



# 软件工程 *SPOC*

## 第九章 面向对象实现

深圳大学 计算机与软件学院 杜文峰

# 问题思考?

面向对象实现与结构化实现的区别在什么地方?

函数

类

如何组织面向对象软件项目呢?

# 1. 项目文件组织

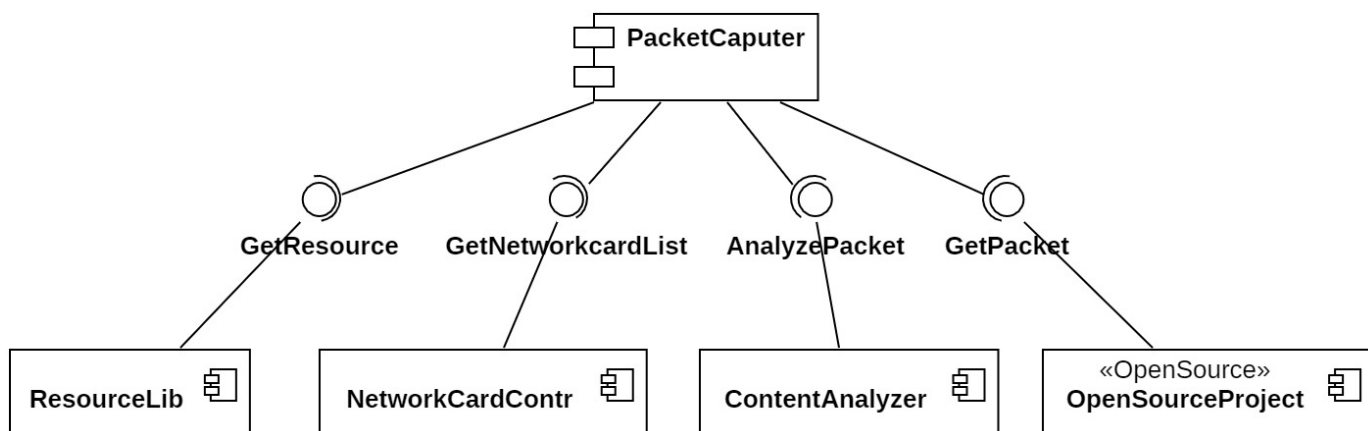
---

## [1] 建立项目文件管理目录



# 1. 项目文件组织

## [2] 依次构建各个构件的包含内容



Data (D:) > PacketCaptuer >

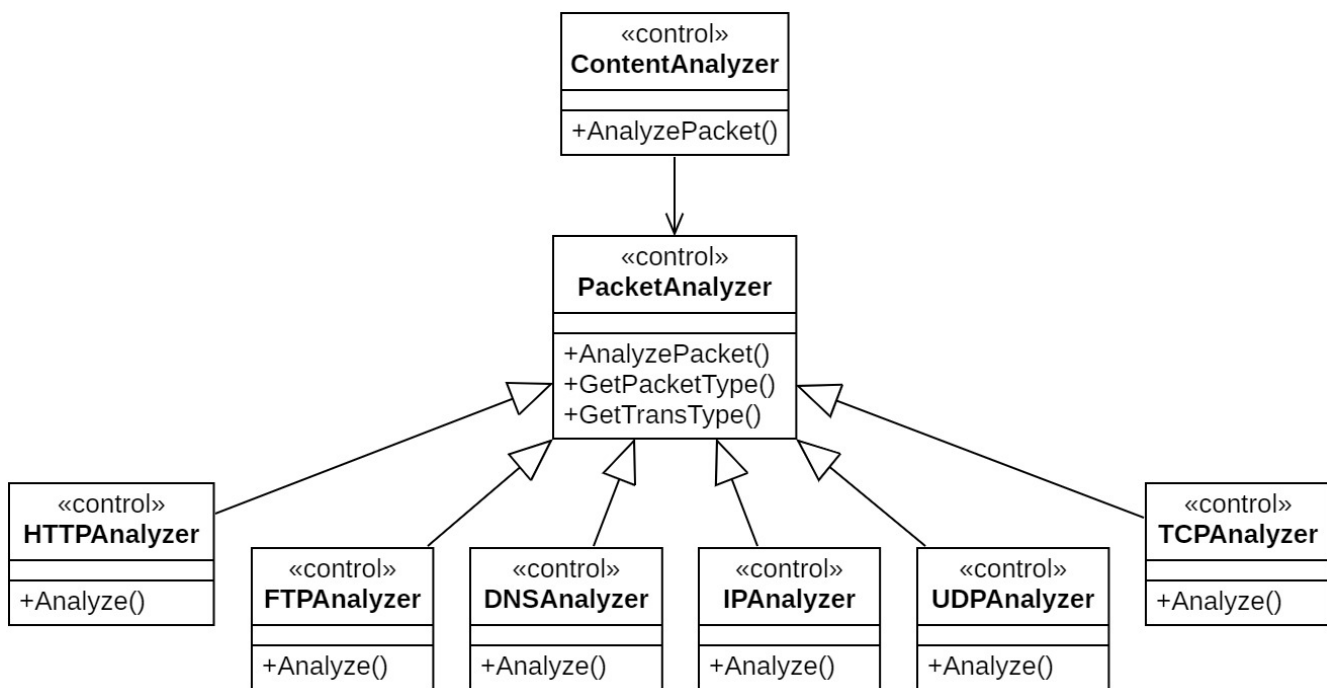
名称

类型

ContentAnalyzer	文件夹
NetworkCardContr	文件夹
OpenSourceProject	文件夹
ResourceLib	文件夹

# 1. 项目文件组织

## [3] 编译并测试软件构件



Data (D:) > PacketCaptuer > ContentAnalyzer

名称	类型
ContentAnalyzer	Dev-C++ Project File
ContentAnalyzer.layout	LAYOUT 文件
ContentAnalyzerMain	C++ Source File
ContentAnalyzerTest	C++ Source File
DNSAnalyzer	C++ Source File
DNSAnalyzer	C Header File
FTPAnalyzer	C++ Source File
FTPAnalyzer	C Header File
HTTPAnalyzer	C++ Source File
HTTPAnalyzer	C Header File
IPAnalyzer	C++ Source File
IPAnalyzer	C Header File
PacketAnalyzer	C++ Source File
PacketAnalyzer	C Header File
TCPAnalyzer	C++ Source File
TCPAnalyzer	C Header File
UDPAnalyzer	C++ Source File
UDPAnalyzer	C Header File

# 1. 项目文件组织

## [4] 构件集成

Data (D:) > PacketCaptuer >		
名称	类型	
ContentAnalyzer	文件夹	
NetworkCardContr	文件夹	
OpenSourceProject	文件夹	
ResourceLib	文件夹	
PacketCaptuer	Dev-C++ Project File	
PacketCaptuer.layout	LAYOUT 文件	
PacketCaptuerMain	C++ Source File	
TestMain	C++ Source File	

# 1. 项目文件组织

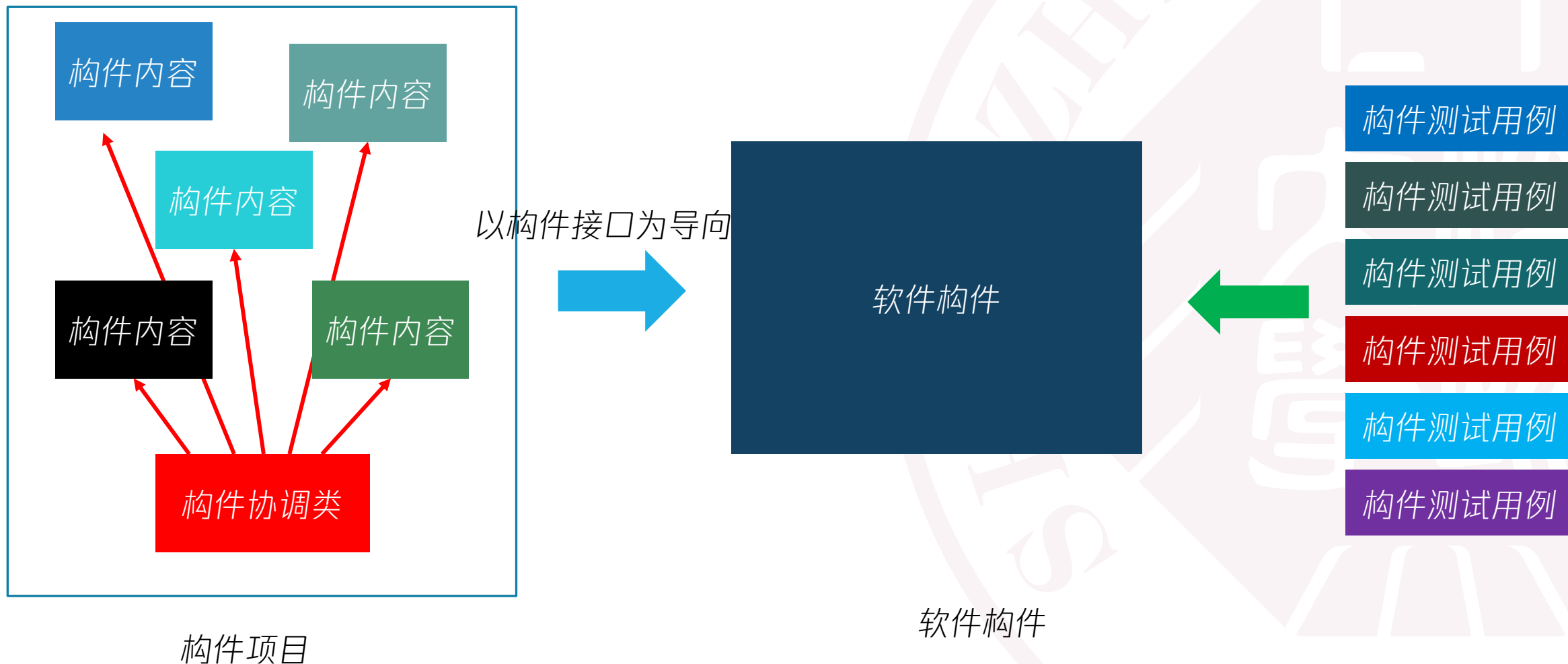
---

## [5] 目标项目业务实现



## 2.面向对象业务实现

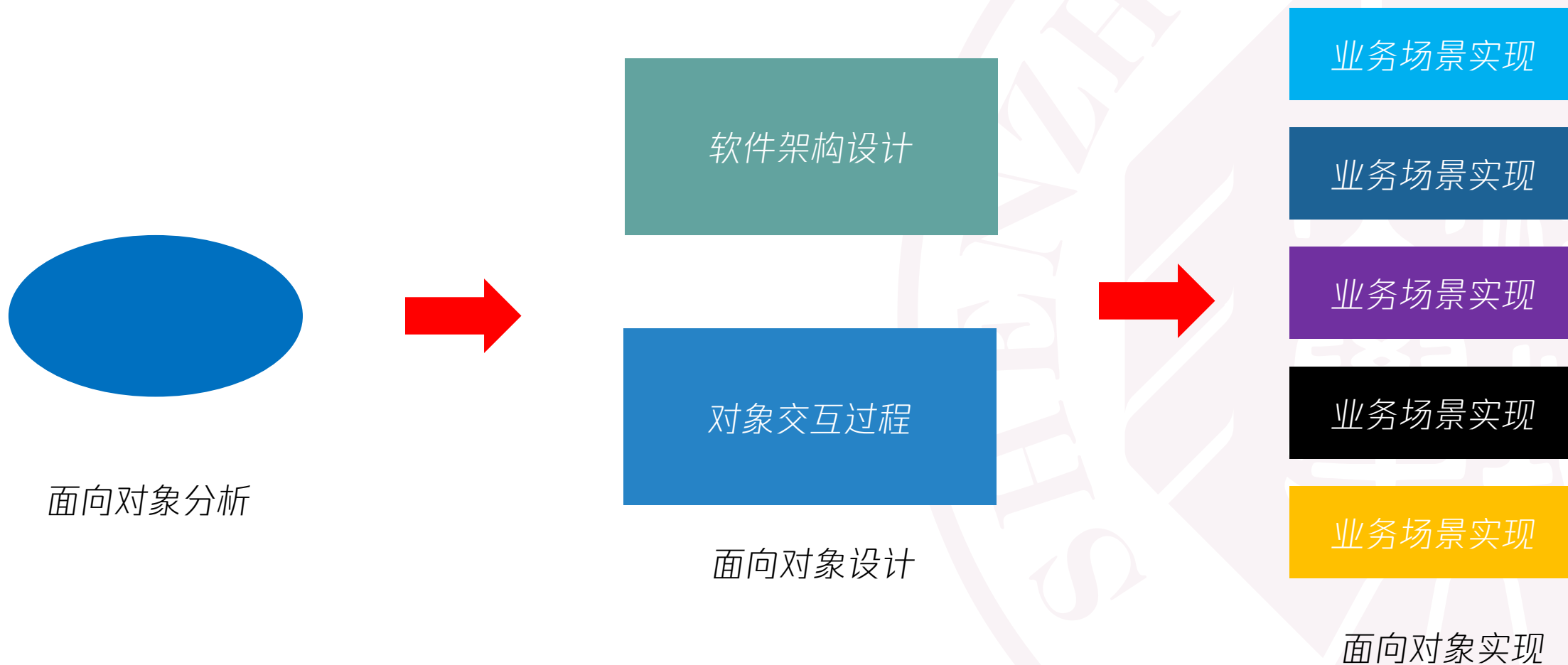
### [1] 构件实现





## 2.面向对象业务实现

### [2] 业务实现



### 3.面向对象软件集成

---

- [1] 传统集成
- [2] 协作集成
- [3] 基于事件〔消息〕的集成
- [4] 基于使用的集成
- [5] 客户端/服务器模式集成
- [6] 分布式集成

## 4. 面向对象测试阶段

[1] 针对面向对象分析的测试

- 对确定的类候选者进行测试
- 对确定的类关联进行测试
- 对主题进行测试
- 对定义的类属性和实例关联进行测试
- 对定义的类服务和消息关联进行测试

## 4. 面向对象测试阶段

### [2] 针对面向对象设计的测试

- 对类设计的测试
- 对软件架构设计的测试
- 对对象交互设计的测试
- 对软件构件的测试
- 对软件部署模式的测试

## 4. 面向对象测试阶段

[3] 针对面向对象编程的测试

- 检测完成的代码是否与设计结果相符
- 检测构件封装是否正确
- 检测完成代码的风格是否符合要求

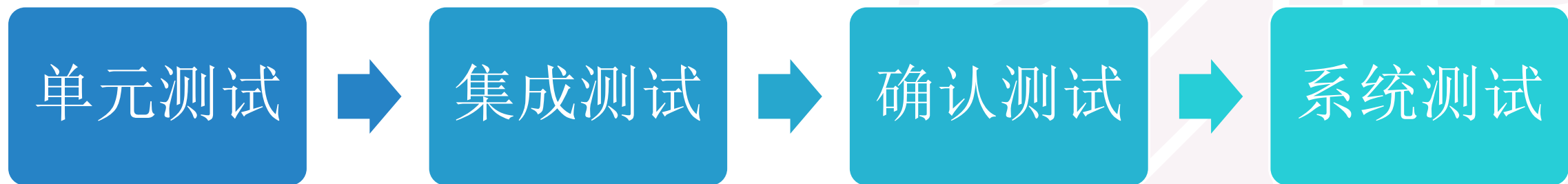
## 4. 面向对象测试阶段

---

### [4] 针对面向对象软件的测试



## 5.面向对象测试策略



## 5.面向对象测试策略

---

面向对象测试与结构化测试的差异

[1] 单元测试

[2] 集成测试



## 6.单元测试用例设计

*SampleClass*类的操作系列如下:

*Initialize().[GetAttribute()|SetAttribute()|UpdateAttribute()|MethodWithAttribute()|MethodWithoutAttribute()]\*.Finalize()*

SampleClass
+Attribute +WorkState
+Initialize() +GetAttribute() +SetAttribute() +UpdateAttribute() +MethodWithAttribute() +MethodWithoutAttribute() +Finalize()

## 6.单元测试用例设计

### [1] 随机测试

### [2] 划分测试

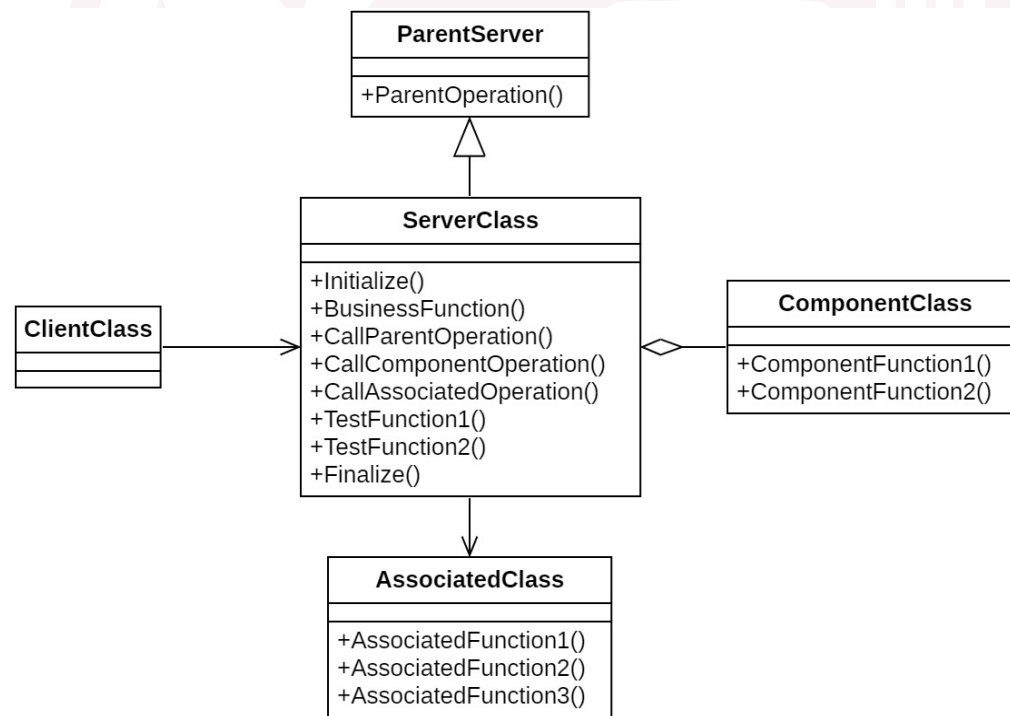
- 基于状态的划分
- 基于属性的划分
- 基于功能的划分

### [3] 基于故障的测试

# 7. 集成测试用例设计

最小操作集如下:

*Initialize().[BusinessFunction()|CallParentOperation()|CallComponentOperation()|  
CallAssociatedOperation()|TestFunction1()|TestFunction2()]\*.Finalize()*



## 7. 集成测试用例设计

---

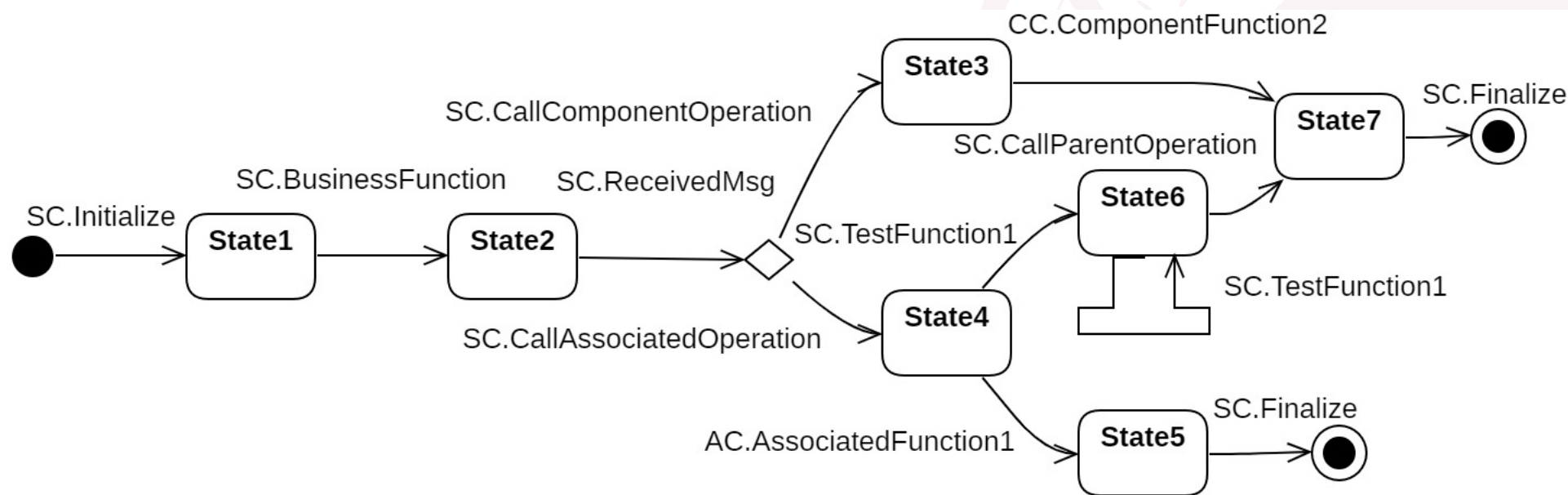
[1] 集成随机测试

[2] 集成划分测试



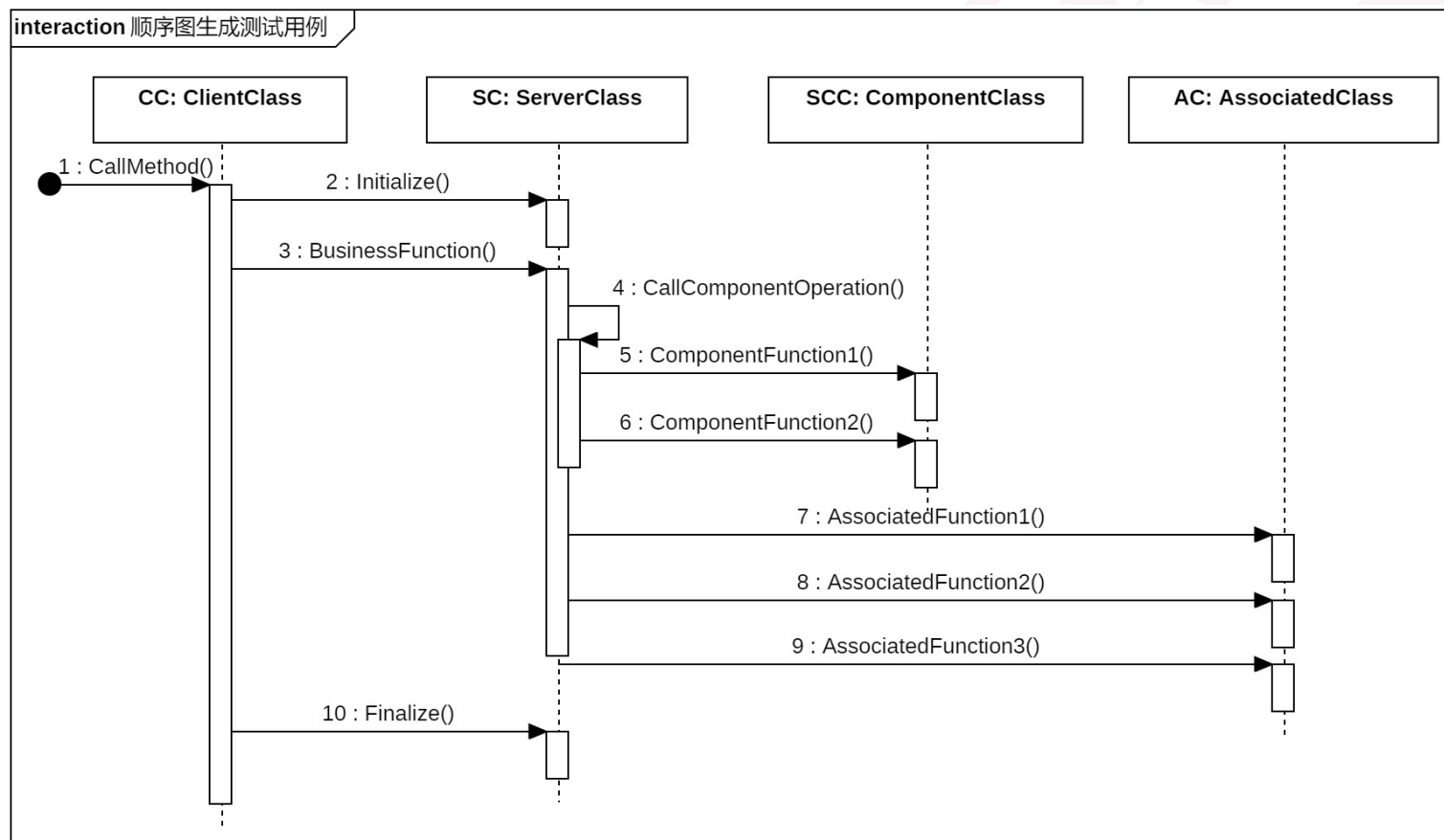
# 7. 集成测试用例设计

## [3] 由状态图导出测试用例



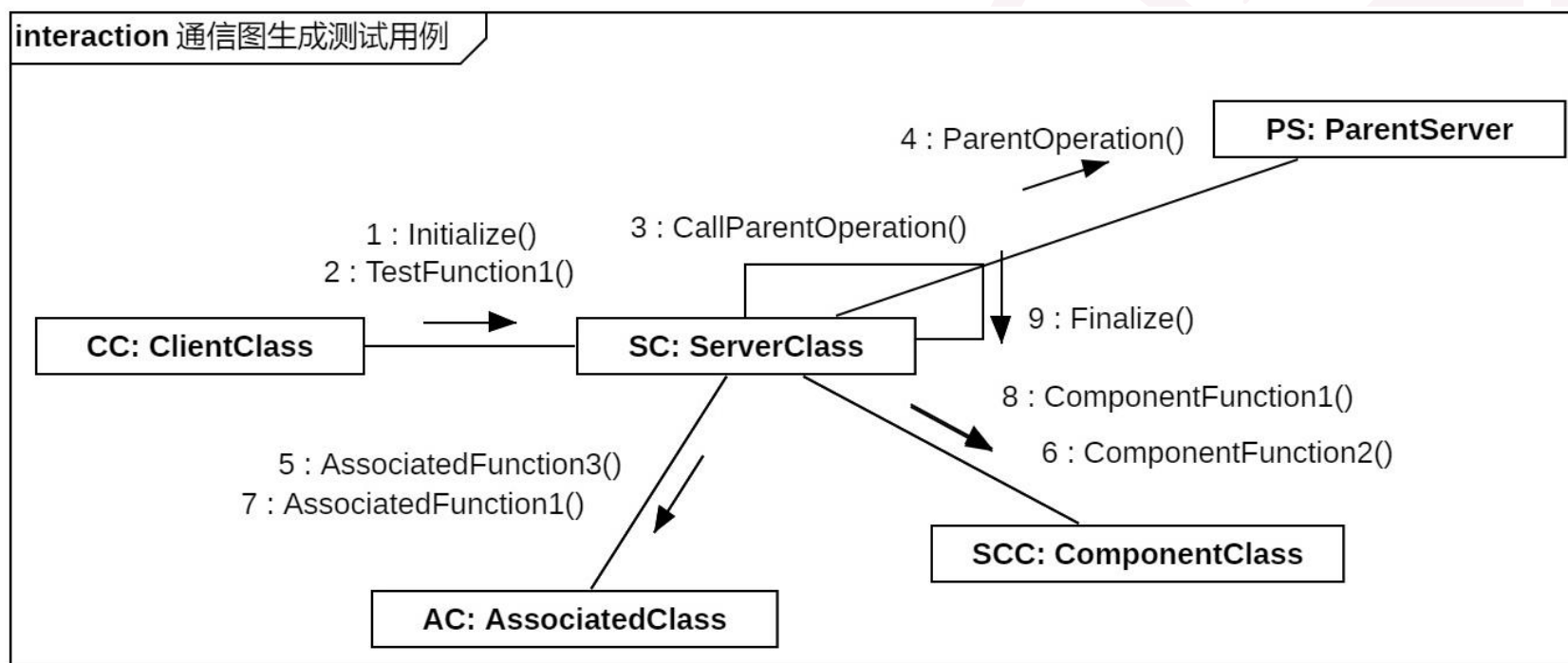
# 7. 集成测试用例设计

## [4] 由顺序图导出测试用例



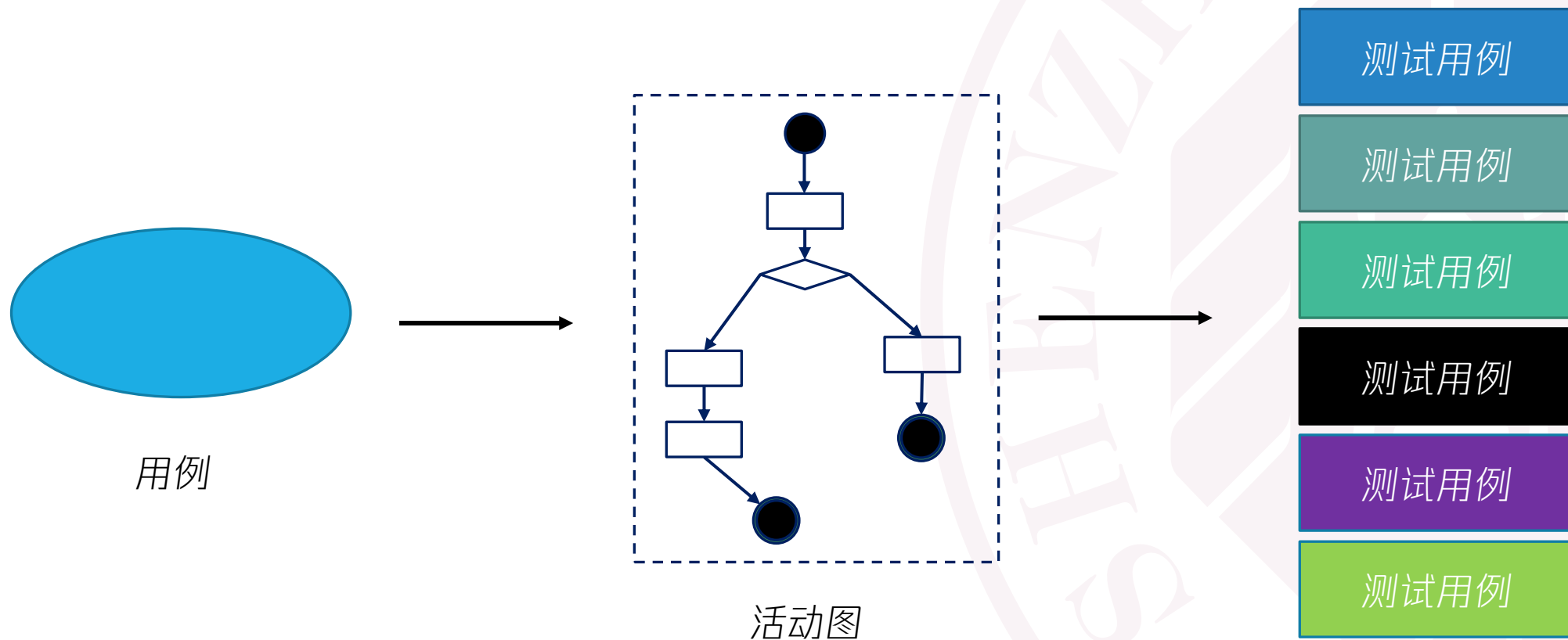
## 7. 集成测试用例设计

### [5] 由通信图导出测试用例



## 7. 集成测试用例设计

### [6] 基于场景的测试 [由活动图导出测试用例]





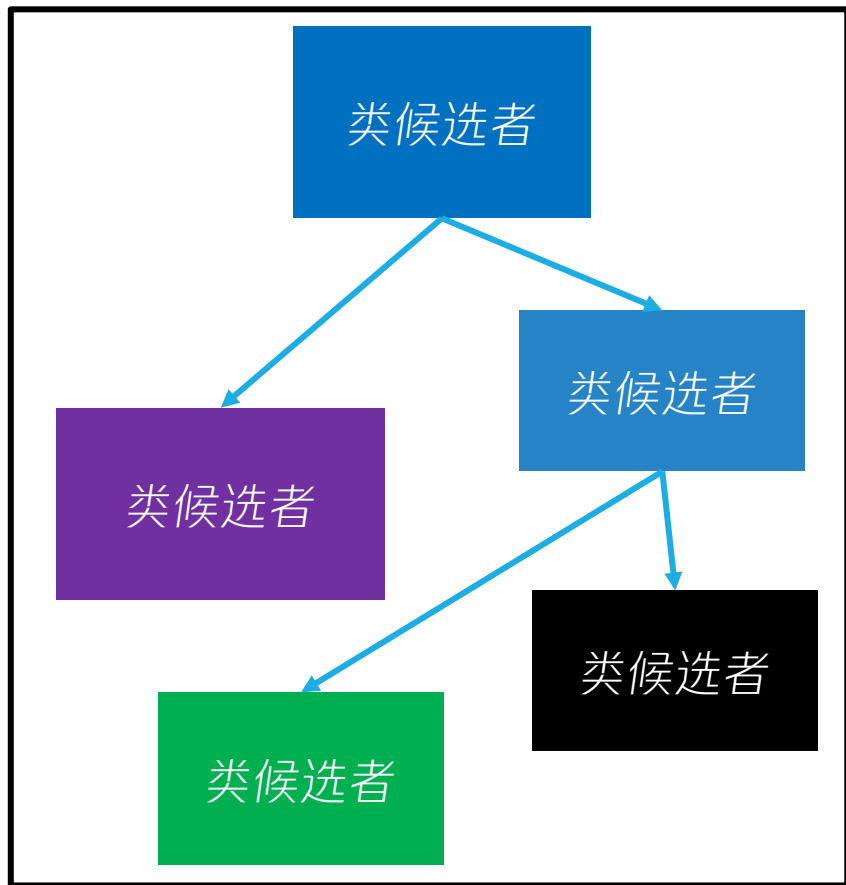
## 7. 集成测试用例设计

---

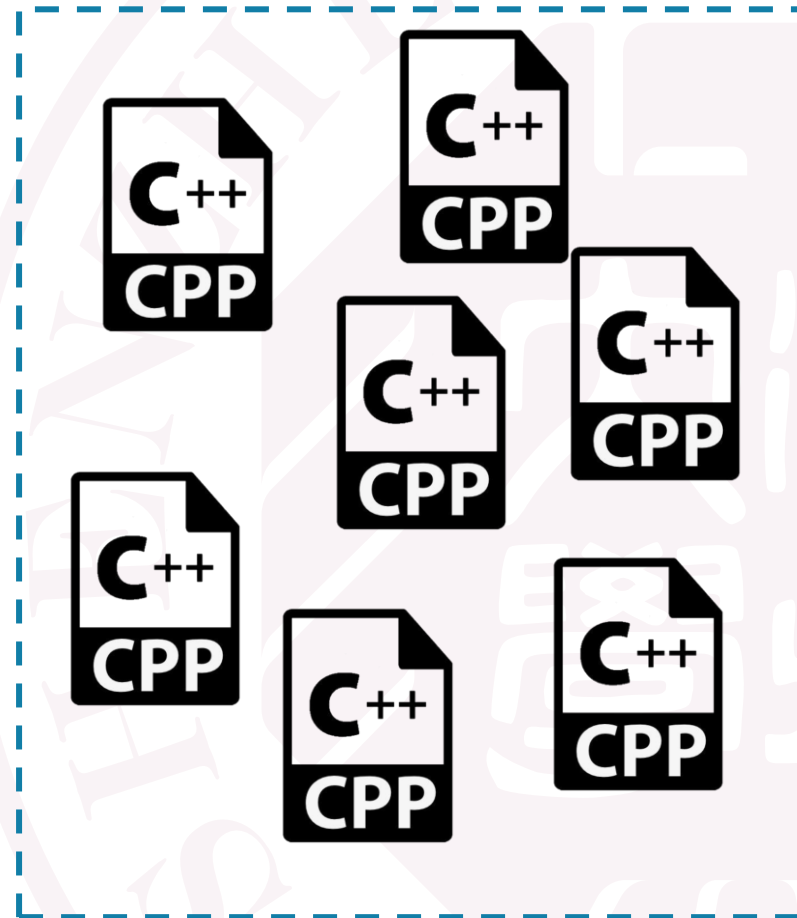
### [7] 基于异常、故障的测试

依赖于软件测试人员对面向对象分析和设计阶段的成果进行分析。软件测试人员通过推测软件中可能存在的错误，针对推测的错误设计测试用例。

## 8. 小结



面向对象设计



程序源代码

## 8. 小结

