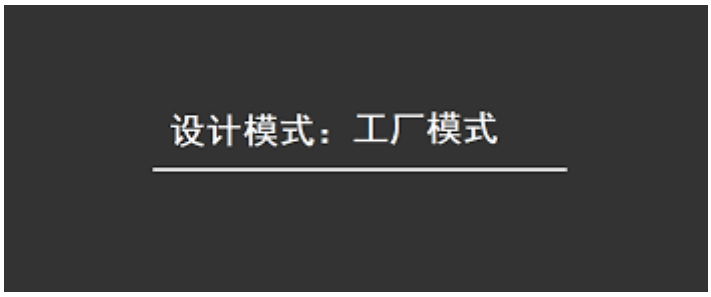
设计模式-工厂模式



参考

- 菜鸟教程
- 程序员麻辣烫

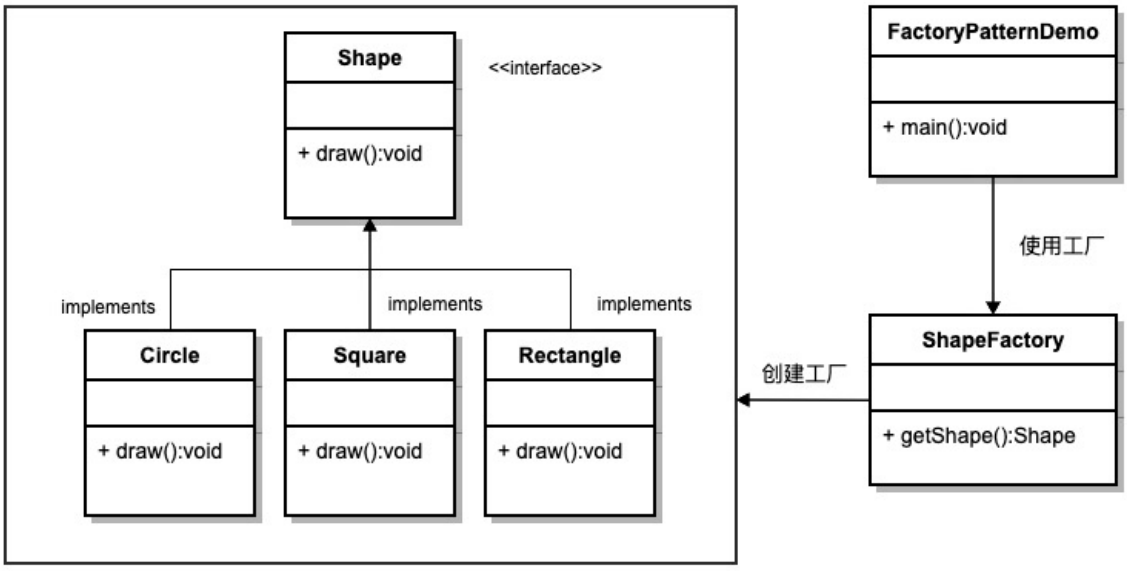
工厂模式是最常用的设计模式之一。这种类型的设计模式属于创建型模式，它它提供了一种创建对象的最佳方式。在工厂模式中，我们在创建对象时不会客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

使用场景

工厂模式一般用于对于不同的场景，需要创建不同的对象，但是这些对象实现的功能是很相似的，可以抽象出一个父类实现对应对象的创建。

- 在加载配置文件时，通过后缀来解析配置文件，并将文件内容写入内存。

Demo分析



> 分析：
>
> 根据上图逻辑，我们有一个 `Shape` 的抽象类，同时有对应的三个子类。我们可以通过 `ShapeFactory` 来创建对应的实例。

Python实现

```
class Shape():
  """
    ## 形状的抽象类
  """
  def __init__(self, shapeName):
    self.shapeName = shapeName
  def draw(self):
    print("drawing " + self.shapeName)

class Circle(Shape):
  def __init__(self, shapeName, radius):
    super().__init__(shapeName)
    self.radius = radius

class Square(Shape):
  def __init__(self, shapeName, xWidth):
    super().__init__(shapeName)
    self.xWidth = xWidth

class Rectangle(Shape):
  def __init__(self, shapeName, xWidth, yHeight):
    super().__init__(shapeName)
    self.xWidth = xWidth
    self.yHeight = yHeight

class ShapeFactory():
  """
    ## 工厂模式实现
  """
  shapeConfig = {
    "circle": Circle,
    "square": Square,
    "rectangle": Rectangle
  }
  @classmethod
  def getShape(self, shapeName, *args, **kwargs):
    obj = self.shapeConfig.get(shapeName, None)
    if obj is None:
      raise Exception("Shape not found")
    return obj(shapeName, *args, **kwargs)

def test():
  circle = ShapeFactory().getShape("circle", 10)
  square = ShapeFactory().getShape("square", 10)
  circle.draw()
  square.draw()

if __name__ == '__main__':
  test()
```

输出

```
drawing circle
drawing square
```

Go实现

```go
package main

import "fmt"

type ShapeDrawer interface {
    Draw()
}

type shapeCommFuncs struct {
    Name string
}

type Circle struct {
    x, y, radius int
    shapeCommFuncs
}

func (t *shapeCommFuncs) Draw() {
    fmt.Printf("绘制%s\n", t.Name)
}

type Square struct {
    x, y, length int
    shapeCommFuncs
}

type Rectangle struct {
    x, y, width, height int
    shapeCommFuncs
}

/**
 * 工厂模式创建对象
 */
func createFactory(shapeName string) ShapeDrawer {
    switch shapeName {
    case "Circle":
        return &Circle{
            shapeCommFuncs: shapeCommFuncs{Name: shapeName},
        }
    case "Square":
        return &Square{
            shapeCommFuncs: shapeCommFuncs{Name: shapeName},
        }
    case "Rectangle":
        return &Rectangle{
            shapeCommFuncs: shapeCommFuncs{Name: shapeName},
        }
    default:
```

```
        return nil
    }
}

func test() {
    c := createFactory("Circle")
    s := createFactory("Square")
    c.Draw()
    s.Draw()
}

func main() {
    test()
}
```

输出

```
绘制Circle
绘制Square
```