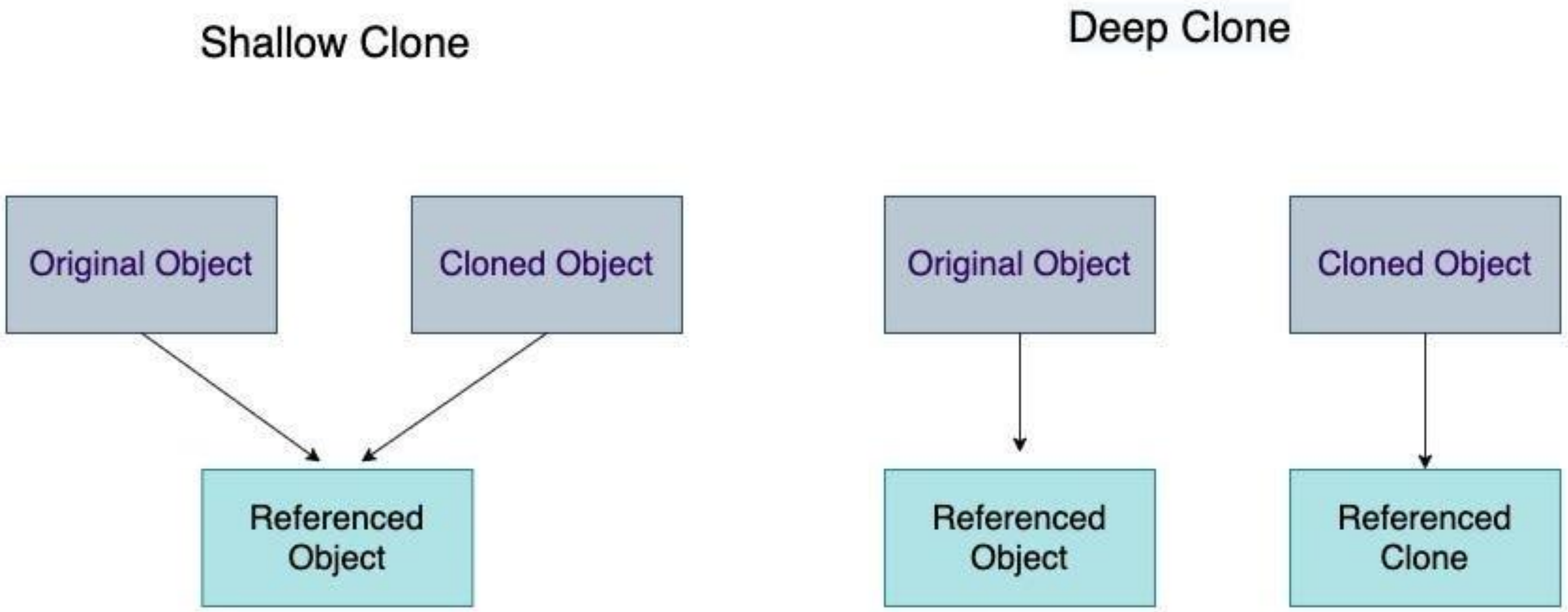
"If a class implements Cloneable, Object’s clone method returns a field-by-field copy of the object; otherwise it throws CloneNotSupportedException. This is a highly atypical use of interfaces and not one to be emulated." - J.Bloch "Effective Java"



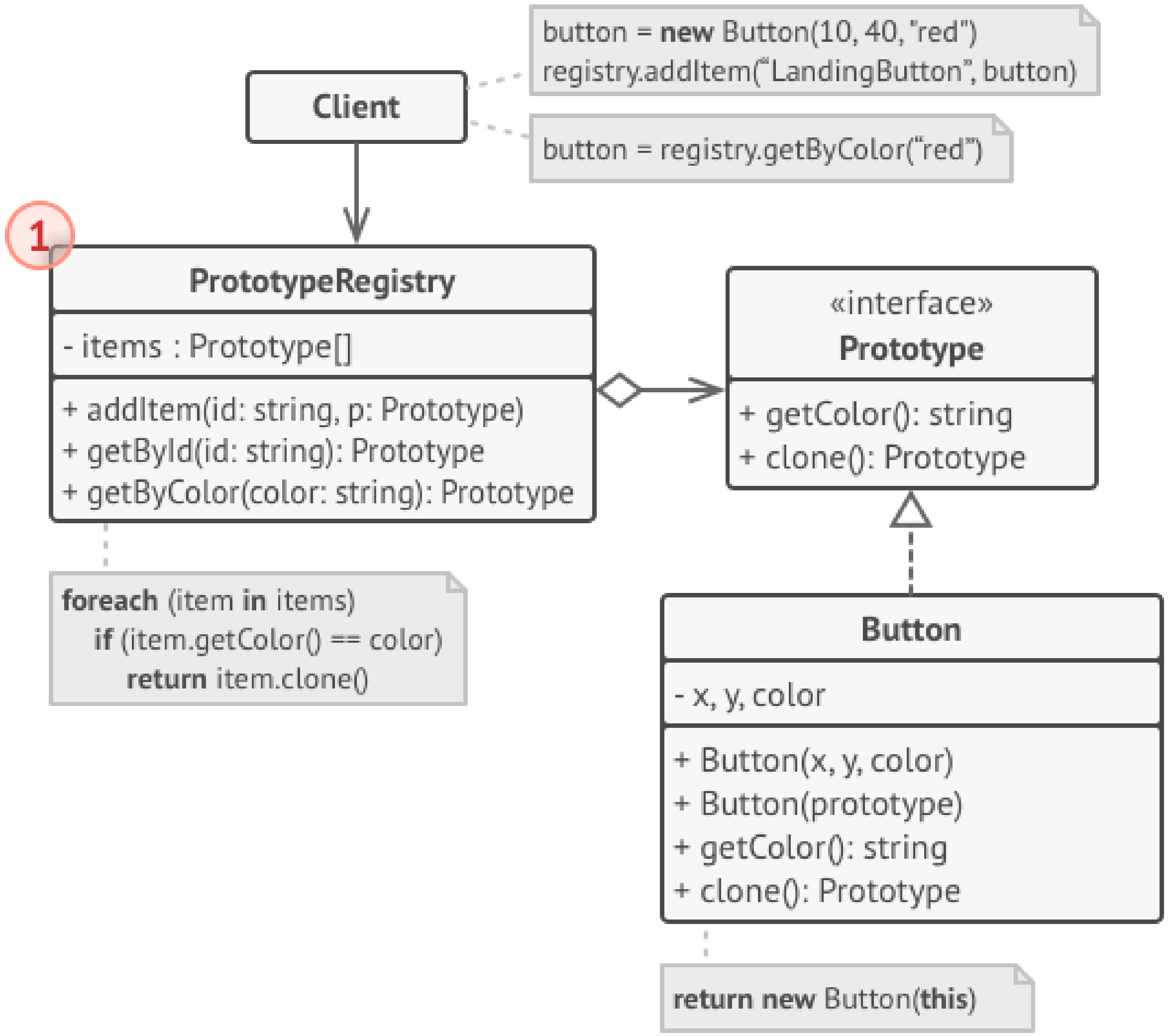
```java
class Person implements Cloneable { // Step 1
    private String name;
    private int income;
    private City city; // deep copy
    private Country country; // shallow copy

    public Person clone() throws CloneNotSupportedException { // Step 2
        Person clonedObj = (Person) super.clone(); // Step 3
        clonedObj.city = this.city.clone(); // Making deep copy of city
        return clonedObj;
    }
```

Shallow and Deep copy



Prototype Registry



Client

```
button = new Button(10, 40, "red")
registry.addItem("LandingButton", button)
```

```
button = registry.getByColor("red")
```

1

PrototypeRegistry

- items : Prototype[]

+ addItem(id: string, p: Prototype)
+ getById(id: string): Prototype
+ getByColor(color: string): Prototype

```
foreach (item in items)
    if (item.getColor() == color)
        return item.clone()
```

«interface»
Prototype

+ getColor(): string
+ clone(): Prototype

Button

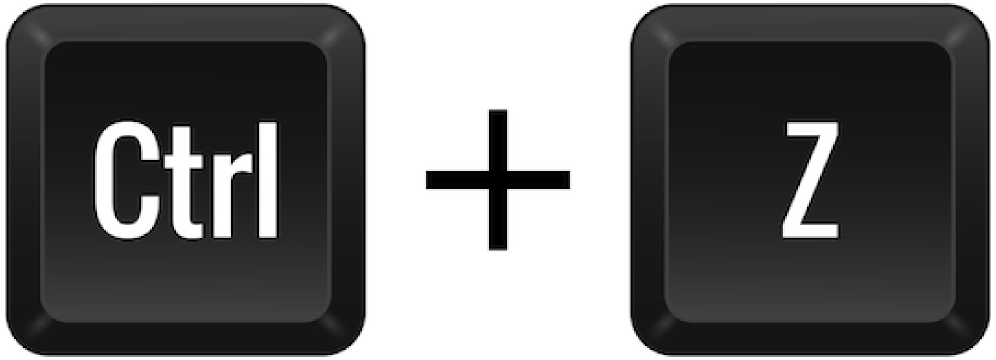
- x, y, color

+ Button(x, y, color)
+ Button(prototype)
+ getColor(): string
+ clone(): Prototype

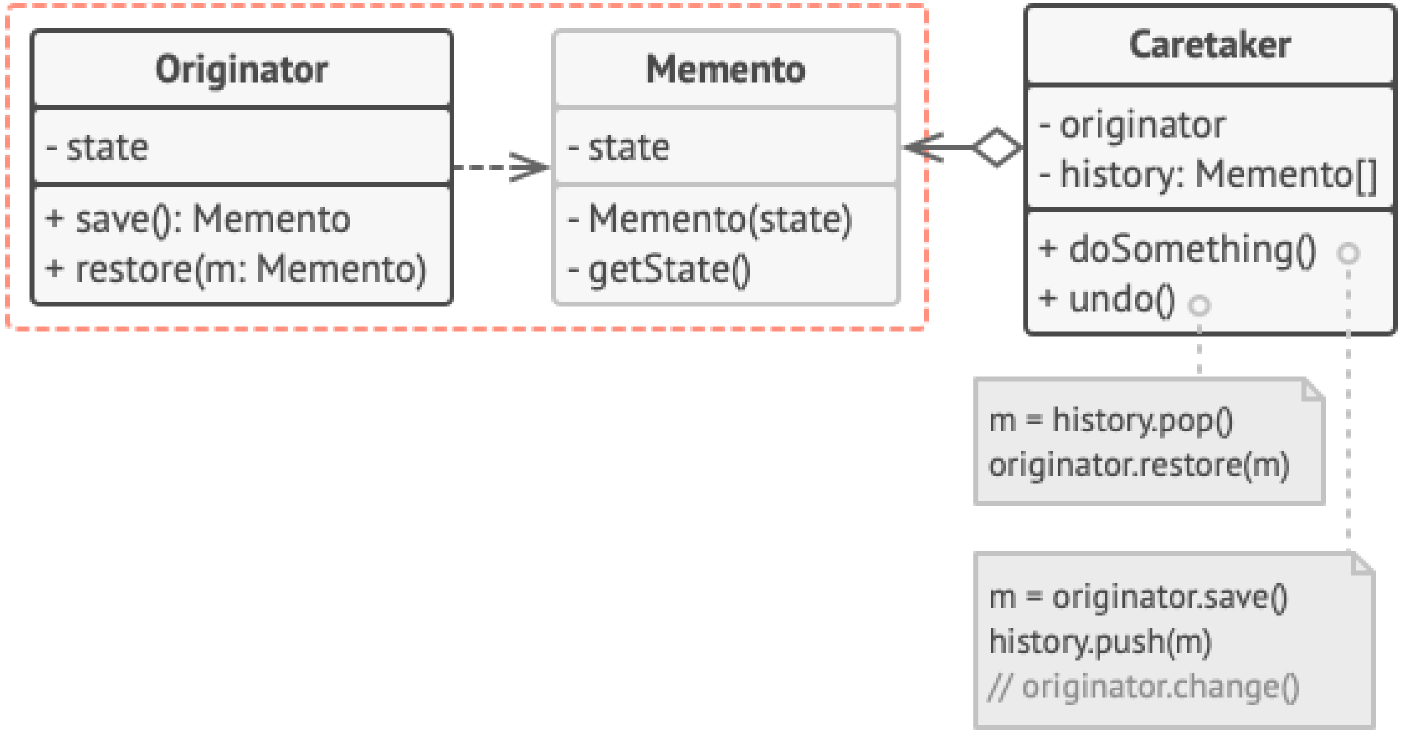
```
return new Button(this)
```

Memento

Memento is a behavioral design pattern that allows making snapshots of an object’s state and restoring it in future.



Memento



Originator
- state
+ save(): Memento
+ restore(m: Memento)
Memento
- state
- Memento(state)
- getState()
Caretaker
- originator
- history: Memento[]
+ doSomething()
+ undo()
m = history.pop()
originator.restore(m)
m = originator.save()
history.push(m)
// originator.change()