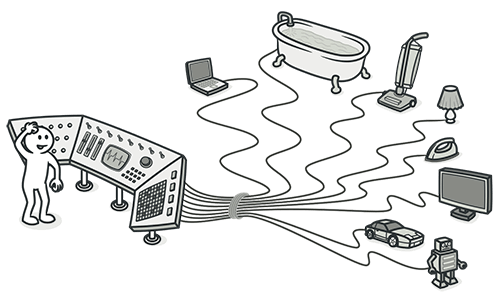
# **1.Switch Statements**

问题：有多个if-else判断，或者一个非常复杂的switch判断语句



问题原因：

1.当if-else分布在代码的各个地方时，你需要添加一个新的判断条件时，需要去各个地方同步修改。

2.当看到一个复杂的switch语句的时候，应该尝试使用多态去优化代码；

解决方法：

1. 将代码块封装出来成为一个新方法
2. 如果一个判断语句基于类型控制程序执行不同的方法时，将各模式写作子类，通过不同的值判断执行不同的方法，或者使用状态/策略模式去修改代码
3. 根据type编写不同的子类重写父类方法
4. 一个方法被分成几个部分，每个部分都根据参数的值运行时，将方法拆分为几个不同的方法。
5. 判断语句其中一个为空时，创建一个空类用来判空，使代码简洁（null有两种含义）

改动好处：

代码结构更清晰，提高了代码的可读性

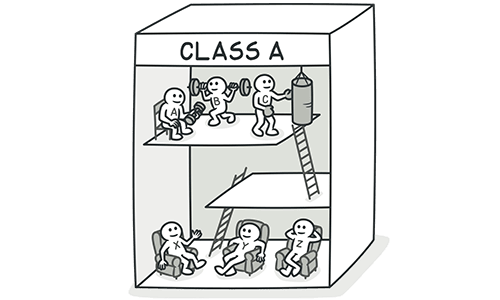


忽略问题的时机：

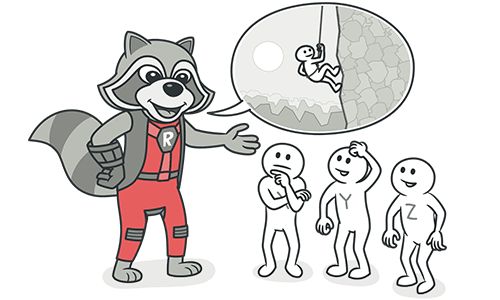
1. 当判断语句条件过于简单时，不需要进行重构
2. 工厂模式通常使用运算符来选择创建的类

# **2.Temporary Field**

问题：临时字段在多时间段都是未使用/未赋值

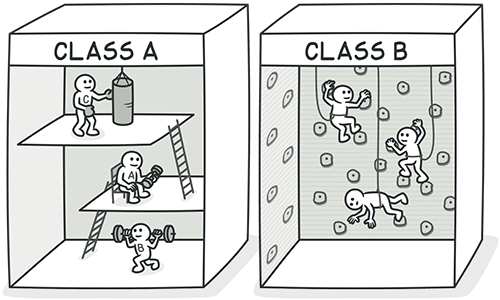


问题原因：对象中的某个变量仅为了某一个特定情形或者特定方法而设置，在正常使用时，该变量不会被使用，



解决方法：

1. 通过 Extract Class将临时字段和操作他的代码抽出放到一个单独的类中
2. 引入 Null 对象并将其集成，以取代用于检查临时字段值是否存在的条件代码

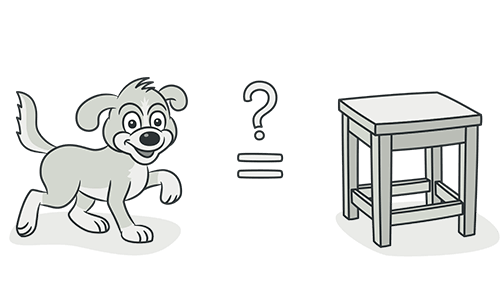


改动好处：

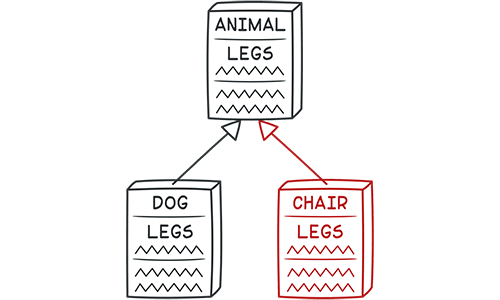
代码更清晰，可读性更强

# **3.Refused Bequest**

问题：子类只继承了父类的一部分函数和数据

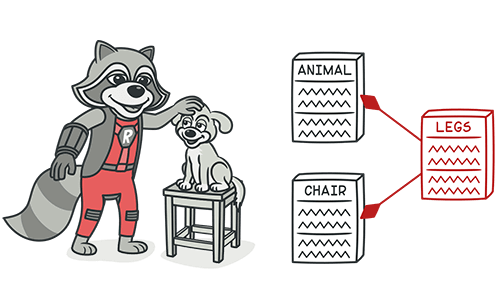


问题原因：有些人想在父类中重用代码



解决方法：

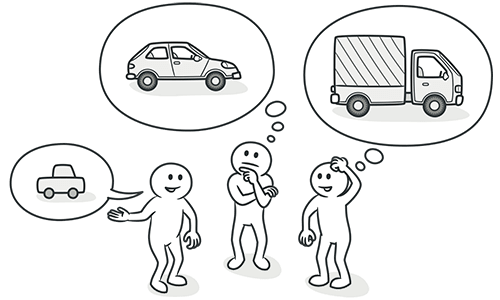
1. 如果继承无意义，请使用委派机制代替继承
2. 如果继承是合理的，则删除子类中不需要的字段和方法。从父类中提取子类需要的所有字段和方法，将它们放在一个新的子类中，并将两个类都设置为从父类中继承(提取父类)。



改动好处：提高代码的清晰度和组织性。将不再需要怀疑为什么该类是从该类继承的

# **4.Alternative Classes with Different Interfaces**

问题：两个类执行相同的函数，但具有不同的方法名



问题原因：创建其中一个类的程序员可能不知道功能等价类已经存在

解决方法：

1. 重命名方法，使他们在所有的替代类中相同，
2. Move 方法、 Add Parameter 和 Parameterize 方法使方法的签名和实现相同。
3. 如果两个类只有部分相同，请抽取超类，并选择一个类继承并合并不同的部分。
4. 确定处理完毕以后，删除另外一个重复的类

改动好处：

1. 摆脱重复代码，减少程序体积，代码结构清晰
2. 提升代码可读性和可理解性



何时不执行：

当无法合并，或者合并难度较大时，可以放弃合并。