

6.170 复习

提纲:

1. 解藕
2. 数据抽象
3. 抽象函数与表示不变式
4. 迭代抽象与迭代器
5. 对象模型和不变式
6. 相等、复制和视图
7. 动态分析
8. 模式设计
9. 子类型
10. 用例学习

解藕

L2, L3, Ch1, Ch13: 1-3, Ch2

分解

分工

重用

模块分析

局部变化

自顶向下设计与模块化

解藕

L2: 用例, 依赖, 规范, MDD

用例图: 树, 层, 环

推理

重用

构造顺序

依赖与规格说明: 模块依赖图用于

- 弱化假设
- 评价变化
- 通信
- 多种实现

解藕

L2: MDD, 技术

模块依赖图

- 规范部分
- 实现部分
- Meets, 依赖, 弱依赖关系

技术

- 正面: 位于两部分集合之间新的部分
- 隐藏表示: 避免提到数据是如何表示的
- 多态性: 很多形态
- 回调: 一个过程运行时的引用

解藕

L3: Java 命名空间, 访问控制

Java 命名空间

- 包 → {接口, 类} → {方法, 命名属性}

访问控制

- public: 从任何地方都可以访问
- protected: 在同一包内或包外的子类中可以访问
- default: 在同一包内可以访问
- private: 仅在同一类内可以访问

解藕

L3: 安全的语言, 接口

安全的语言

- 一个部分应该仅仅以来于命名此部分的另外一个部分
- 强拼写检查: 在程序运行过程中对类型 t 的访问受到保护
- 编译时间的类型检查: 静态拼写检查

接口: 更灵活的子类型

- 表达纯规范
- 允许一个规范部分的多个部分实现

解藕

L3: 创建一个程序

- 通过参数表示的抽象
- 带接口的解藕
- 接口与抽象类的比较
- 静态属性

数据抽象

L4, L5, Ch3-5, Ch9

规格说明

前置条件 (必须的)

客户端 (方法的调用者) 的职责

被忽略也是正确的, 什么也不需要做

后置条件 (效果)

实现者的职责

不能被忽略

框架条件 (修改的)

描述哪一个小过程被修改了

如被忽略, 不需要做任何修改

数据抽象

L4: 规格说明

- 操作的规格说明: 一系列方法执行的步骤
- 声明的规格说明: 不需要给出步骤实现的细节
- 异常与前置条件 (决定)
 - 前置条件: 检查的代价, 方法的范围
 - 通过运行确认来检查
 - 一旦违反, 抛出没有检查的异常 (在规格说明中没有提及到)

数据抽象

L4: 规格说明

- 简写
 - 返回: 不作修改, 返回一个值
 - 抛出: 在抛出字句中给出条件和异常
- 规格说明顺序: 一个规格说明 A 至少和规格说明 B 一样强, 如果:
 - A 的前置条件不强于 B
 - A 的后置条件不弱于 B, 过程满足 B 的前置条件
 - (总是能弱化前置条件, 总是能加强后置条件)

数据抽象

L4: 规格说明

- 评价规格说明
 - 一致
 - 需要播报
 - 足够健壮
 - 足够弱
- 在实现者和客户端的关键的防火墙

数据抽象

L5: 抽象类型

- 数据抽象: 类型由你执行的操作来定
- 易变的: 可以改变, 在执行的字句导致同一对象的其它操作时, 提供操作哪个来给出不同的结果 (向量)
- 不变的: 不能改变 (字符串)

数据抽象

L5: 抽象类型

- 操作 (T=抽象类型, t=某些其它类型)
 - 构造者: $t \rightarrow T$
 - 生产者: $T, t \rightarrow T$
 - 改变者: $T, t \rightarrow \text{void}$
 - 观察者: $T, t \rightarrow t$
- 举例: List

数据抽象

L5: 抽象类型

- 设计一个抽象类型
 - 少的、简单的操作可以合并成强大的操作
 - 操作应该有定义好的目的、连贯的行为
 - 操作集应该是足够的
 - 类型可以是一般的 (列表、集合、图) 或域特殊的 (接到地图, 员工数据库、电话簿), 但是不是二者的综合

数据抽象

L5: 抽象类型

- 表示: 实现一个抽象类型的类提供了一个表示
- 表示独立性
 - 确保一个抽象类型的使用与表示是独立的
 - 表示的变化应该不影响代码的使用

- 表示的暴露
 - 表示传输给客户端
 - 客户端允许直接访问表示
 - 需要仔细的编程纪律

数据抽象

L5: 抽象类型

- 语言机制
 - 私有的属性：阻止对表示的访问
 - 接口：表示独立性（列表→数组列表，链式列表）
 - ◆ 非静态的属性不允许
 - ◆ 不能有构造函数

数据抽象

L6: 抽象函数与表示不变式

- 表示不变式 (RI)
 - 关于是否一个抽象数据类型 (ADT) 实例是合式的约束
 - RI: Object-->Boolean
 - 对象模型 (OM) 的一些属性，而不是 RI (如共享/多策略)
 - RI 的一些属性，而不是 OM (如原始类型)

数据抽象

L6: 抽象函数与表示不变式

- 归纳推理
 - 表示不变式：使得模块推理变成可能
 - 构造函数创建了一个统计不变式的对象
 - 产生者保存不变式
 - 改变者：RI 一旦开始保持，那么要保持到最后
 - 观察者不修改，所以 RI 应该保持

数据抽象

L6: 抽象函数与表示不变式

- 抽象函数：解释表示
 - 具体对象：实现的真正的对象
 - 抽象对象：抽象类型描述其值的规范方式的数学对象
 - 介于抽象和具体之间的函数是抽象函数
 - 可能是部分的
 - 不同的表示有不同的抽象函数

数据抽象

L6: 抽象函数与表示不变式

- 善意的影响：允许观察者改变不变式，只要抽象值被保持
- 表示不变式
 - 模块推理
 - 帮助捕捉错误

- 抽象函数 (AF)：指定一个 ADT 表示是如何作为一个抽象值被解释的

数据抽象

L7: 迭代抽象与迭代器

- 表示暴露：remove 方法抛出 UnsupportedOperationException 异常

- 参考课本的第 6 章

对象模型和不变式

L8, Ch12:1

- 对象模型：集合配置描述
 - 对象分类
 - 对象之间的关系
 - 子集（实现，继承）
 - 关系与标签
 - 多样性：在一个类中有多少个对象可以与另一个类的一个给定对象关联
 - 可变性：多少个状态可以改变

对象模型与不变式

- 多样性符号：
 - * (≥ 0)
 - + (≥ 1)
 - ? (0 or 1)
 - ! (exactly 1)
- 源 \rightarrow 目标
 - 箭头尾：每个源与多少个目标关联？
 - 箭头头：多少个源可以映射到某个目标？
- 实例图

对象模型与不变式

- 程序对象模型
- 抽象和具体的观点
 - 抽象函数：能够显示具体对象是如何被解释为抽象值的
 - 表示不变式：对象模型是一个表示不变式类型——在程序生命周期中一直保持的约束
 - 表示暴露：ADT 提供在表示不变式的轮廓中对某一对象的直接访问

相等、复制和视图

L9, Ch5: 5-7

- 对象契约
 - equals()
 - hashCode()
- 相等属性（点和有色点）
 - 灵活性
 - 对等性
 - 传递性
- 哈希：如果两个对象相等（equals），那么必须有相同的哈希码（hashCode）

相等、复制和视图

- 复制
 - 浅的：属性指向旧对象相同的属性
 - 深的
- 可克隆的接口
- 元素和容器相等
 - Liskov 解决方案

- 相等：行为上的等同
- 相似：观察上的等同

相等、复制和视图

- 表示暴露：轮廓包括元素类（以 LinkedList 为例）
 - 改变哈希键
- 视图
 - 提供对优先级数据结构进行不同访问的独特对象
 - 视图和优先级结构是可以修改的

动态分析

L10, L11, Ch10

- 执行程序并观察它的行为
- Dijkstra: “测试可以显示错误的存在而不是没有错误”
- 不能仅仅依赖于动态分析——需要好的规范说明和设计

动态分析

L10: 防御性编程

- 指导方法
 - 插入冗余的检查——运行时断言
 - 当你正在编写代码时
 - 哪里？
 - 过程的开始（前置条件）
 - 复杂过程的结尾（后置条件）
 - 当一个操作可能产生外部影响时

动态分析

L10: 防御性编程

- 捕捉常见异常
 - NullPointerException
 - ArrayIndexOutOfBoundsException
 - ClassCastException
- 检查表示不变式
 - `public void repCheck () throws (runtime expn)`
- 断言框架
 - `public static assert(Boolean b, String loc)`
 - `Assert.assert (..., “MyClass.myMethod”);`

动态分析

L10: 防御性编程

- 子类中的断言
- 响应失败
 - 修复：复杂的，更多错误，如果你知道原因→那么你是否可以避免
 - 执行特殊的动作：依赖于系统→很难决定动作集
 - 取消执行：依赖于程序、编译器和单词处理器

动态分析

L11: 测试

- 测试考虑
 - 你想要测试的属性（问题域，程序知识）

- 你想要测试的模块（标准，复杂度，可能产生的故障）
- 如何产生测试用例
- 如何检查结果
- 测试应该何时结束

动态分析

L11: 回归测试

- 可以被重新执行的测试集
- 第一次测试编程：在应用代码之前写的回归测试构造（极限编程的一部分）

动态分析

L11: 标准

- $S(t, P(t)) = \text{false}$, t 时一个失效的测试用例
- $C: \text{Suite, Program, Spec} \rightarrow \text{boolean}$
- $C: \text{Suite, Spec} \rightarrow \text{Boolean}$ is specification-based criterion; black box
- $C: \text{Suite, Program} \rightarrow \text{Boolean}$ is a program-based criterion; glass box

动态分析

L11: 子域

- 子域：输入空间分开
 - 决定测试集是否足够好
 - 把测试驱动到错误最可能出现的区域
- 揭示子域

动态分析

L11: 子域标准

- 过程覆盖：每个过程必须至少执行一次
- 决定覆盖：控制流图中的每个边界必须执行
- 条件覆盖：布尔表达式用于评价正确或错误
- 边界测试：每个条件的边界用例
- 基于标准的规范说明：仅仅在子域中
 - 空集，非空且包含元素，非空且不包含元素

动态分析

L11: 灵活性和可行性

- 如果可能满足，那么标准就是灵活的
- 使用基于标准的规范说明来指导测试集的开发
- 采用基于标准的程序来评价（衡量代码的覆盖性）

设计模式

L12, L13, L14, ch15

- 迄今为止：
 - 封装（数据隐藏）
 - 子类（继承）
 - 迭代
 - 异常
- 不要过早地使用设计模式
- 复杂的降低了可理解性

设计模式

L12: 创造性模式

- 工厂
 - 工厂方法：可以制造一个特殊类型的对象方法
 - 工厂对象：封装工厂方法的对象
 - 原型：可以克隆它自己的对象，对象传递给了一个方法（而不是一个工厂对象）

设计模式

L12: 创造性模式

- 共享
 - 单例：一个类仅仅有一个对象
 - 驻留：重用对象而不是创建一个新的，仅仅适用于不可改变的对象
 - 享元：（驻留的泛化），如果大多数对象是不可改变的，那么就可以使用
 - 内在的和外在的状态
 - 仅仅在空间是一个瓶颈时使用

设计模式

L13: 行为模式

- 多路通信
 - 观察者：当状态改变时，维护观察者列表（遵循一个特殊的接口），需要增加和删除观察者方法
 - 黑板：（泛化观察者模式），多个数据源和多个视图，异步的
 - 所有处理器可读和可写的消息仓库
 - 互操作性，易于理解的消息格式
 - 调解器：（观察者和黑板之间的媒介），解藕信息，但是不控制，同步的

设计模式

L13: 组合

- 支持很多不同的操作
- 在组合的部分上执行操作
- 解释器：某一特殊对象类型的操作组合在一起
- 过程：实现一个特殊操作的所有代码组合在一起
- 访问者：在层次结构中方法，节点接受访问者，访问者访问节点

设计模式

L14: 结构化模式

- 包装

模式	功能性	接口
■ 适配器	相同	不同
（互操作性）		
■ 装饰器	不同	相同
（继承）		
■ 代理	相同	相同
（控制或限制）		

设计模式

L14: 结构化模式

- 包装（Wrapper）的实现
 - 继承
 - 授权：在属性中存储对象，倾向于包装的实现

- 组成
 - 允许客户以同样方式使用一个单元或单元的集合
- 子类型**
L15, Ch7

- 模块依赖图 MDDS
- 替换原则
 - 签名
 - 方法
 - 需要较少的变化
 - 保证更多的协方差
 - 属性
- Java 的子类与子类型
- 接口
 - 保证行为 w/o 共享代码
 - 多继承

用例学习: Java 集合 API

- 类型层次
 - 接口: Collection, Set, SortedSet, List
 - 框架实现: AbstractCollection, AbstractSet, AbstractList, AbstractSequentialList
 - 具体实现: TreeSet, HashSet, ArrayList, LinkedList
- 并行结构
 - 接口与抽象类

用例学习: Java 集合 API

L16, Ch13, Ch14

- 可选择的方法: 抛出 UnsupportedOperationException 异常
- 多态
- 框架实现 (模板方法和钩子方法)
- 容量, 分配, 垃圾回收
- 复制, 转化, 包装
- 有序集合: Comparable 与 Comparator
- 视图

用例学习: JUnit

L17

- 模块依赖图: 完全连接
- 设计模式
 - 模板方法
 - 命令模式
 - 组合模式
 - 观察者模式
- 使用 Java 反射的测试集

用例学习: Tagger

L18

- 设计方面
 - 动作

- Cross Referances
- 属性映射
- 自编号
- 风格表单视图
- 安全类型列举
- 质量需求
- 模板密度

概念对象模型

L19, Ch11-12

- 原子：
 - 不可分的
 - 不可变的
 - 不间断的
- 集合：原子集
 - 域：没有超集的集合
 - 关系：关联原子
 - 转置：~关系
 - 传递闭包：+关系
- 自反闭包：*关系

概念对象模型

- 三重关系
- 索引关系
- 举例

设计策略

L20

- 开发过程
 - 程序分析（对象建模和操作）
 - 设计（代码对象建模，MDD，模块规范）
 - 实现
- 测试
 - 回归测试
 - 运行断言
 - 表示不变式

设计策略

设计属性

- 扩展性
 - 对象模型充足
 - 局部性和解藕
- 可靠性
 - 仔细建模
 - 评审、分析和测试
- 有效性
 - 对象建模

- 避免偏差
- 表示的选择

设计策略

对象模型转换

- 介绍一个泛化（子集）
- 插入一个集合
- 颠倒一个关系
- 移动一个关系
- 关联一个表
- 增加一个冗余
- 可变关系
- 窜改一个接口
- 消除动态集合