编写卓越的代码：如何保障软件质量

配合：软件测试和质量保障课程

- 软件质量属性
- 软件质量问题分类
- 常见代码质量问题、原因及解决方法
- 提高代码质量的基本方法

课前准备

内容	负责人	备注
准备分析用的代码	老师和同学	

代码重构

https://www.refactoring.com/catalog/

https://github.com/tobyweston/Refactoring-Chapter-1

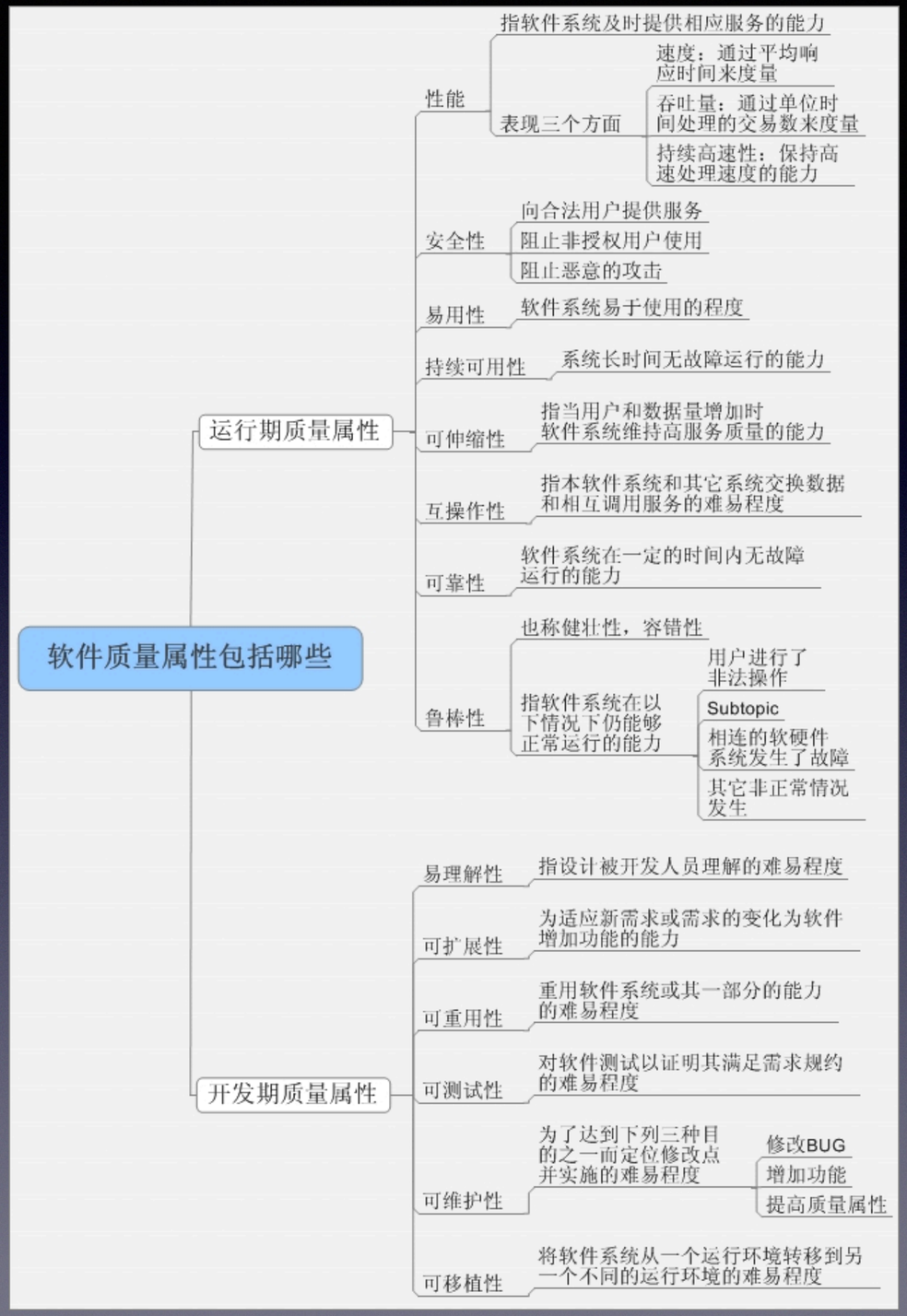
Java代码规范

- https://google.github.io/styleguide/javaguide.html
- https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

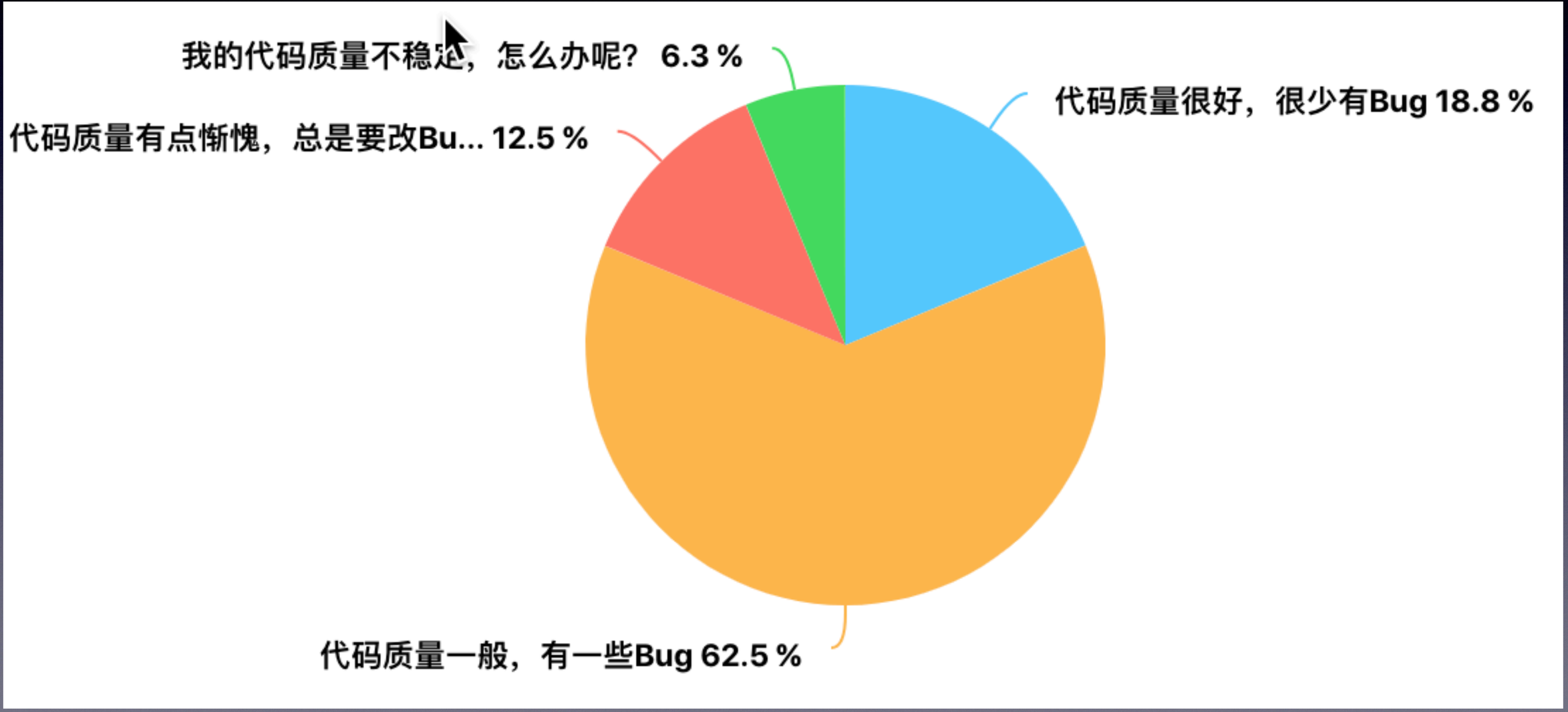
Python 代码规范

- https://www.python.org/dev/peps/pep-0008/

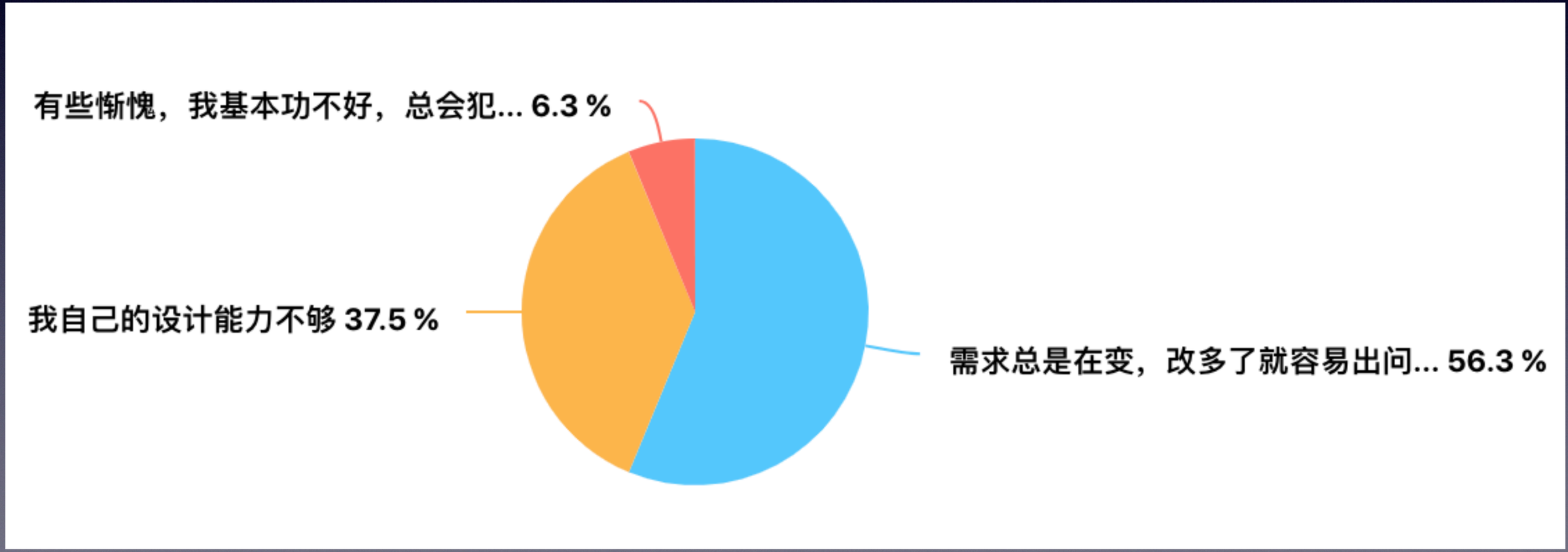
软件质量属性


软件质量属性包括哪些
运行期质量属性
性能
指软件系统及时提供相应服务的能力
表现三个方面
速度：通过平均响应时间来度量
吞吐量：通过单位时间处理的交易数来度量
持续高速性：保持高速处理速度的能力
安全性
向合法用户提供服务
阻止非授权用户使用
阻止恶意的攻击
易用性
软件系统易于使用的程度
持续可用性
系统长时间无故障运行的能力
可伸缩性
指当用户和数据量增加时软件系统维持高服务质量的能力
互操作性
指本软件系统和其它系统交换数据和相互调用服务的难易程度
可靠性
软件系统在一定的时间内无故障运行的能力
鲁棒性
也称健壮性，容错性
指软件系统在以下情况下仍能够正常运行的能力
用户进行了非法操作
Subtopic
相连的软硬件系统发生了故障
其它非正常情况发生
开发期质量属性
易理解性
指设计被开发人员理解的难易程度
可扩展性
为适应新需求或需求的变化为软件增加功能的能力
可重用性
重用软件系统或其一部分的能力的难易程度
可测试性
对软件测试以证明其满足需求规约的难易程度
可维护性
为了达到下列三种目的之一而定位修改点并实施的难易程度
修改BUG
增加功能
提高质量属性
可移植性
将软件系统从一个运行环境转移到另一个不同的运行环境的难易程度

代码质量分布


我的代码质量不稳定，怎么办呢？ 6.3 %
代码质量很好，很少有Bug 18.8 %
代码质量有点惭愧，总是要改Bu... 12.5 %
代码质量一般，有一些Bug 62.5 %

代码质量不好的原因



常见代码质量问题、原因及解决方法

- 问题?
- 原因?
- 解决办法?

常见代码质量问题、原因及解决方法

- 输入输出错误
- 逻辑错误
- 计算错误
- 接口错误
- 数据错误
- https://en.wikipedia.org/wiki/Software_bug

输入输出错误

- 格式
- 数值
- 数据类型

逻辑错误

- Infinite loops and infinite recursion.
  - Off-by-one error, counting one too many or too few when looping.

计算错误

- Division by zero.
  - Arithmetic overflow or underflow.
  - Loss of arithmetic precision due to rounding or numerically unstable algorithms.

接口错误

- Incorrect API usage.[35]
- Incorrect protocol implementation.
- Incorrect hardware handling.
- Incorrect assumptions of a particular platform.
- Incompatible systems.

数据错误

多线程错误

- Deadlock, where task A can't continue until task B finishes, but at the same time, task B can't continue until task A finishes.
- Race condition, where the computer does not perform tasks in the order the programmer intended.
- Concurrency errors in critical sections, mutual exclusions and other features of concurrent processing. Time-of-check-to-time-of-use (TOCTOU) is a form of unprotected critical section.

常见代码质量问题、原因及解决方法

- 代码基本功差
- 设计能力差
- 需求变更太频繁

提高代码质量的基本方法

- 防御性的编码技巧
- 如何有效地调试程序
- 代码的团队合作技巧
- 管理源代码

提高代码质量的基本方法?

防御式的编程技巧

不是

- 检查错误
- 测试
- 调试： 补救措施

是

- 细致、谨慎的编程方法
- 防止代码以将会展现错误行为的方式被调用

防御式的编程技巧

- ▶使用好的编码风格和合理的设计
- ▶不要仓促地编写代码、磨刀不误砍柴工
- ▶增加安全检查
- ▶写清晰的代码而非简洁的代码： 复杂的代数运算拆分为一系列简单的语句
- ▶审慎地处理内存和其他资源:
- ▶ Java .NET 循环引用
- ▶在声明位置初始化所有变量
- ▶尽可能推迟一些声明变量

有效地调试程序

- 原始类：通过计算机找错，依赖大量的现场信息，从而找到错误线索
- 回溯类：从出现错误征兆处开始，人工地沿控制流程向回追踪，直至出错根源
- 排除类：基于归纳和演绎原理 采用分治的概念

什么样的人写什么样的代码。

好大喜功的爱抽象，

喜欢冒险的不屑处理异常，

脑子混乱的常出死循环，

强迫症就看变量命名，

偷懒的藕合度极高。

每每回顾自己写的代码，仿佛开批斗会，经常提醒我是一个怎么垃圾的人。

代码的团队合作技巧

- 沟通
- 谦卑
- 处理冲突
- 学习和适应
- 了解不足

代码的团队合作工具

- 源代码控制
- 缺陷数据库
- 协同工作系统
- 方法
- 项目计划

代码的团队合作原则

- ▶集体代码所有制
- ▶尊重他人的代码
- ▶制定编码准则
- ▶定义成功
- ▶定义责任

管理源代码