School of Economics ang Management, Beihang University

现代程序设计技术

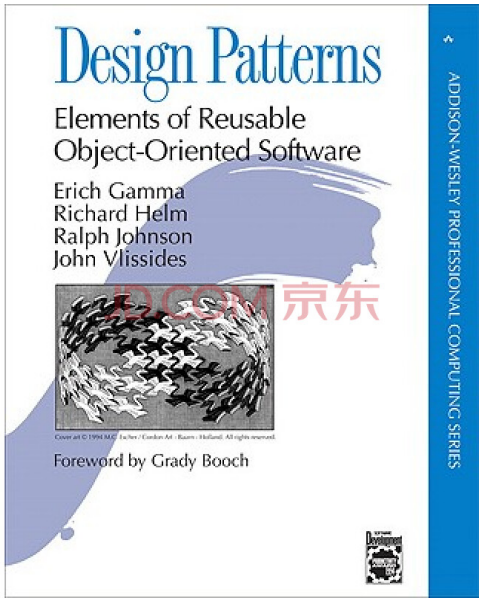
赵吉昌
jichang@buaa.edu.cn

本周内容

- 面向对象编程
  - 单例模式(singleton)
  - 继承
  - 运算符重载

Singleton

- 设计模式
  - Design Patterns
    - 最佳实践，软件开发人员通过试验和错误总结而来
    - 重用代码、让代码更容易被他人理解，并保证代码的可靠性
    - Design Patterns - Elements of Reusable Object-Oriented Software
  - 项目中应合理地运用设计模式
    - 注意平衡扩展性与效率



设计模式

- 开闭原则（Open Close Principle）
  - 对扩展开放，对修改关闭
- 里氏代换原则（Liskov Substitution Principle）
  - 任何基类可以出现的地方，派生类一定可以出现
  - 即基类可被派生类替换
- 依赖倒转原则（Dependence Inversion Principle）
  - 针对接口编程，依赖抽象而不依赖具体

设计模式

- 接口隔离原则（Interface Segregation Principle）
  - 使用多个隔离的接口，比使用单个接口要好
  - 降低类之间的耦合度
- 最小知道原则（Demeter Principle）
  - 一个实体应当尽量少地与其他实体发生作用
  - 系统功能模块应相对独立
- 合成复用原则（Composite Reuse Principle）
  - 尽量使用合成/聚合的方式，而不是使用继承

Singleton

- 单例模式
  - 全局只有一个实例
- 应用场景
  - 输入法
  - 全局配置参数

Singleton

- class Singleton:
  - def __init__(self):
    - pass
  - def __new__(cls,*args,**kwargs):
    - if not hasattr(Singleton, "_instance"):
      - Singleton._instance = object.__new__(cls)
    - return Singleton._instance
  - Demo: singleton.py
  - 注意该实现方式在多线程场景下不安全

继承

- 类的继承
  - 继承其他类的类称为派生类(derived class)
  - 被其他类继承的类称为这些类的基类(base class)
- 继承语法
  - `class DerivedClassName(BaseClassName):`
  - `    <statement-1>`
  - `    .`
  - `    .`
  - `    .`
  - `    <statement-N>`

继承

- 作用域
  - 基类必须与派生类定义在一个作用域内
  - 可以用表达式指定模块
  - `class DerivedClassName(modname.BaseClassName):`
- `    pass`
- 派生类
  - 派生类定义的执行过程与基类类似
  - 如果在类中找不到请求调用的属性会搜索基类
  - 如果基类由别的类派生而来，则会递归式搜索

继承

- 派生类的实例化
  - 搜索对应的类属性，必要时沿基类链逐级搜索
    - 递归搜索
  - 派生类可能会覆盖（override）其基类的方法
    - 派生类对功能进行定义或更新
    - 多态的一种体现形式
- Demo
  - People, ChinesePeople, AmericanPeople

继承

- 继承的检查
  - `isinstance()` 用于检查实例类型
    - `isinstance(obj, int)` 只有在 `obj.__class__` 是 `int` 或其它从 `int` 继承的类型时返回`True`
  - `issubclass()` 用于检查类继承
    - `issubclass(bool, int)` 为 `True`，因为 `bool` 是 `int` 的子类
    - `issubclass(float, int)` 为 `False`，因为 `float` 不是 `int` 的子类

继承

- 多继承
  - python支持多继承
  - 派生类可以同时继承多个基类
  - class DerivedClassName(Base1, Base2, Base3):
  - <statement-1>
  - .
  - .
  - <statement-N>
  - Demo: Study, ChineseStudent

继承

- 多继承
  - 需要注意圆括号中基类的顺序
    - 从左到右搜索
  - 多继承会导致菱形（diamond)关系
    - 有至少一个基类可以从子类经由多个继承路径到达
    - 基类方法可能被多次调用
  - `super()`方法
    - 防止重复访问，每个基类只调用一次
    - Demo: call.py
  - 通过子类实例对象可调用父类已被覆盖的方法
  - 慎用多继承（二义性）

运算符重载

- 运算符重载
  - operator overload
  - 对已有的运算符重新进行定义，赋予其另一种功能，以适应不同的数据类型
  - 运算符重载不能改变其本来寓意
  - 运算符重载只是一种 “语法上的方便”（sugar）
  - 是一种函数调用的方式

运算符重载

- 类的专有方法
  - `__init__`：初始化函数，在生成对象时调用
  - `__del__`：析构函数，释放对象时使用
  - `__repr__`：返回对象的字符串表达式
  - `__setitem__`：按照索引赋值
  - `__getitem__`：按照索引获取值
  - `__len__`：获得长度
  - `__call__`：实例对象可调用，可调用对象，可以有参数

运算符重载

- 类的专有方法
  - `__add__`：加运算
  - `__sub__`：减运算
  - `__mul__`：乘运算
  - `__truediv__`：整数除运算
  - `__floordiv__`：浮点除运算
  - `__mod__`：求余运算
  - `__pow__`：乘方
  - `__str__`：提供一个不正式的字符串表示，使得对象可用`print`输出

运算符的重载

- 类的专有方法
  - `__or__`: 运算符|
  - `__bool__`: 布尔测试，`bool(x)`
  - `__lt__`: <
  - `__gt__`: >
  - `__le__`: <=
  - `__ge__`: >=
  - `__eq__`: ==
  - `__ne__`: !=
  - `__iter__`, `__next__`: 迭代
  - `__contains__`: in运算符

运算重载

- 通过覆盖专用方法来实现
  - 定义实例对象的加法
  - 定义实例对象的减法
  - 定义实例对象的比较
    - `__lt__`等函数
    - 在需要对自定义的实例对象进行排序时可能有用
  - Demo
    - point.py

继承的应用

- 多态
  - 同一类对象的行为“多样”性
  - Demo
    - shape.py
- 对内部类的继承
  - 实现一个联系人列表
  - 能够按联系人的姓名排序
  - 能够按联系人的姓名进行检索
  - Demo
    - contact.py

继承的应用

- 复杂网络
  - 节点与边
  - 复杂系统的建模
  - 第三方库: networkx
  - 第三方工具：Gephi
- 词网络
  - 语义上的相似性可通过反复共现来“捕获”
  - 语义上的重要性可通过结构上的位置来度量
  - 查找关键词（可能比频率的效果要好）
  - 语义“形态”或话题“图景”的差异
    - 不同话题的词的组织结构可能不同

本周作业

- 通过构建词网来比较好评跟差评在用词、话题、用语习惯上的可能差别。
  - 提供京东计算机类产品的5分评论和1分评论
    - 资源/data/1分和5分评论文本.txt
    - 一行一条评论，内容和分值用\t隔开,共20000条，各10000条
  - 实现一个类WordNetwork，继承networkx里的Graph，并实现新的生成词网络的方法，当两个词在一条评论中的距离小于w时，认为两个词之间有语义关联并建立边（滑动窗口）
  - 实现边的过滤方法，即当两个词共现的次数小于t时，应该滤去，认为语义相关性不显著
  - 比较5分评论与1分评论词网的差别
    - 用结构属性对词排序(degree, hits, pagerank)，看共有词位置的差别
      - 比如看看topn的词的差别
      - 比如对所有共有词，产生一个反映重要性的归一化的向量，比较两个向量的差异（相关性小等）
    - 可视化共有词（或关键共有词）的连接结构有何差异（用networkx库里的可视化方法，或者直接用Gephi工具）
  - 利用networkx里的社团发现方法看看不同评论里的话题分布
    - 话题一般会体现为由一些连接紧密的词组成的子结构
    - 尝试用Gephi可视化
  - 当调整w（一般认为w=10以内）和t的时候，上述差异等比较有何变化？
  - 其他一些思考
    - 了解一下DeepWalk，如何在网络上对词的语义进行“嵌入”表达并将文本转成向量空间
    - 如何度量两个词的相似性，通过结构
    - 如果有时间信息，怎么通过结构的变化来量化语义的“时光变迁”(drift)