气味	重构
具有不同接口的替代类：当两个类的接口不同，但又非常相似时，就会出现这种情况。如果能找到两个类之间的相似之处，通常就能重构这两个类，使它们共享一个共同的接口 [F 85, K 43].	用适配器统一接口 [K 247] 重命名方法 [F 273] 移动方法 [F 142]
组合爆炸：这是一种微妙的重复形式，当众多代码使用不同的数据或行为组合做同样的事情时，就会产生这种味道。[K 45]	用解释器取代隐式语言 [K 269]
评论（又名除臭剂）：当你想写评论时，先尝试 "重构， 让评论变得多余"[法 87]	重命名方法 [F 273] 提取方法 [F 110] 引入断言 [F 267]
条件复杂性：条件逻辑在其萌芽阶段是天真无邪的，因为它简单易懂，只需几行代码即可完成。遗憾的是，条件逻辑很少能很好地与时俱进。当你实现了几个新功能后，你的条件逻辑突然变得 复杂而广阔。[K 41]	引入空对象 [F 260, K 301] 将点缀移至装饰器 [K 144] 用策略取代条件逻辑[K 129] 用状态[K 166]替换状态改变条件句
数据类：具有字段、字段获取和设置方法的类，以及 其他什么都没有。这类类是愚蠢的数据持有者，几乎可以肯定，它们被其他类操纵的细节太多了。[F 86]	移动方法 [F 142] 封装字段 [F 206] 封装集合 [F 208]
数据块：数据集群：数据集群在一起，确实应该成为自己的对象。一个很好的测试方法是考虑删除其中一个数据值：如果你这 样 做了，其他数据值还有意义吗？如果没有意义，那就说明你有一个 即将诞生的物体。[F 81]	提取类 [F 149] 保存整个对象 [F 288] 引入参数对象 [F 295]
分歧变化：当一个类别通常以不同的方式变化时，会出现以下情况 不同的原因。将这些不同的责任分开，可以减少一个变化影响另一个变化的机会，并降低维护成本。[F 79]	提取类 [F 149]
重复代码：重复代码是软件中最普遍、最刺鼻的气味。它往往是显性或隐性的。显性重复存在于相同的代码中，而隐性重复则存在于外表不同但本质相同的结构或处理步骤中。[F76, K 39］	链式构造器 [K 340] 综合摘要 [K 214] 提取法 [F 110] 提取类 [F 149] 表格模板法 [F 345, K 205] 引入空对象 [F 260, K 301] 用工厂方法引入多态创建 [K 88] 拉升法 [F 322] 上拉区域 [F 320] 用 "复合 "取代 "一个/多个 "区分 [K 224] 替代算法[法 139] 用适配器统一接口 [K 247]
功能羡慕：数据和作用于数据的行为是相辅相成的。当一个方法 过多地调用其他类来获取数据或功能，则会引起对功 能 的嫉妒。[F 80]	提取法 [F 110] 移动方法 [F 142] 移动字段 [F 146]
免费班（又称懒惰班）：一个班级如果做得不够好，就应该被淘汰。[F 83, K 43]	折叠层次结构 [F 344] 内联类 [F 154] 内联单例 [K 114］
不适当的亲密接触：有时，课堂会变得过于亲密，花 太 多 时间探究对方的隐私部位。我们也许对人不太谨慎，但我们认为我们的课堂应该遵守严格的清规戒律。过于亲密的班级需要像古代的恋人一样分手。[F 85]	移动方法 [F 142] 移动字段 [F 146] 将双向关联改为单向关联 [F 20 提取类 [F 149] 隐藏代表 [F 157] 用委托取代继承 [F 352]
不完整的库类：当我们的代码中出现明显应转移到库类的职责，但我们无法或不愿修改库类以接受这些新职责时，就会出现这种情况。[F 86]	引进外部方法 [F 162］ 引入本地扩展 [F 164]
不雅暴露：这种气味表明缺乏 David Parnas 著名的信息隐藏[Parnas]。当客户不应该看到的方法或类被公开时，就会产生这种味道。公开这些代码意味着客户知道了不重要或只是间接重要的代码。这有助于 设计的复杂性。[K 42]	用工厂封装类 [K 80］
	提取类 [F 149]

F - 马丁-福勒重构：改进现有代码的设计。K - Kerievsky, Joshua.重构模式。

http://industriallogic.com
Copyright © 2005, Industrial Logic, Inc.
版权所有

大类：Fowler 和 Beck 指出，出现过多的实例变量通常表明一个类试图做的事情太多。一般来说，大类通常包含太多的职责。[F 78, K 44]	提取子类 [F 330]
	提取接口 [F 341]
	用对象替换数据值 [F 175]
	用命令取代条件调度程序 [K 191]
	用解释器取代隐式语言 [K 269]
	用状态[K 166]替换状态改变条件句

F - 马丁-福勒重构：改进现有代码的设计。K - Kerievsky, Joshua.重构模式。

http://industriallogic.com
Copyright © 2005, Industrial Logic, Inc.
版权所有

气味	重构
长法：Fowler 和 Beck 在描述这种味道时，解释了短方法优于长方法的几个充分理由。一个主要原因是逻辑共享。两个长方法很可能包含重复的代码。然而，如果将这些方法拆分成更小的方法，往往可以找到让两个方法共享逻辑的方法。Fowler 和 Beck 还描述了小方法如何帮助解释代码。如果你不明白某段代码是做什么的，而你将其提取到一个命名良好的小方法中，那么就会更容易理解原始代码。拥有大多数小方法的系统往往更容易扩展和维护，因为它们更容易理解，包含的重复内容也更少。[F 76，K 40]	提取法 [F 110] 作曲方法 [K 123] 引入参数对象 [F 295] 将累积值移至采集参数 [K 313] 将积累移至游客 [K 320] 分解条件式 [F 238] 保存整个对象 [F 288] 用命令取代条件调度程序 [K 191] 用策略取代条件逻辑[K 129] 用方法对象取代方法 [F 135] 用查询代替温度 [F 120]
长参数列表：方法中的长参数列表虽然在过程代码中很常见，但很难理解，而且很可能不稳定。考虑一下这个方法真正需要哪些对象来完成它的工作--让这个方法做一些工作来追踪它所需要的数据也是可以的。[F 78]	用方法替换参数 [F 292] 引入参数对象 [F 295] 保存整个对象 [F 288]
信息链：当你看到一长串方法调用或临时 变量来获取一些数据。这一链条使代码依赖于许多可能互不相关的对象之间的关系。[F 84]	隐藏代表 [F 157] 提取法 [F 110] 移动方法 [F 142]
中间人授权是件好事，也是对象的重要基本特征之一。但是 过多的好东西会导致对象不增值，只是把信息传递给另一个对象。[F 85]	移除中间人 [中文本 160] 内联法 [F 117] 用继承取代委托 [F 355]
怪解：当一个问题在整个系统中以一种方式解决，而同一问题在同一系统中又以另一种方式解决时，其中一种解决方案就是怪解或不一致的解决方案。出现这种气味通常表明代码存在微妙的重复。 [K 45]	用适配器统一接口 [K 247]
并行继承层次结构：这其实是 "猎枪手术"（Shotgun Surgery）的一个特例--每 当 你为一个类创建一个子类时，你就必须为另一个类创建一个子类。[F 83]	移动方法 [F 142] 移动字段 [F 146]
迷恋基元基元（包括整数、字符串、双倍、数组和其他低级语言元素）是通用的，因为很多人都在使用它们。而 类 则 可以根据你的需要而具体化，因为你创建它们是为了特定的目的。在许多情况下，类提供了比基元更简单、更自然的建模方式。此外，一旦你创建了一个类，你经常会发现系统中的其他代码是如何属于该类的。Fowler 和 Beck 解释了当代码过于依赖基元时，基元迷是如何表现出来的。这通常发生在你还没有发现更高层次的抽象是如何澄清或简化你的代码时。[F 81, K 41]	用对象替换数据值 [F 175] 用构建器封装复合材料 [K 96] 引入参数对象 [F 295] 提取类 [F 149] 将点缀移至装饰器 [K 144] 用策略取代条件逻辑[K 129] 用解释器取代隐式语言 [K 269] 用复合树取代隐含树 [K 178] 用状态[K 166]替换状态改变条件句 用等级[F 218，K 286]代替类型代码 用State/战略取代类型代码 [F 227] 用子类别替换类型代码 [F 223] 用对象替换数组 [F 186]
拒绝遗赠：这种气味来自于继承了你不想要的代码。而不是 在容忍继承的情况下，你编写代码来拒绝 "遗赠"--这至少会导致代码丑陋、混乱。[F 87]	按下字段 [F 329] 按下法 [F 322] 用委托取代继承 [F 352]
霰弹枪手术：当你仅仅为了添加一个新的或扩展的行为而必须在不同的地方修改大量代码时，这种味道就很明显了。[F 80]	移动方法 [F 142] 移动字段 [F 146] 内联类 [F 154]
解决方案蔓延：当代码和/或用于履行职责的数据分散在多个类中时，解决方案就会出现蔓延。出现这种情况的原因通常是系统中快速添加了一个功能，但却没有花足够的时间简化和整合设计，以最好地适应该功能。[K 43]	将创造知识转移到工厂 [K 68]
投机通用性：当你拥有今天实际上并不需要的通用或抽象代码时，就会产生这种气味。这些代码往往是为了支持未来的行为而存在的，而未来的行为可能需要，也可能不需要。[F 83]	折叠层次结构 [F 344] 重命名方法 [F 273] 删除参数 [F 277] 内联类 [F 154]
开关语句：当同一开关语句（或 "if...else if...else i f "语句）在整个系统中重复出现时，就会产生这种味道。这种重复的代码表明缺乏面向对象的意识，错失了利用多态性的良机。[F	将积累移至游客 [K 320] 用命令取代条件调度程序 [K 191] 用多态性取代条件性 [F 255] 用子类别替换类型代码 [F 223] 用State/战略取代类型代码 [F 227] 用显式方法替换参数 [F 285]

F - 马丁-福勒重构：改进现有代码的设计。K - Kerievsky, Joshua.重构模式。

http://industriallogic.com
Copyright © 2005, Industrial Logic, Inc.
版权所有

82, K 44]	引入空对象 [F 260, K 301]
临时字段：对象有时包含一些似乎并不总 是 需要的字段。在其他时间，字段是空的，或者包含难以理解的无关数据。这通常是长参数列表的替代方法。[F 84]	摘录类 [F 149] 引入空对象 [F 260, K 301]

F - 马丁-福勒重构：改进现有代码的设计。K - Kerievsky, Joshua.重构模式。

http://industriallogic.com
Copyright © 2005, Industrial Logic, Inc.
版权所有