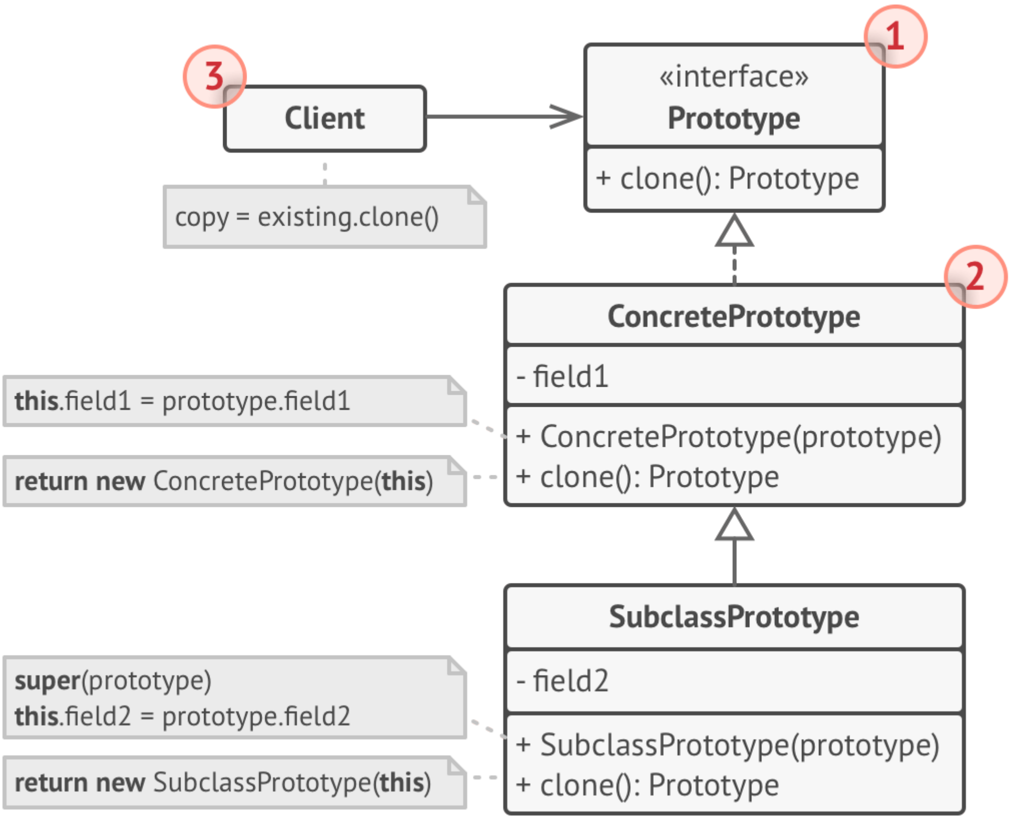
the process of mitotic cell division (biology, remember?). After mitotic division, a pair of identical cells is formed. The original cell acts as a prototype and takes an active role in creating the copy.

Structure

Basic implementation

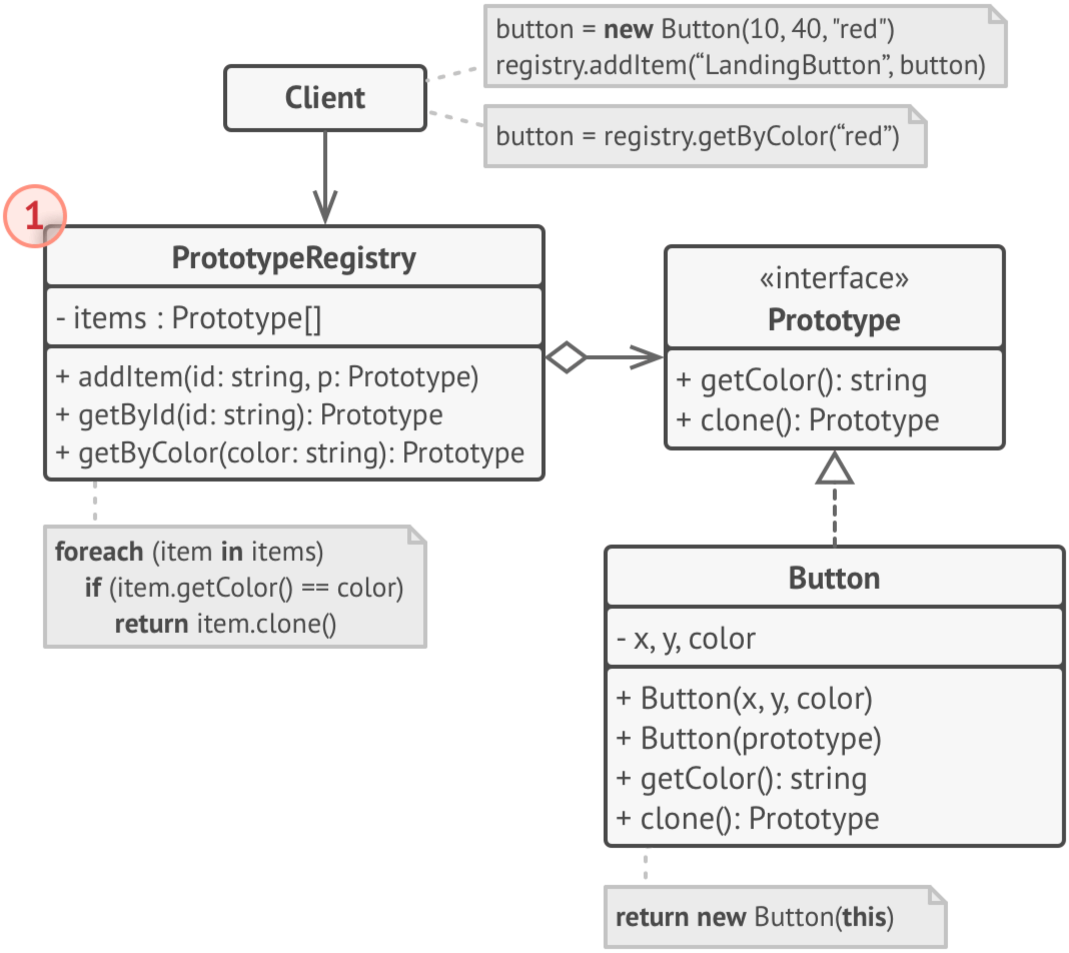

3
Client
copy = existing.clone()
1
«interface»
Prototype
+ clone(): Prototype
2
ConcretePrototype
- field1
+ ConcretePrototype(prototype)
+ clone(): Prototype
this.field1 = prototype.field1
return new ConcretePrototype(this)
SubclassPrototype
- field2
+ SubclassPrototype(prototype)
+ clone(): Prototype
super(prototype)
this.field2 = prototype.field2
return new SubclassPrototype(this)

1. The **Prototype** interface declares the cloning methods. In most cases, it's a single `clone` method.

2. The **Concrete Prototype** class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc.

3. The **Client** can produce a copy of any object that follows the prototype interface.

Prototype registry implementation


button = **new** Button(10, 40, "red")
registry.addItem("LandingButton", button)
Client
button = registry.getByColor("red")
1
PrototypeRegistry
- items : Prototype[]
+ addItem(id: string, p: Prototype)
+ getById(id: string): Prototype
+ getByColor(color: string): Prototype
«interface»
Prototype
+ getColor(): string
+ clone(): Prototype
foreach (item **in** items)
if (item.getColor() == color)
return item.clone()
Button
- x, y, color
+ Button(x, y, color)
+ Button(prototype)
+ getColor(): string
+ clone(): Prototype
return new Button(**this**)

1. The **Prototype Registry** provides an easy way to access frequently-used prototypes. It stores a set of pre-built objects that are ready to be copied. The simplest prototype registry is a `name → prototype` hash map. However, if you need better search criteria than a simple name, you can build a much more robust version of the registry.

Pseudocode

In this example, the **Prototype** pattern lets you produce exact copies of geometric objects, without coupling the code to their classes.