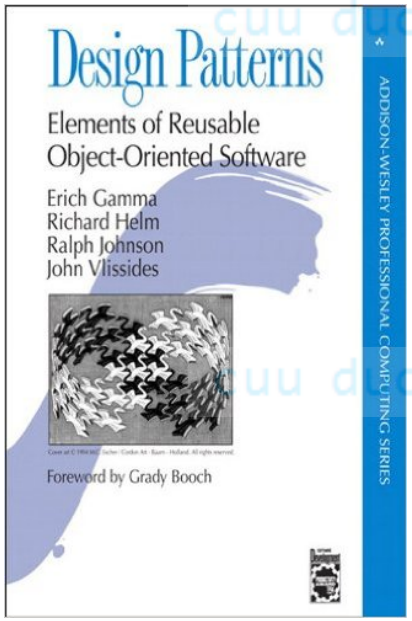
Design Patterns

nlhdung@fit.hcmus.edu.vn

What Are Design Patterns?

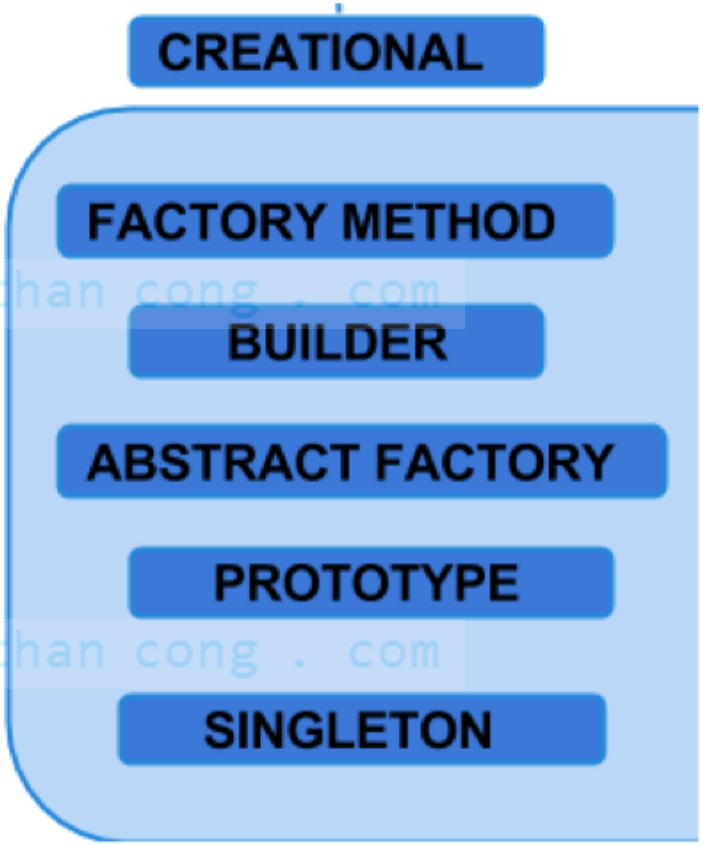
- Solutions to common problems
- Targets of refactoring, not design
- Powerful, flexible, reusable



Pattern Types

Creational Patterns

Concerned with the creation of objects and instances.

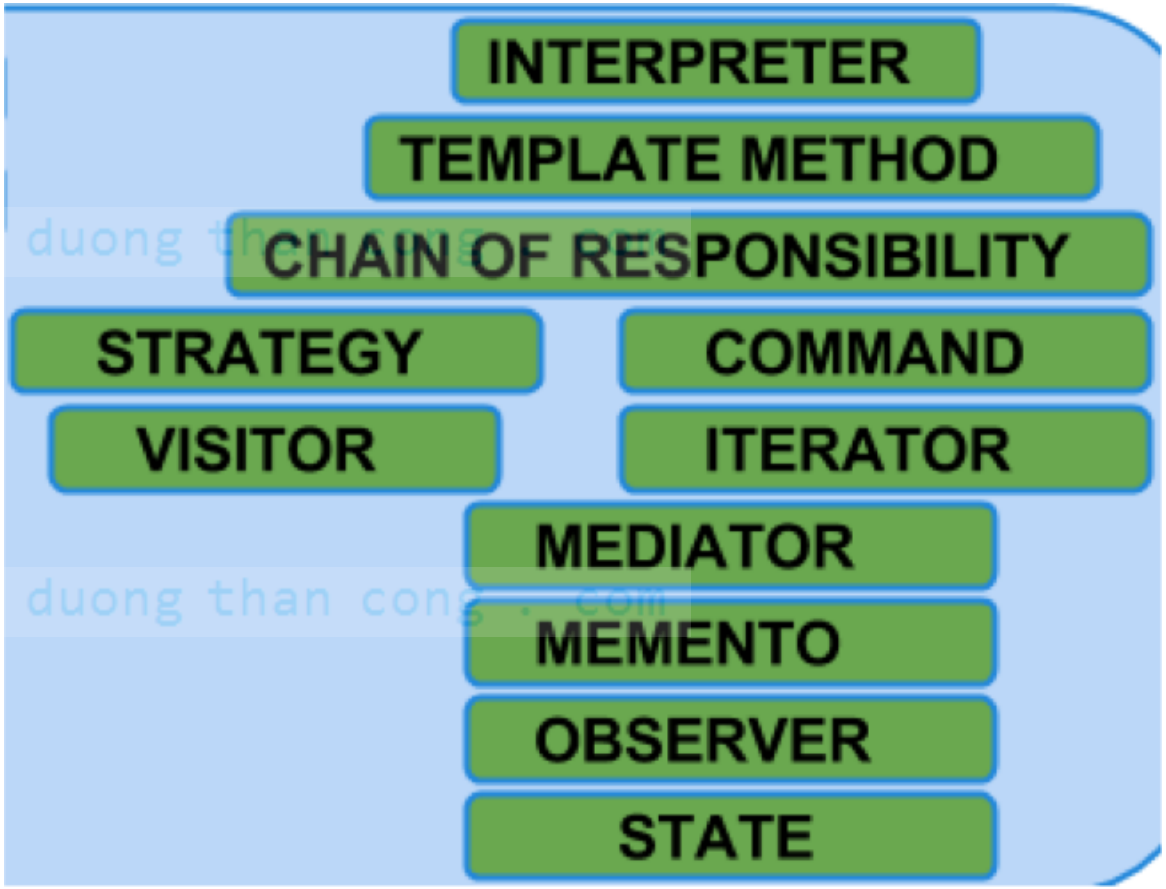


Structural Patterns

Concerned with the overall design of the system and its constituent classes and objects.

Behavioral Patterns

Concerned with the assignment of responsibilities to objects and classes.



Style for Describing Patterns

- We will use this structure:
  - *Pattern name*
  - *Purpose:* what problem the pattern addresses and the general approach of the pattern
  - *UML for the pattern*
    - *Participants:* a description as a class diagram
  - *Use Example(s):* examples of this pattern, in C# and other

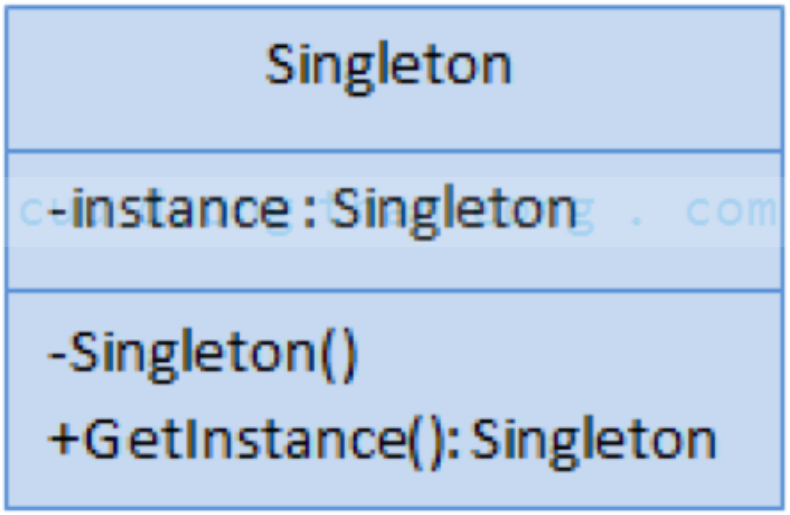
Singleton

Singleton - Purpose

- This pattern ensures that a class has only one instance and provides a global point of access to it.
  - Exactly one instance of a class is required.
  - Controlled access to a single object is necessary

cuu duong than cong . com

Singleton - UML



Singleton Pattern

Singleton – Sample 1

Eager initialization of singleton

```
1   public class Singleton
2   {
3       private static Singleton instance = new Singleton();
4       private Singleton() { }
5
6       public static Singleton GetInstance
7       {
8           get
9           {
10              return instance;
11          }
12      }
13  }
```

Singleton – Sample 2

lazy initialization of singleton

```
1   public class Singleton
2   {
3       private static Singleton instance = null;
4       private Singleton() { }
5
6       public static Singleton GetInstance
7       {
8           get
9           {
10              if (instance == null)
11                  instance = new Singleton();
12              return instance;
13          }
14      }
15  }
```

Singleton – Sample 3

Thread safe initialization of singleton

```
1   public class Singleton
2   {
3
4
5                                                    ();
6
7
8
9
10
11
12
13
14
15
16
17
18
19  }
```

Composite

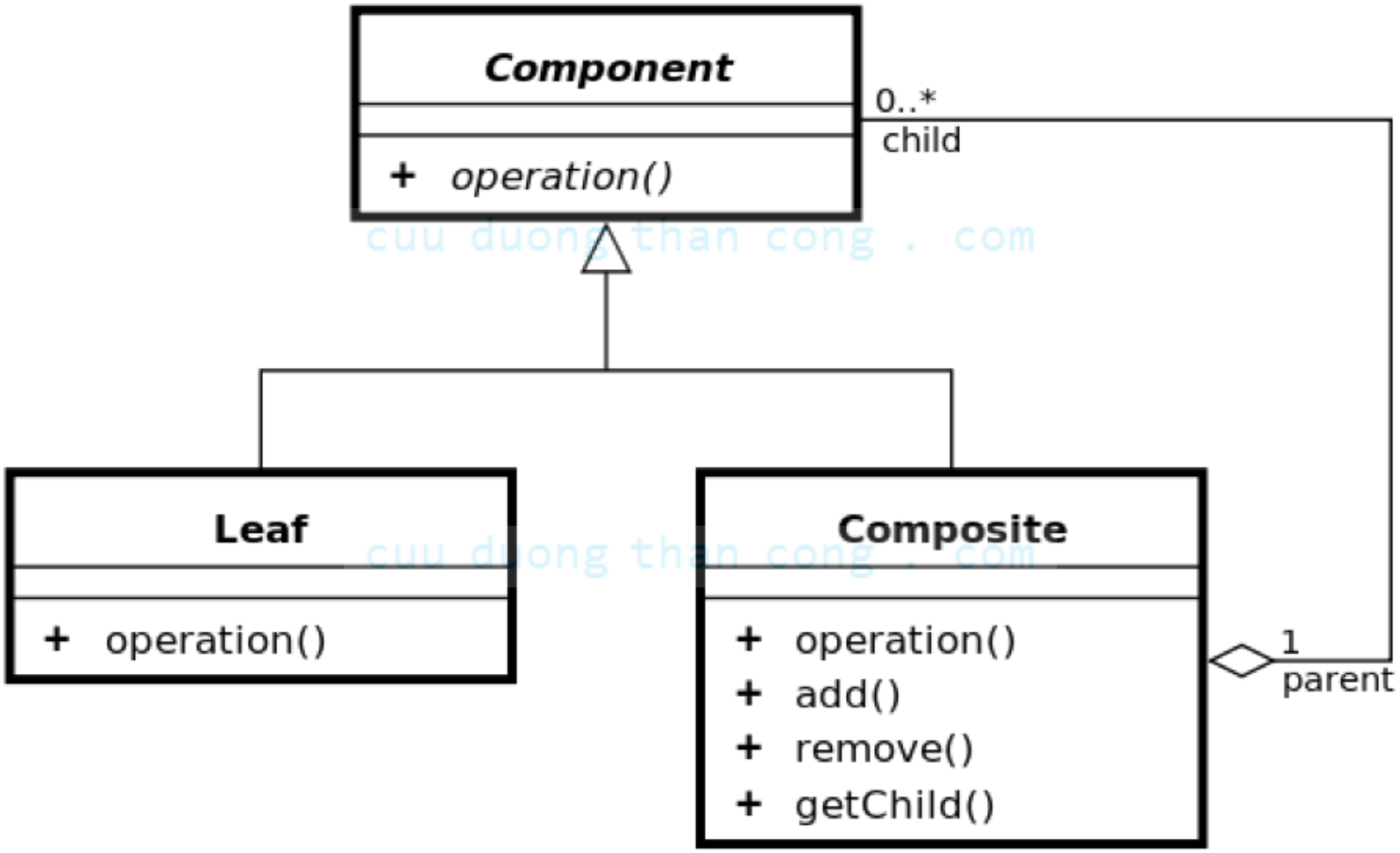
Composite - Purpose

- This pattern is used to separate an abstraction from its implementation so that both can be modified independently
- It is used when we need to treat a group of objects and a single object in the same
- It creates a class contains group of its own objects. This class provides ways to modify its group of same objects

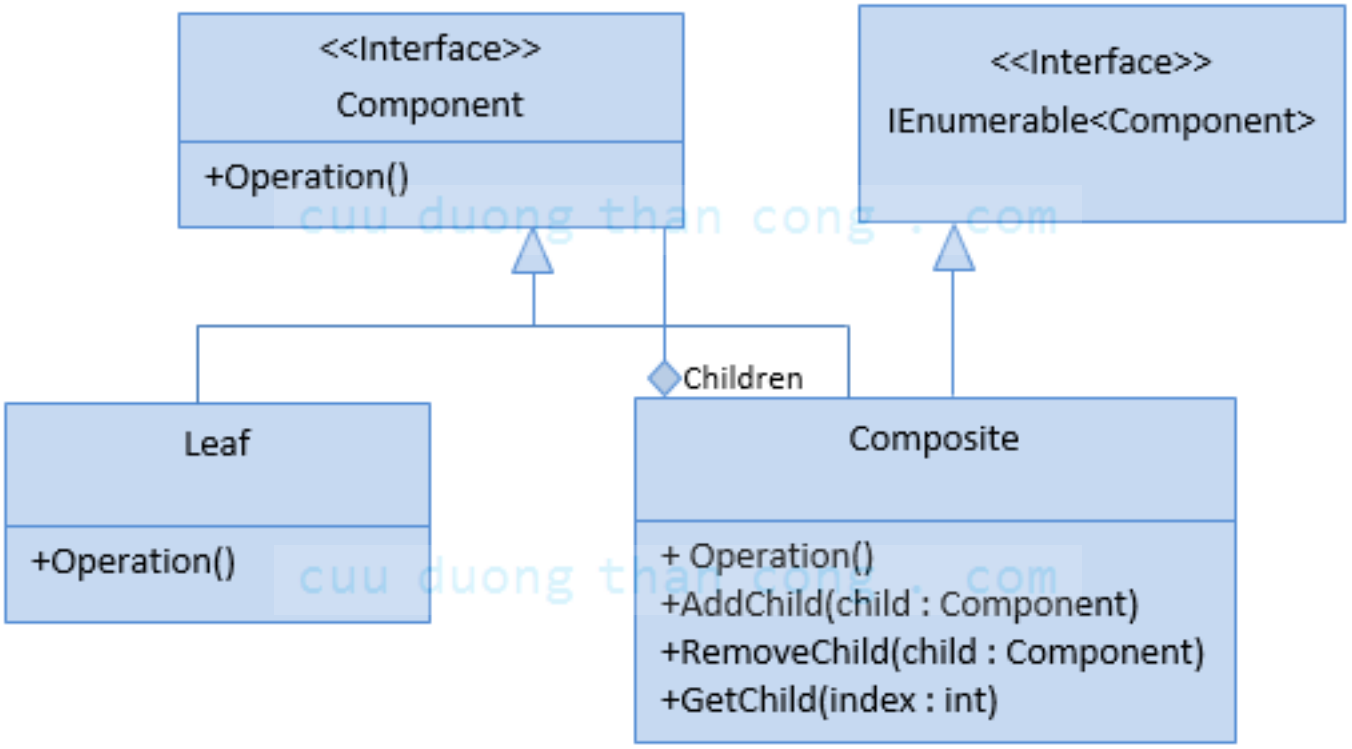
Composite – When to use

- Hierarchical representations of objects are required.
- The Composite pattern is used when data is organized in a tree structure
- A single object and a group of objects should be treated in the same way.

Composite – UML



Composite – UML (2)

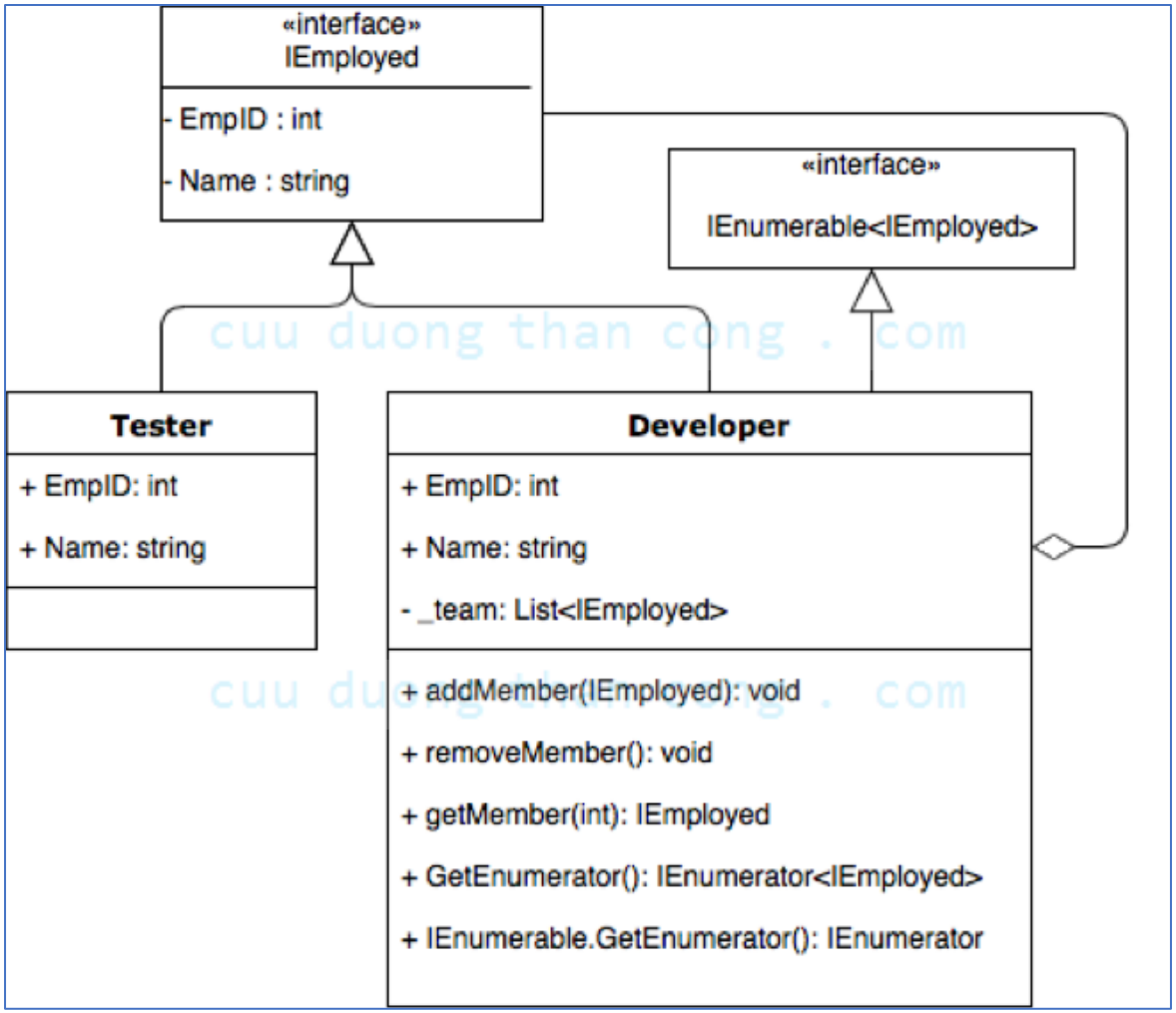


Composite Pattern

Composite – Sample

- The classes, interfaces and objects:
  - **IEmployed** - Component Interface.
  - **Employee**- Composite Class.
  - **Contractor**- Leaf Class

Composite – Sample



Composite – Sample

- Project’s source code…

Composite – Sample

```
EmpID=1, Name=Rahul

 EmpID=2, Name=Amit
          EmpID=4, Name=Rita
          EmpID=5, Name=Hari

 EmpID=3, Name=Mohan
          EmpID=6, Name=Kamal
          EmpID=7, Name=Raj
          EmpID=8, Name=Sam
          EmpID=9, Name=Tim
```

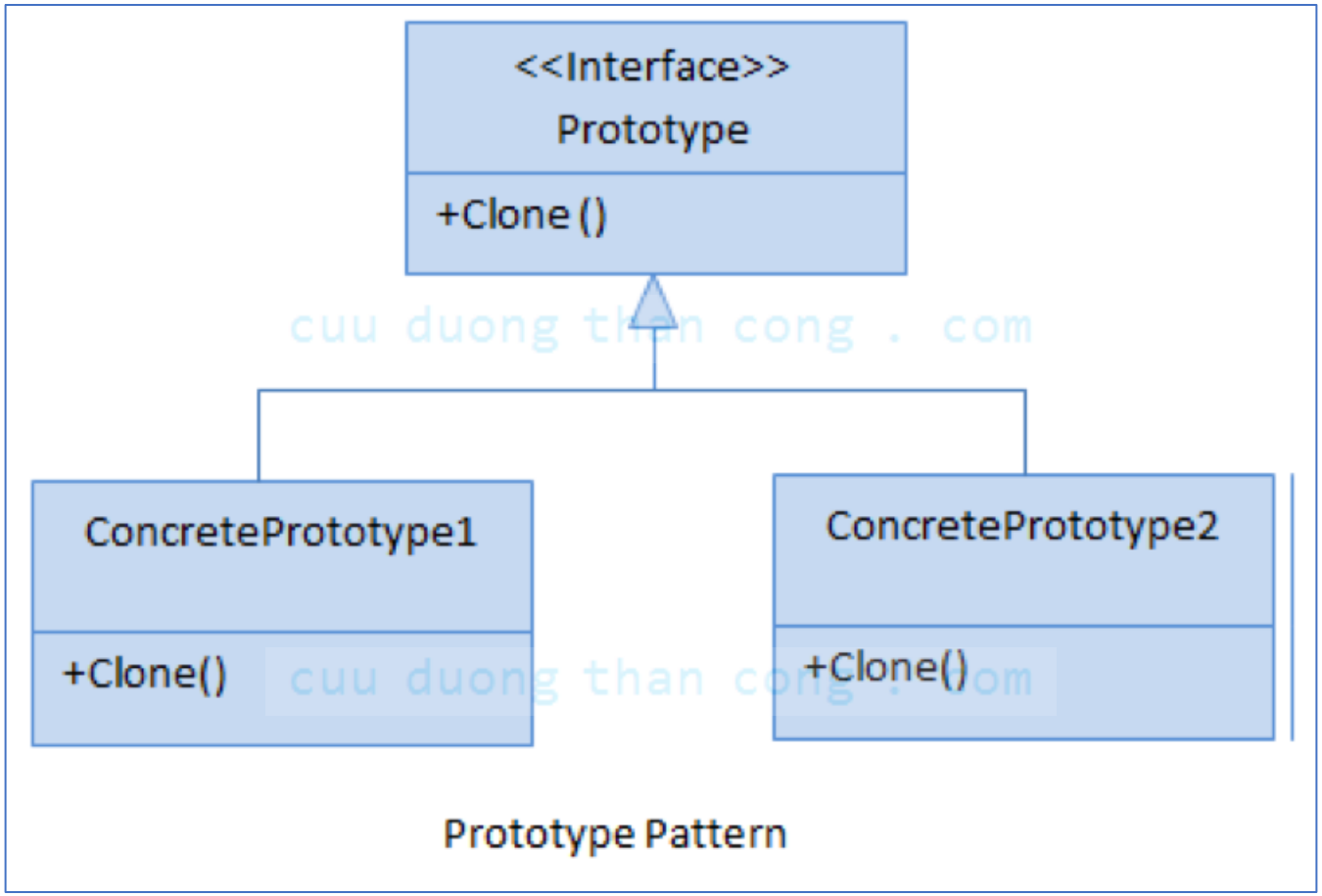
Prototype

Prototype - Purpose

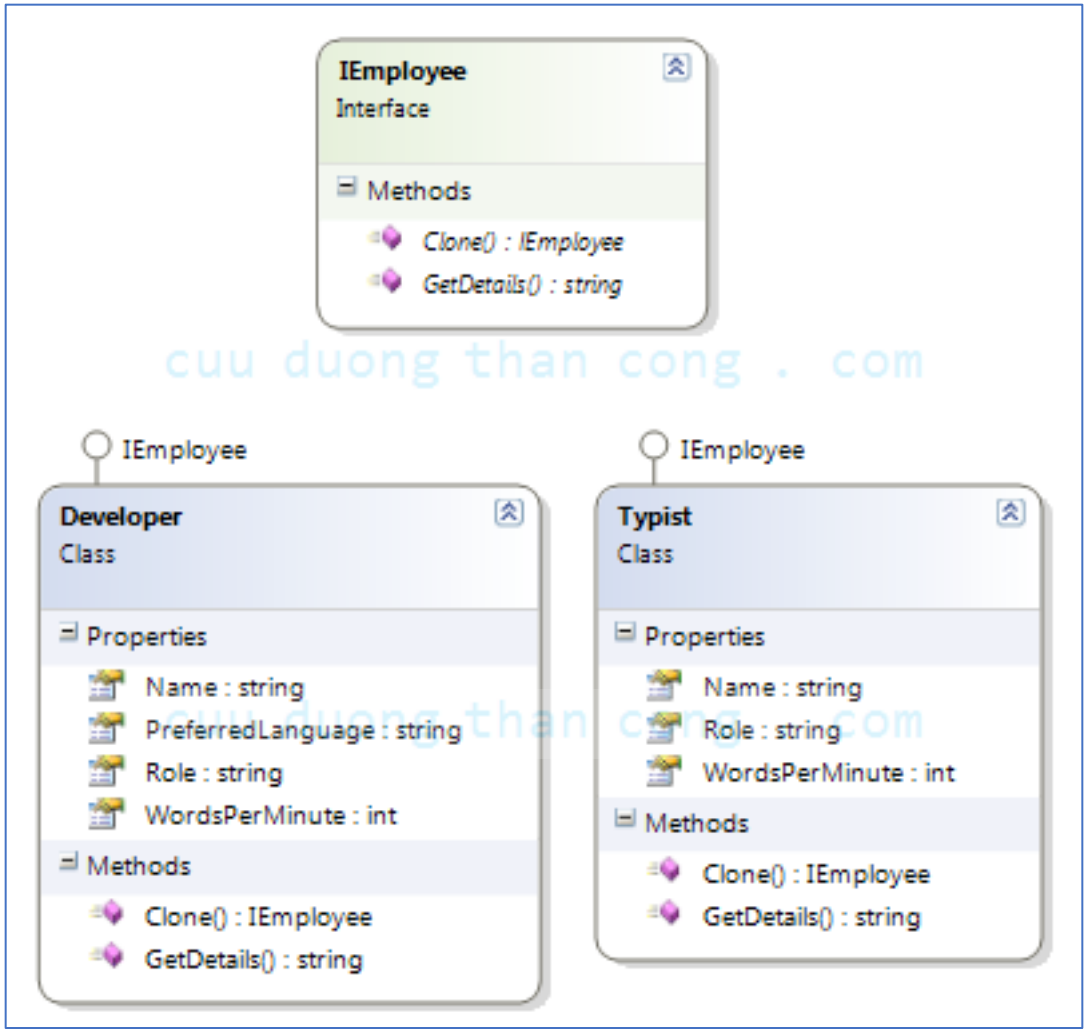
- Create objects by cloning a prototypical instance.
- Consumes less resources than creating new objects.

cuu duong than cong . com

Prototype – UML



Prototype – Sample



Prototype – Sample

- Project's source code…

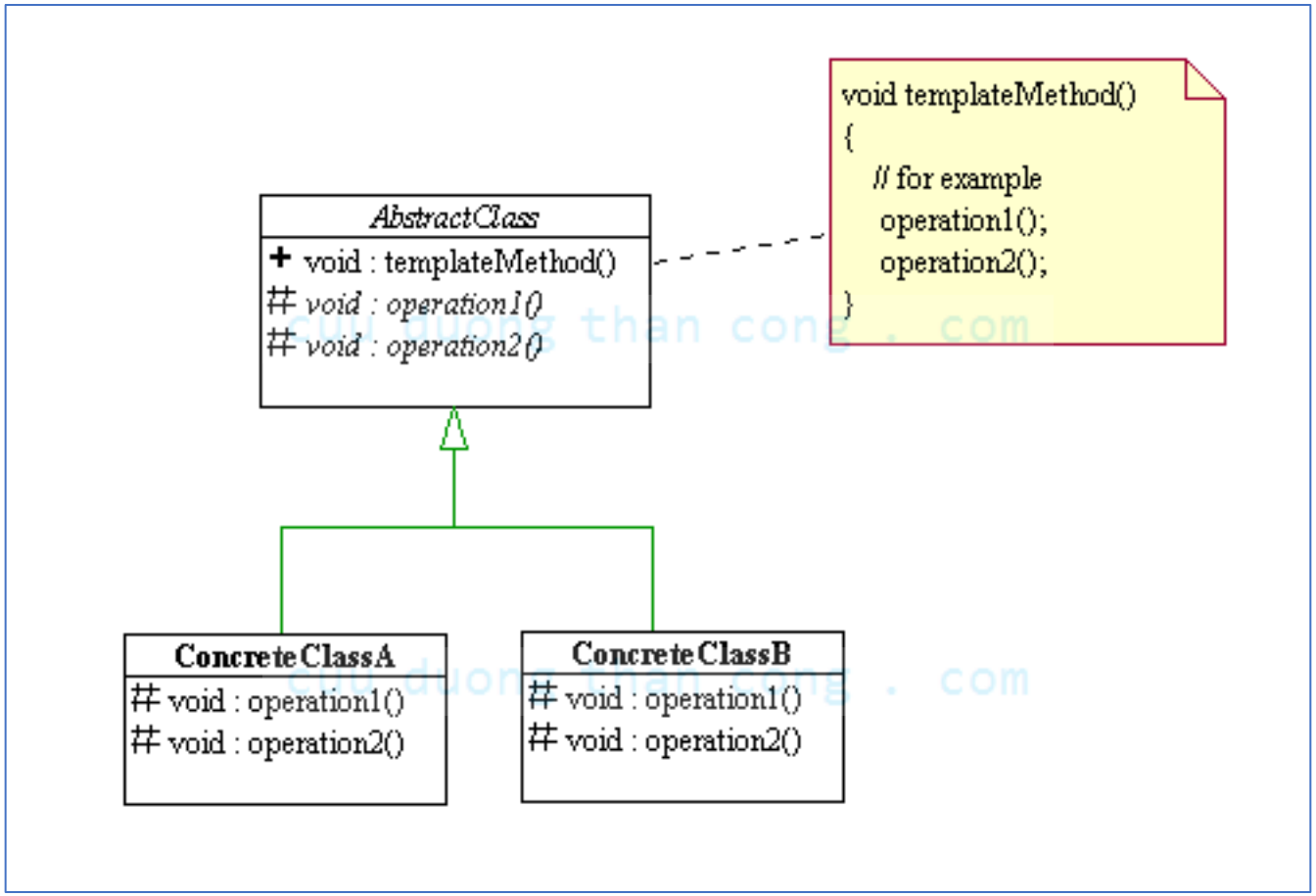
cuu duong than cong . com

Template

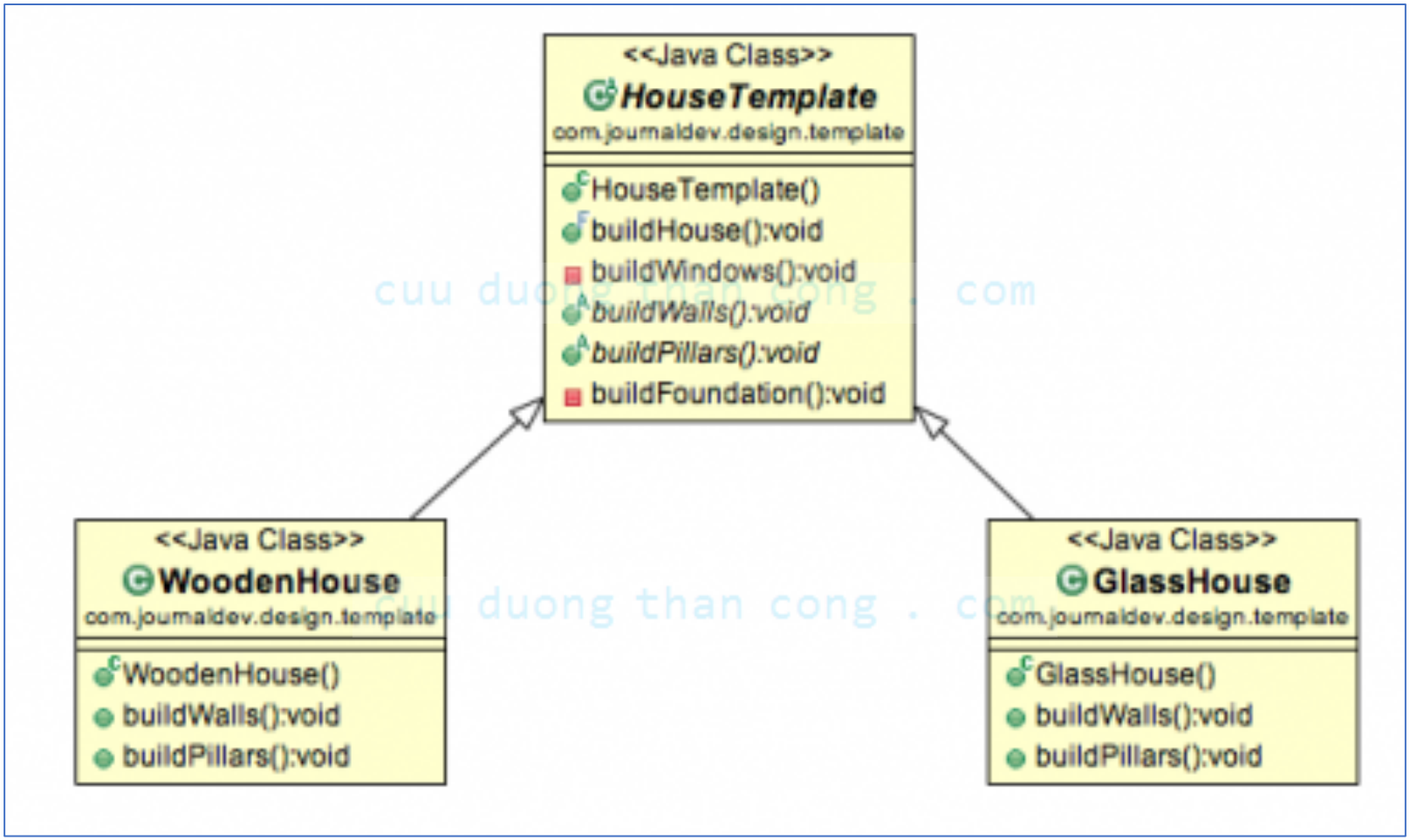
Template - Purpose

- This pattern is used to define the basic steps of an algorithm/task and allow the implementation of the individual steps to be changed.

Template – UML



Template – Sample



Prototype – Sample

- Project's source code…

cuu duong than cong . com

Questions?

Thanks!

Reference:

- Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm, “Design Patterns: Elements of Reusable Object-Oriented Software”, 1994
- Bert Bates, Kathy Sierra, Eric Freeman, Elisabeth Robson, “Head First Design Patterns”, 2009