

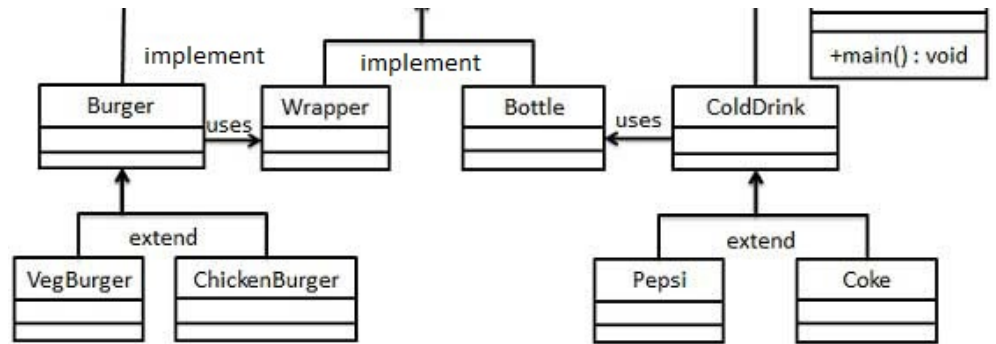
设计模式-功能-类图

2018年10月26日 星期五 下午9:42

一、创建型

new Object()

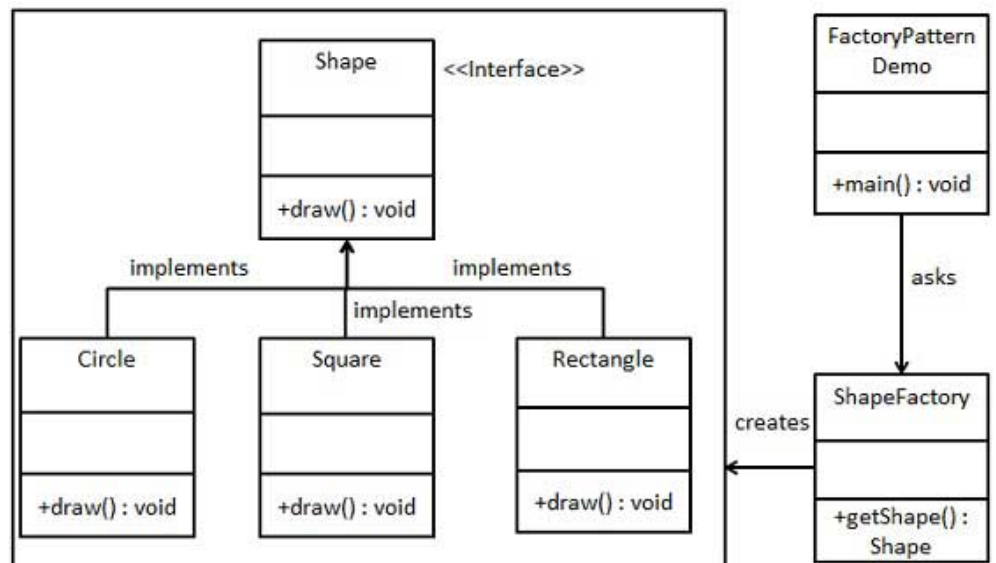
模式名称&功能	示例类图
1.1 Abstract Factory	<pre>classDiagram class AbstractFactory { +getShape() Shape +getColor() Color } class ShapeFactory { +getShape() Shape } class ColorFactory { +getColor() Color } class Shape { <<Interface>> +draw() void } class Color { <<Interface>> +fill() void } class Circle class Square class Rectangle class Red class Green class Blue class FactoryProducer { +getFactory() AbstractFactory } class AbstractFactoryPatternDemo { +main() void } AbstractFactory < -- ShapeFactory AbstractFactory < -- ColorFactory ShapeFactory --> Shape : creates ColorFactory --> Color : creates Shape < -- Circle Shape < -- Square Shape < -- Rectangle Color < -- Red Color < -- Green Color < -- Blue FactoryProducer --> AbstractFactory : uses AbstractFactoryPatternDemo --> FactoryProducer : uses</pre>
1.2 Builder	<pre>classDiagram class Item { +name() String +packing() Packing +price() float } class Meal { -items ArrayList<Item> +name() String +packing() Packing +price() float } class Packing { } class MealBuilder { +prepareVegMeal() Meal +prepareNonVegMeal() Meal } class BuilderPatternDemo { } Item < -- Meal Meal --> Item : uses Meal --> Packing : implement MealBuilder --> Meal : builds BuilderPatternDemo --> MealBuilder : asks</pre>



模式名称&功能

示例类图

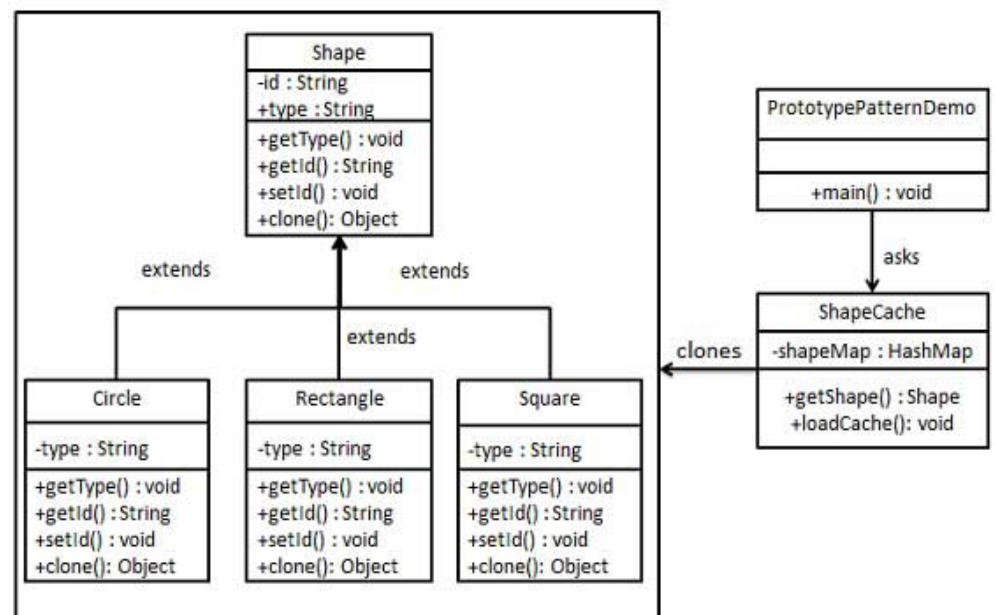
1.3 Factory Method



模式名称&功能

示例类图

1.5 Prototype



模式名称&功能	示例类图
1.6 Singleton	<pre>classDiagram class SingletonPatternDemo { +main() : void } class SingleObject { -instance: SingleObject -SingleObject() +getInstance(): SingleObject +showMessage(): void } SingletonPatternDemo --> SingleObject : asks SingleObject --> SingletonPatternDemo : returns</pre>

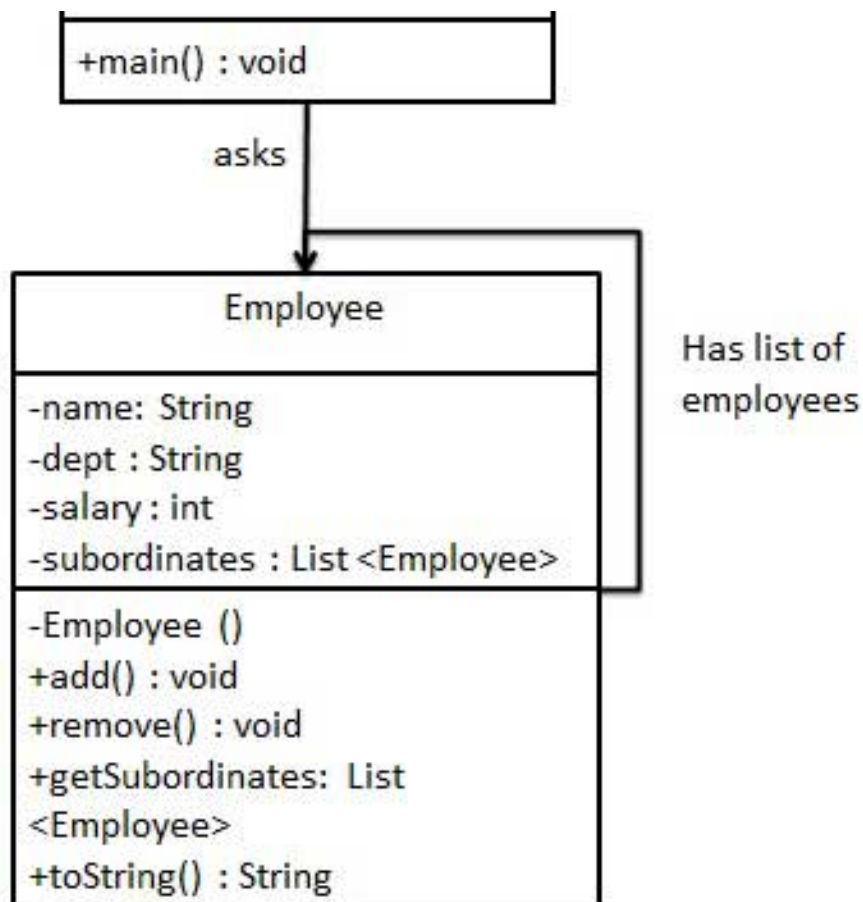
二、结构型

模式名称&功能	示例类图
2.1 Proxy <ul style="list-style-type: none">• 一键收获• 兑换券	<pre>classDiagram class Image { <<Interface>> +display() : void } class RealImage { +fileName : String +RealImage() +display() : void +loadFromDisk() : void } class ProxyImage { +realImage : RealImage +fileName : String +ProxyImage() +display() : void } class ProxyPatternDemo { +main() : void } Image < -- RealImage Image < -- ProxyImage ProxyPatternDemo --> ProxyImage : asks</pre>

模式名称&功能	示例类图
<div>2.2 Flyweight</div> <ul style="list-style-type: none">• 主地• 地砖• 摆放家具	<pre>classDiagram class Shape { <<Interface>> +draw() void } class Circle { -x, y, radius :int -color : String +Circle() +setX() void +setY() void +setRadius()void +draw() void } class FlyWeightPatternDemo { +main() void -getRandomColor String -getRandomX() int -getRandomY() int } class ShapeFactory { -circleMap HashMap +getCircle() Shape } Shape < -- Circle ShapeFactory --> Circle : creates FlyWeightPatternDemo --> ShapeFactory : asks</pre>
模式名称&功能	示例类图
<div>2.3 Decorator</div> <ul style="list-style-type: none">• 建筑物升级，装饰	<pre>classDiagram class Shape { <<interface>> +draw() void } class Circle { +draw() void } class Rectangle { +draw() void } class ShapeDecorator { +shape : Shape +ShapeDecorator() +draw(): void +setRedBorder() void } class RedShapeDecorator { +shape : Shape +RedShapeDecorator() +draw(): void -setRedBorder() void } class DecoratorPatternDemo { +main() void } Shape < -- Circle Shape < -- Rectangle ShapeDecorator < -- RedShapeDecorator ShapeDecorator --> Shape : decorates DecoratorPatternDemo --> ShapeDecorator : asks</pre>
模式名称&功能	示例类图
<div>2.4 Composit</div> <ul style="list-style-type: none">• 树结构（重新设• ...	<pre>classDiagram class CompositePatternDemo</pre>

计)

- 部门
- 宝箱

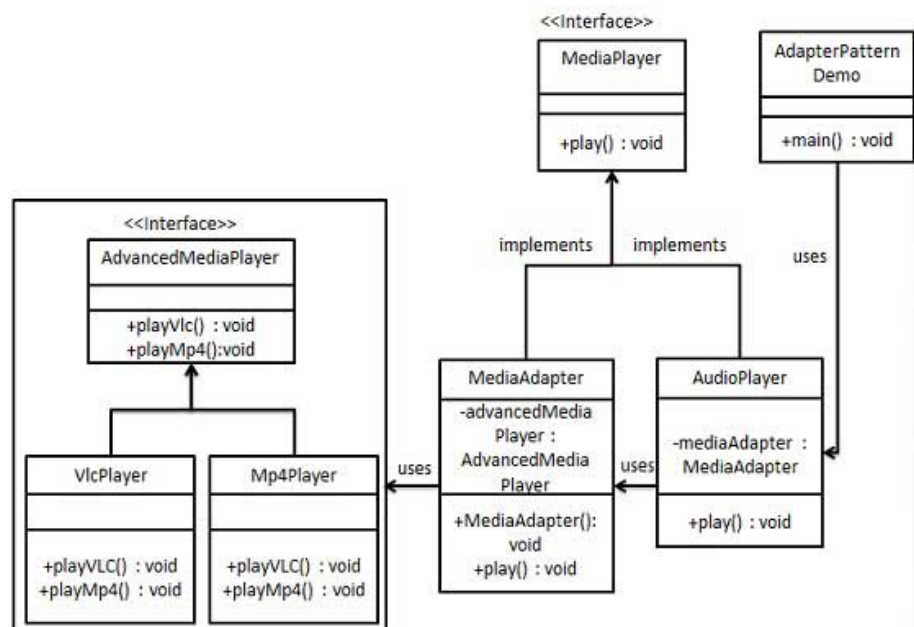


模式名称&功能

示例类图

2.5 Adaptor

- 土地适应种子 (状态)



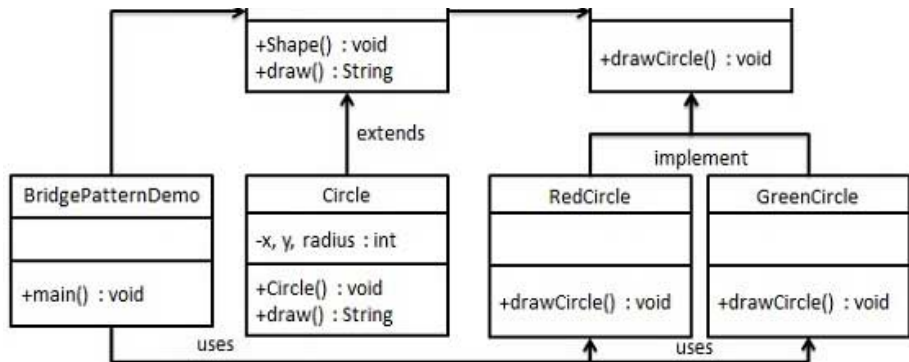
模式名称&功能

示例类图

2.6 Bridge

- 工具 . 功能



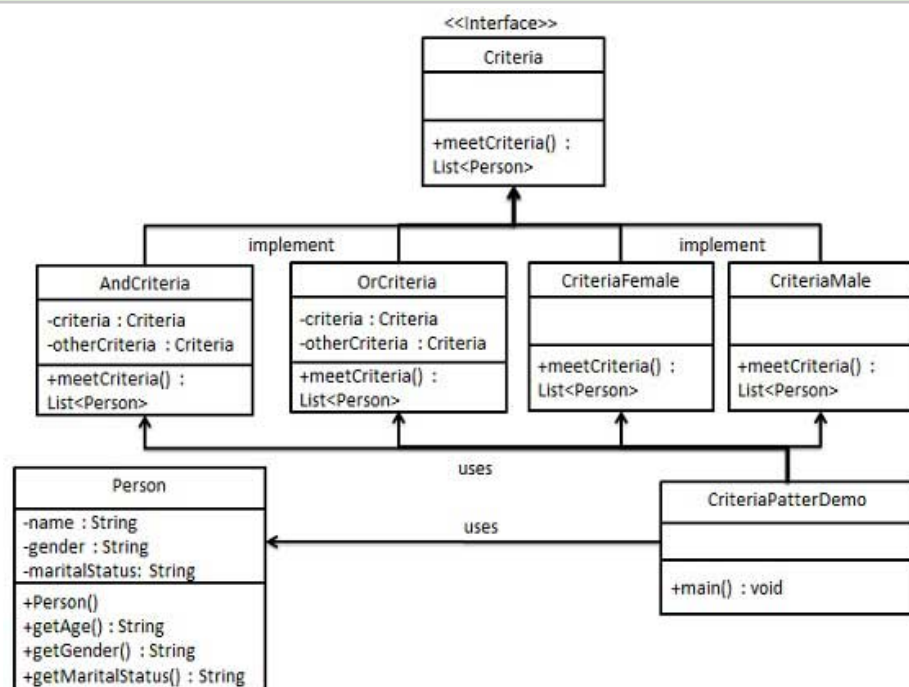


模式名称&功能

示例类图

2.7 Filter

- 安排雇员工作
- 农副产品挑选

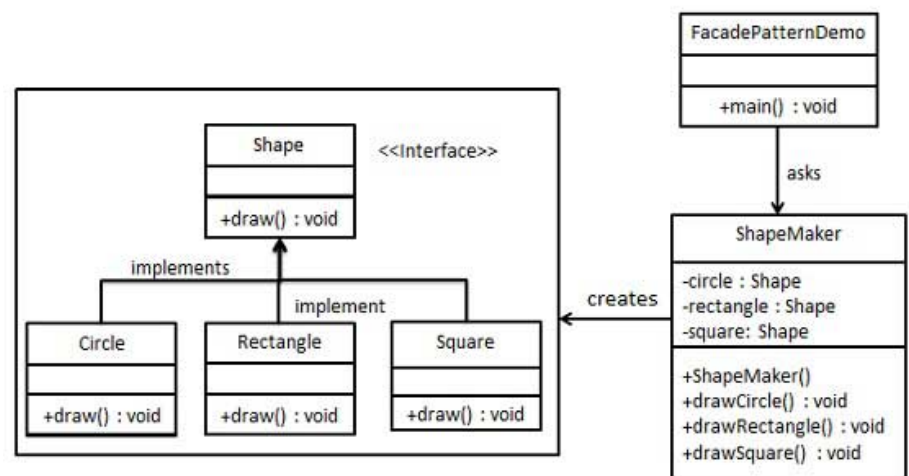


模式名称&功能

示例类图

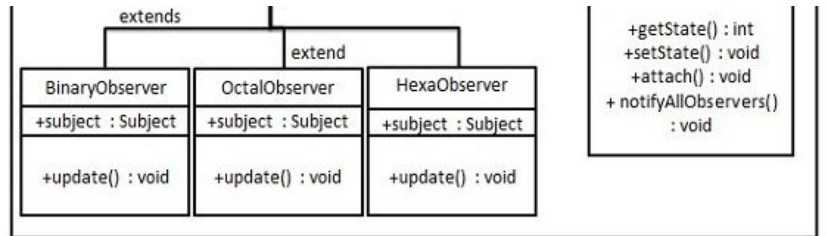
2.8 Facade

- 安保系统
- 子系统：加工农副产品



三、行为型

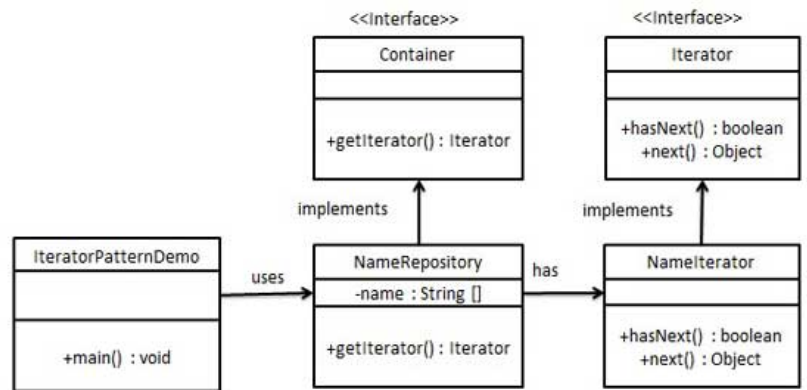
模式名称&功能	示例类图
<h3>3.1 Strategy</h3> <ul style="list-style-type: none">工具，农副产品自己使用/出售	
模式名称&功能	示例类图
<h3>3.2 Template Method 操作</h3> <p>(0) 1. 选择操作对象 2. 选择操作 3. 操作完成</p> <p>(1) 1. 检查成熟状态 2. 收获 3. 放入仓库</p> <p>(2) 收割 : 2.1 选择工具 2.2</p> <p>.....</p>	
模式名称&功能	示例类图
<h3>3.3 Observer</h3> <ul style="list-style-type: none">雇员（看留言板）	



模式名称&功能

示例类图

3.4 Iterator

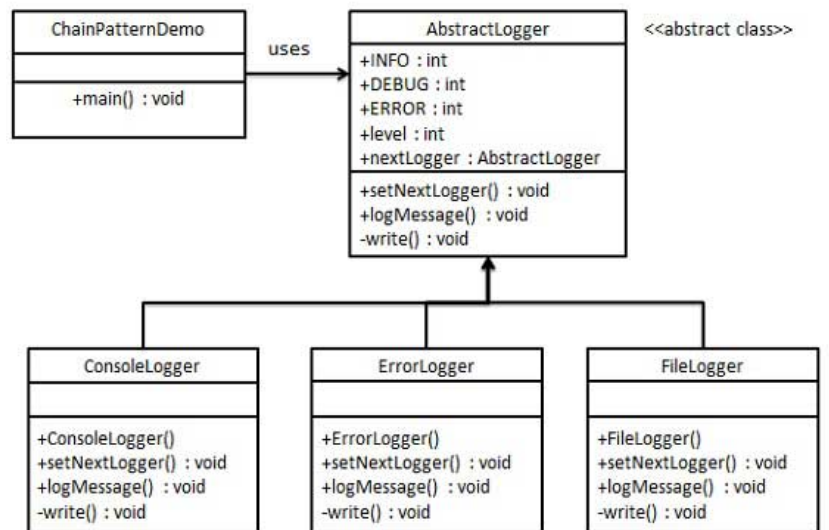


模式名称&功能

示例类图

3.5 责任链

- 兑换券

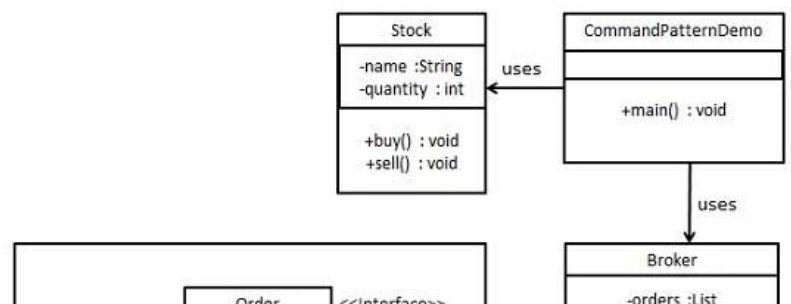


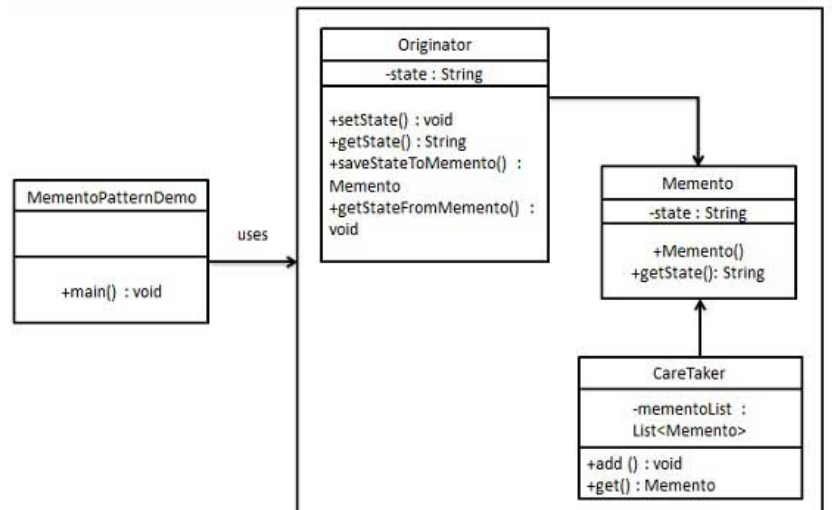
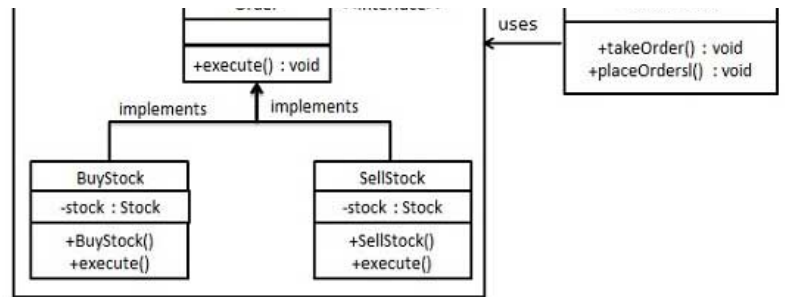
模式名称&功能

示例类图

3.6 Command 命令模式+3.7 Memento

- 装饰农场undo&redo



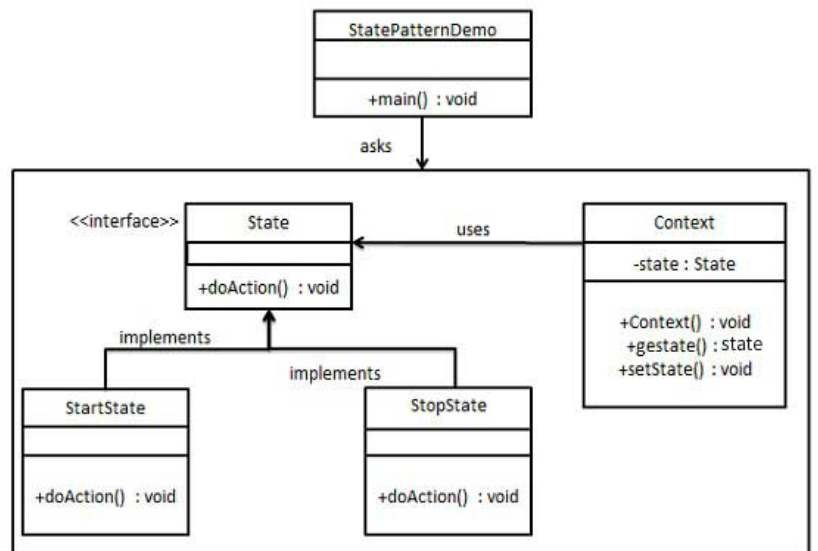


模式名称&功能

示例类图

3.8 State状态

- 农作物状态

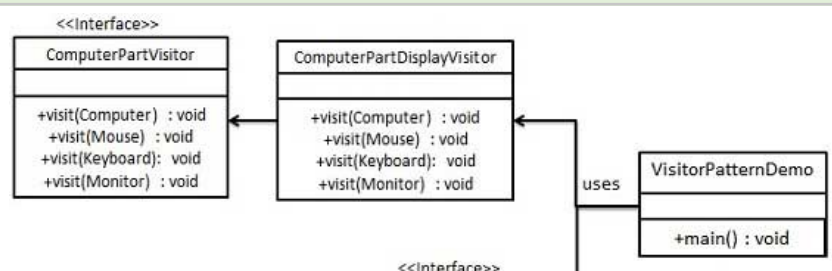


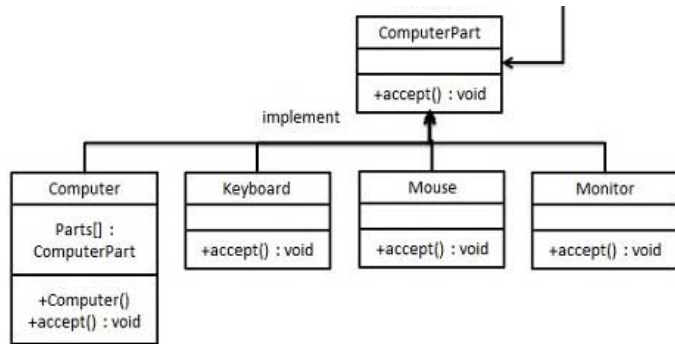
模式名称&功能

示例类图

3.9 Visitor

- 节日活动



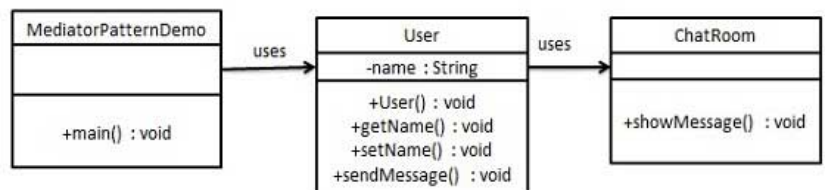


模式名称&功能

示例类图

3.10 Mediator 中介者

- 留言板、农作物工作日志

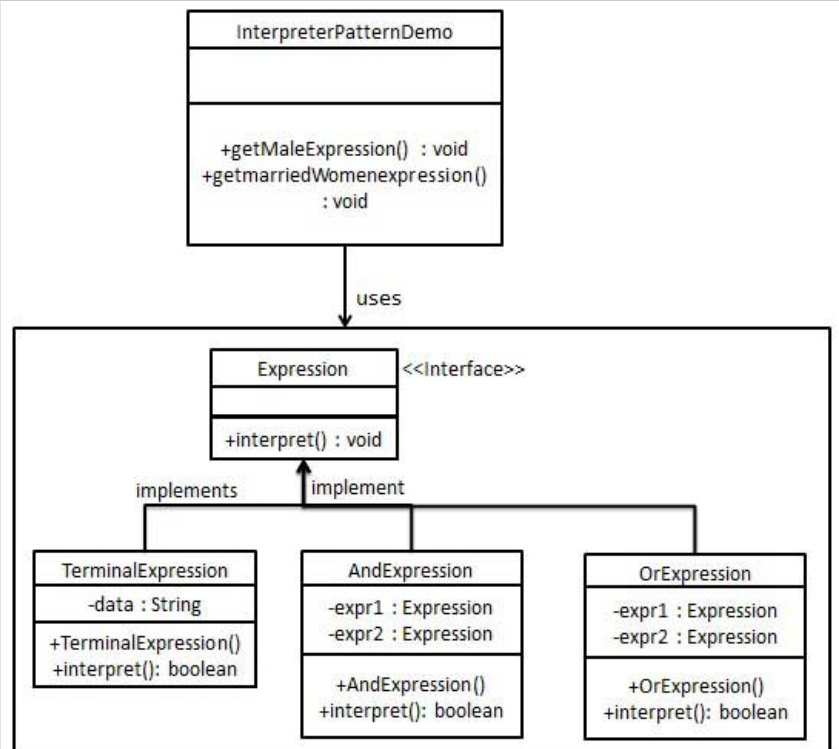


模式名称&功能

示例类图

3.11 解释器

- 计算器

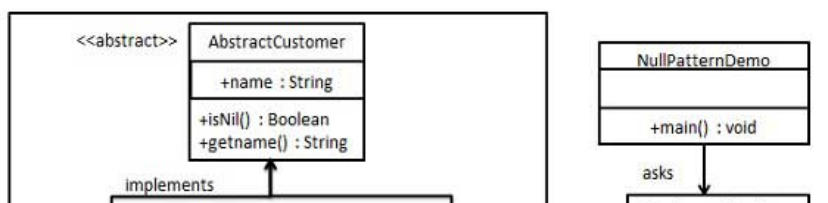


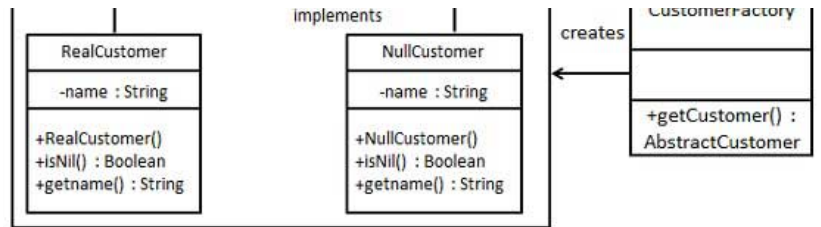
模式名称&功能

示例类图

3.12 Null Object 空对象模式

- No Command





四、其他

模式名称&功能	示例类图
4.3 Composite Entity 组合实体模式 • 屋子	<pre> classDiagram class Client { -entity: CompositeEntity +printData() : void +setData() : void } class CompositeEntityPatternDemo { +main() : void } class CompositeEntity { -cgo: CoarseGrainedObject +getData() : String[] +setData() : void } class CoarseGrainedObject { -object1 : DependentObject1 -object2 : DependentObject2 +getData() : String[] +setData() : void } class DependentObject1 { -data : String +getData() : String +setData() : void } class DependentObject2 { -data : String +getData() : String +setData() : void } Client --> CompositeEntity : uses CompositeEntityPatternDemo --> Client : uses CompositeEntity --> CoarseGrainedObject : uses CoarseGrainedObject --> DependentObject1 : uses CoarseGrainedObject --> DependentObject2 : uses </pre>
模式名称&功能	示例类图
Specification • 关键词搜索	<pre> classDiagram class IUserSpecification { <<interface>> +boolean isSatisfiedBy(User user) +IUserSpecification and(IUserSpecification spec) +IUserSpecification or(IUserSpecification spec) +IUserSpecification not() } class CompositeSpecification { +IUserSpecification and(IUserSpecification spec) +IUserSpecification not() +IUserSpecification or(IUserSpecification spec) } class UserByAgeThan { } class UserByNameEqual { } IUserSpecification < -- CompositeSpecification CompositeSpecification < -- UserByAgeThan CompositeSpecification < -- UserByNameEqual </pre>

	<div><div><div>AndSpecification</div><div></div><div>and规格书</div></div><div><div>OrSpecification</div><div></div><div>or规格书</div></div><div><div>NotSpecification</div><div></div><div>not规格书</div></div></div>
模式名称&功能	示例类图
双重检查锁	