目录

•不良设计的特征(Design Smells)

- 腐烂的软件的气味:
  - 刚性
  - 脆弱性
  - 不动性
  - 粘性
  - 不必要的复杂性
  - 不必要的重复
  - 不透明度

气味 - 刚性

- 因为每一次改变都会迫使系统其他部分发生许多其他变化，系统是很难改变的。
- 如果单个更改导致依赖模块的后续更改级联，则设计是严格的。
- 必须改变的模块越多，设计越严格。
- 根本原因是模块化不好，耦合度高。

耦合性

- 耦合描述了一个对象如何依赖另一个对象（它使用的）。 松散耦合的对象可以根本地改变，而不会相互影响。 对紧密耦合的对象稍作修改可能会导致一系列问题。

耦合性

- 高耦合问题：
  - 更改一个模块会强制更改其他模块
  - 模块难以单独理解
  - 模块很难重用或测试，因为必须包含依赖模块。
- 目标是低耦合

例子

```
class A {
int x;
...
}
class B extends A {
void b() {
x = 5;
}
}
```

耦合

- 对类和继承的强调会导致某些类型的不好的耦合。
- Erich Gamma (GoF): “赞成继承类继承的对象组合。”
- Erich Gamma (GoF): “编程接口，而不是实现。”

气味- 脆弱性

- 更改会导致系统在与已更改部分没有概念关系的地方中断。
- 与僵化密切相关（相同的根本原因）
- 管理人员（现在也是开发人员）会担心变化
- 脆弱性趋于恶化，软件无法维护

气味 - 不动性

- 很难将系统分解成可以在其他系统中可重用的组件。
- 因为模块没有设计用于重用，可能会发生。
- 例如：当模块依赖于基础设施或模块太专门化时。与**低凝聚力**有关
- 其结果是软件被重写而不是重用。

气味 - 粘性

- 做正确的事情比做错事更困难。
- 两种形式：**设计的粘度或环境的粘度**。
- 如果进行保留设计的变更比做“黑客”更难，那么设计的粘度就会很高。
- 当开发环境缓慢且效率低下时，环境粘度就会产生。

气味 – 不必要的复杂性

- 设计包含的基础设施不会带来直接的好处。
- 当开发人员预测需求发生变化并为这些潜在变化投入实施时，这种情况经常发生。
- 该设计将承载所有未使用的设计元素的重量，并可能使其他更改变得困难。

气味 – 不必要的重复

- 该设计包含可在单一抽象下统一的重复结构。
- 剪切和粘贴的结果
- 所有重复都不好！
- 请注意半重复，代码几乎相同。 更糟糕的是要解决。
- 使软件难以维护

气味 - 不透明度

- 代码很难阅读和理解。 它没有很好地表达它的意图。
- 不遵循编码标准
- 错误或不一致的命名
- 糟糕或缺乏评论
- 模块太大
- 应该进行某种代码审查以避免不透明的代码。

什么刺激了软件的腐烂？

- 需求总是改变！
- 糟糕的设计！
- 思维短浅！