

- 共性问题
 - 作业命名不规范，没有学号等信息
 - 建议在作业中也加上，方便助教统计成绩
 - 在函数中import
 - 未使用pickle进行序列化

- 重构
 - Code refactoring
 - 在不改变软件系统外部行为的前提下，改善其内部结构
 - 通过调整程序代码改善软件的质量、性能，使其程序的设计模式和架构更趋合理，提高软件的扩展性和维护性
 - 重构和设计相辅相成
 - 能够迅速提高编程能力





School of Economics and Management, Beihang University

现代程序设计技术

赵吉昌

jichang@buaa.edu.cn

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

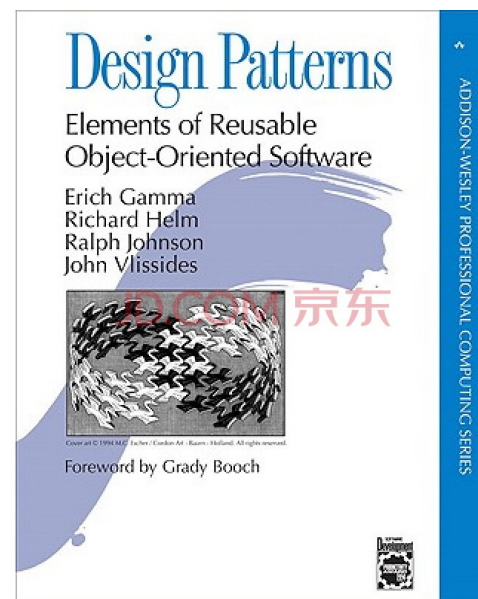
- 设计模式

- 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



- 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 (多态的体现)
- 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

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

- 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>`

- 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

- 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.py`

- 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

- 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: `people.py`

- 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

- Method Resolution Order (MRO)

- Python2.3之前：采用从左至右的深度优先遍历，但是如果遍历中出现重复的类，只保留最后一个

- C3算法：保证局部优先原则，保证单调性原则

- super () 方法

- 防止重复访问，每个基类只调用一次

- Demo: call.py, multib.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)

- MRO

- 多继承会因二义性而违反单调性原则

- `class X:pass`
 - `class Y: pass`
 - `class A(X, Y):pass`
 - `class B(Y, X):pass`
 - `class C(A, B):pass`

- 类A : [A, X, Y, object]

- 类B : [B, Y, X, object]

- 类C : [C, A, B, X, Y, object]

- 基类B的搜索顺序“被”子类C修改
 - 触发TypeError , MRO conflict

- <https://www.python.org/download/releases/2.3/mro/>

- 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

- 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

- 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

- 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

- 通过覆盖专有方法来实现
 - 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`

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

- 卷积神经网络 (CNN) 在图像分类、目标检测等领域取得了远超传统模型的效果，使得大量图片数据的应用成为可能。该模型中的卷积块，灵感一定程度上来自于早期图像处理领域的“滤波器 (filter)”，其用于提取图像中的某类特定特征或对图像做某些特定变换。本次作业要求基于类继承，体会多种滤波器的使用效果与实际应用，并初步掌握python对图片的简单处理。
 - 1. 具体要求见附件。
 - 2. 要求用类及其继承实现所有功能要求，并体现会多态。
 - 3. 安装并学习pillow