Ch2 Code Unit Testing

Write Code to Test Code(1)

Instructor: Haiying SUN

E-mail: hysun@sei.ecnu.edu.cn

Office: ECNU Science Build B1104

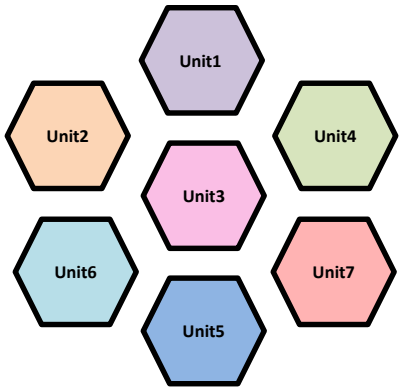
Available Time: Wednesday 8:00 -12:00 a.m.

Agenda

- Introduction to Unit Testing
- Common Code Defect Categories
- Unit Tests Design Heuristic Rules
- Unit Tests Implementation
  - Junit & Mockito & Qualified test scripts
- Code Test Adequacy Criteria
  - Control flow based & Jacoco
  - Data flow based
  - Mutation Based
- Code Test Generation

Dynamic Code Test

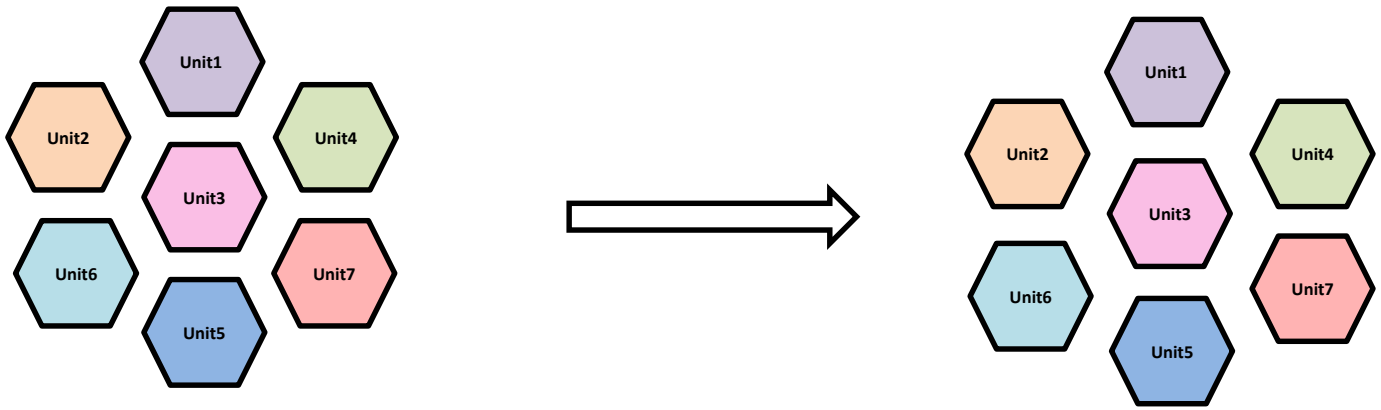
- 在开发环境中，通过运行被测代码以验证其是否满足期望目标而进行的一种测试活动以尽早尽可能发现与目标不一致的缺陷



Unit Test

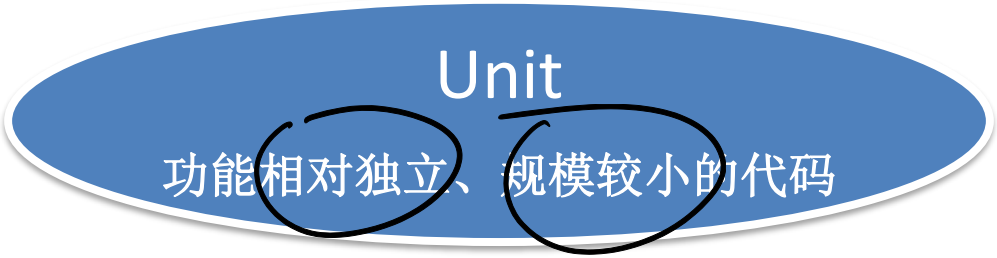
Dynamic Code Test

- 在**开发环境**中，通过**运行被测代码**以验证其**是否满足期望目标**而进行的一种测试活动以尽早尽可能发现**与目标**不一致的缺陷



Unit Test　　Integration Test

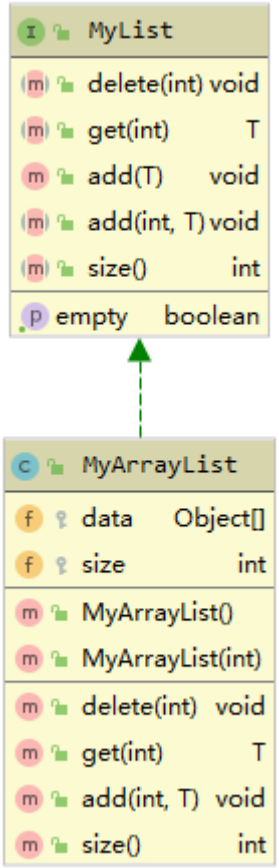
What is a Unit



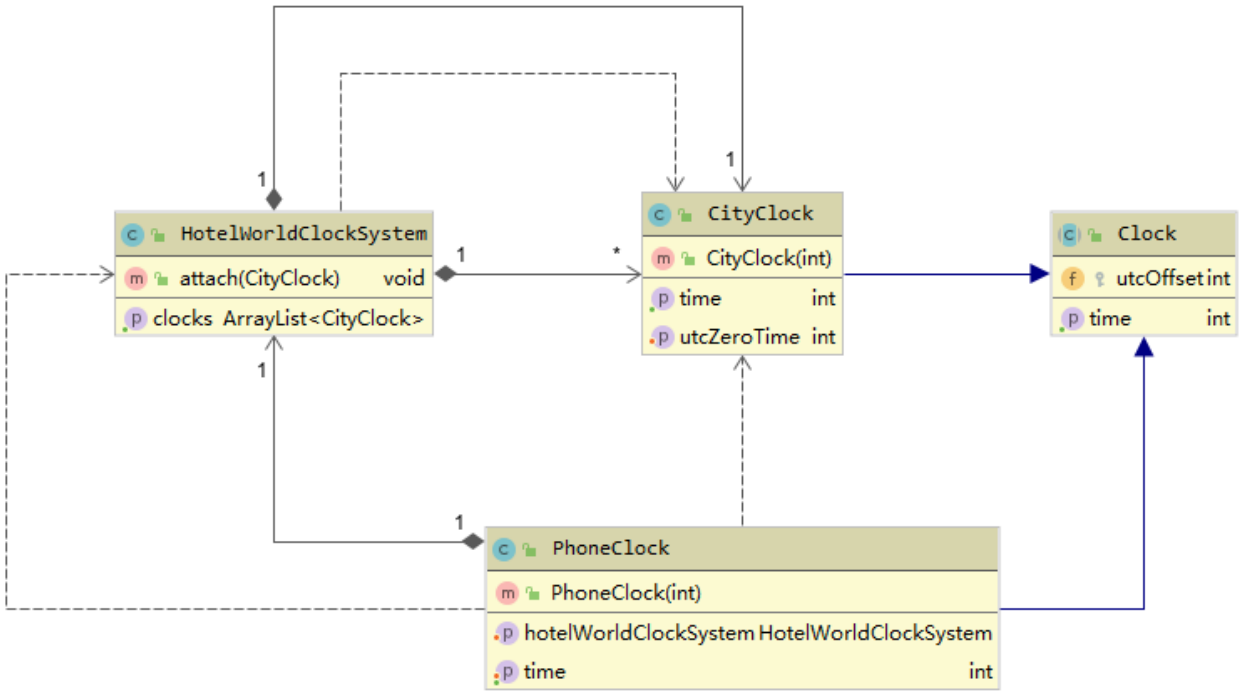
单元是一个小粒度的行为特性。从规模上说可能是一个函数、方法、页面，也可能是几个。

例如，向购物车里加一件商品是一个功能特性，对应的单元测试需要测试购物车里商品数量会增加1且购买的商品在购物车中等等。

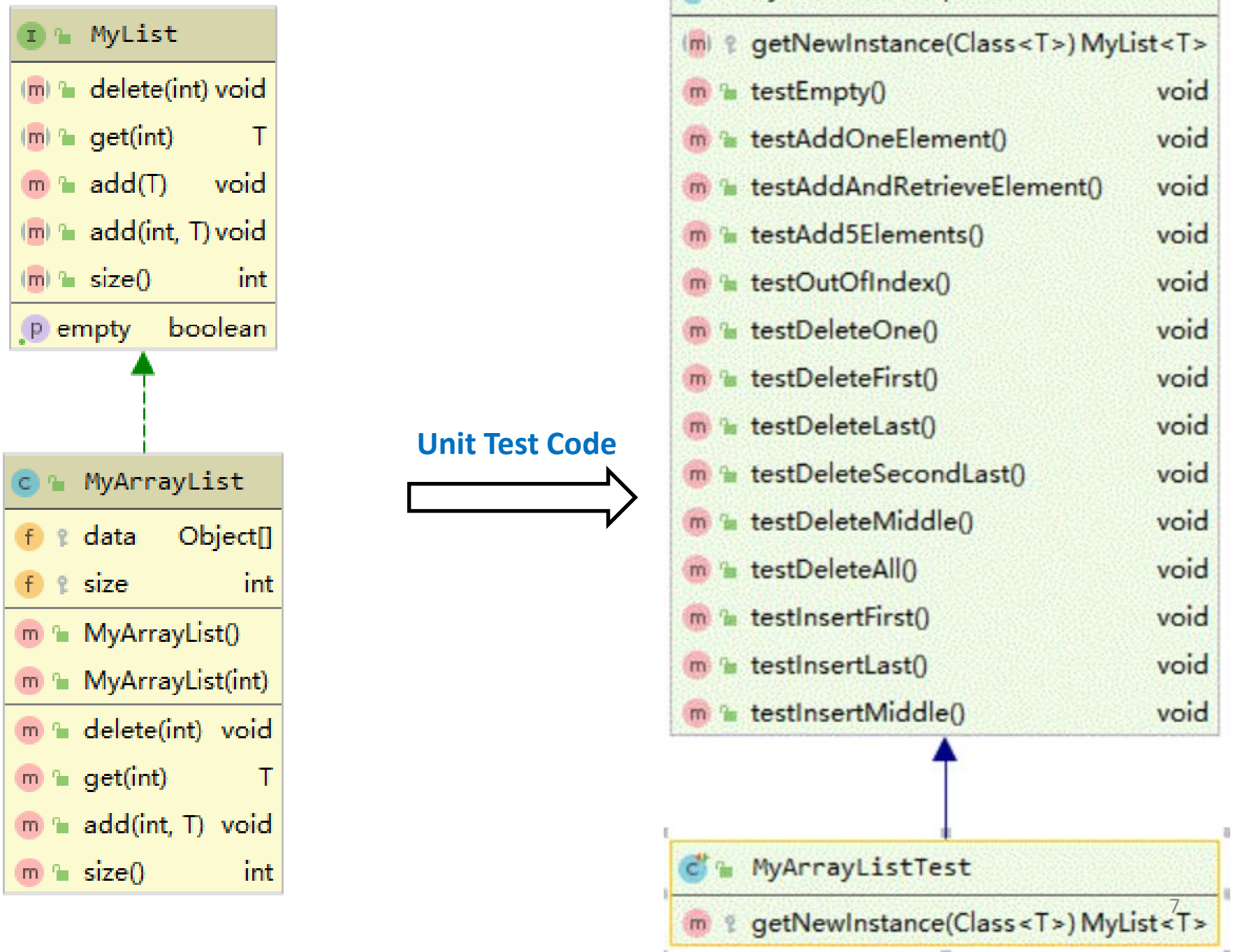
Example

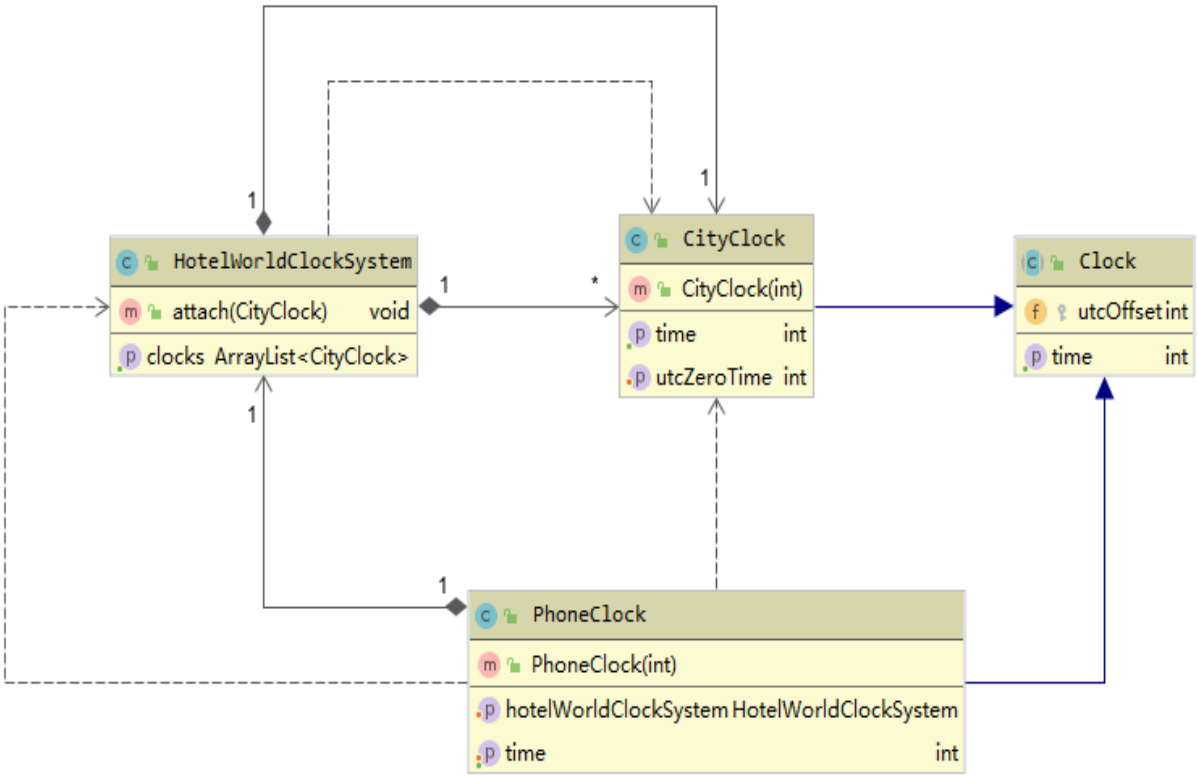


ArrayList

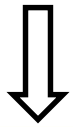


酒店时钟系统：根据手机时间调整酒店前台上展示的世界各地时钟的时间


MyList
delete(int) void
get(int) T
add(T) void
add(int, T) void
size() int
empty boolean
MyArrayList
data Object[]
size int
MyArrayList()
MyArrayList(int)
delete(int) void
get(int) T
add(int, T) void
size() int
Unit Test Code
MyListTestTemplate
getNewInstance(Class<T>) MyList<T>
testEmpty() void
testAddOneElement() void
testAddAndRetrieveElement() void
testAdd5Elements() void
testOutOfIndex() void
testDeleteOne() void
testDeleteFirst() void
testDeleteLast() void
testDeleteSecondLast() void
testDeleteMiddle() void
testDeleteAll() void
testInsertFirst() void
testInsertLast() void
testInsertMiddle() void
MyArrayListTest
getNewInstance(Class<T>) MyList<T>



Unit Test Code



HotelWorldClocksTest	
hotelWorldClockSystem	HotelWorldClockSystem
phoneClock	PhoneClock
initialize()	void
the_time_of_clock_London_should_be_1_after_the_phone_clock_is_set_to_9_Beijing_time()	void
the_time_of_clock_NewYork_should_be_20_after_the_phone_clock_is_set_to_9_Beijing_time()	void
the_time_of_clock_London_and_NewYork_should_be_1_and_20_respectively_after_the_phone_clock_is_set_to_9_Beijing_time()	void
the_time_of_the_phone_clock_should_be_set_correctly_after_its_setTime_method_is_invoked()	void
the_time_of_clock_Moscow_should_be_5_after_the_phone_clock_is_set_to_9_Beijing_time()	void

Unit Test

测试替身

测试执行是否经济快捷界定单元测试，运行要**快，不超过0.1秒**

不是单元测试：

① 跟数据库交互

② 进行了网络间通信

③ 调用了文件系统

④ 需要对环境做特定的准备（如编辑配置文件）才能运行起来

测试替身（Test Double）

替代真实代码中依赖于数据库、网络和文件系统的代码

Benefits of Code Testing

Quick feedback

Automated Regression Checking

Design Aid

Improve confidence

Documentation

Difficulties in Unit Test

- The complexity of test inputs
- The complexity of test outputs
- Unavailable of related codes

The Complexity of Test Inputs

```java
public class Ch2 {

    private static int classAtt = 0;
    private boolean att1;
    private int att2;

    public void WhoAreTestInputs(int para1, int para2) {

        boolean result = method1(para1);
        int att2 = method2(para2);
        if ((att1) && (result) && (att2 >= 0)) {
            classAtt++;
            if (classAtt == 2)
                System.out.println("Tea time!");
        }
    }

    private boolean method1(int a) {
        return false;
    }

    private int method2(int b) {
        return 0;
    }
}
```

【问题】对于WhoAreTestInputs而言，哪些变量属于测试输入需要考虑的范畴？

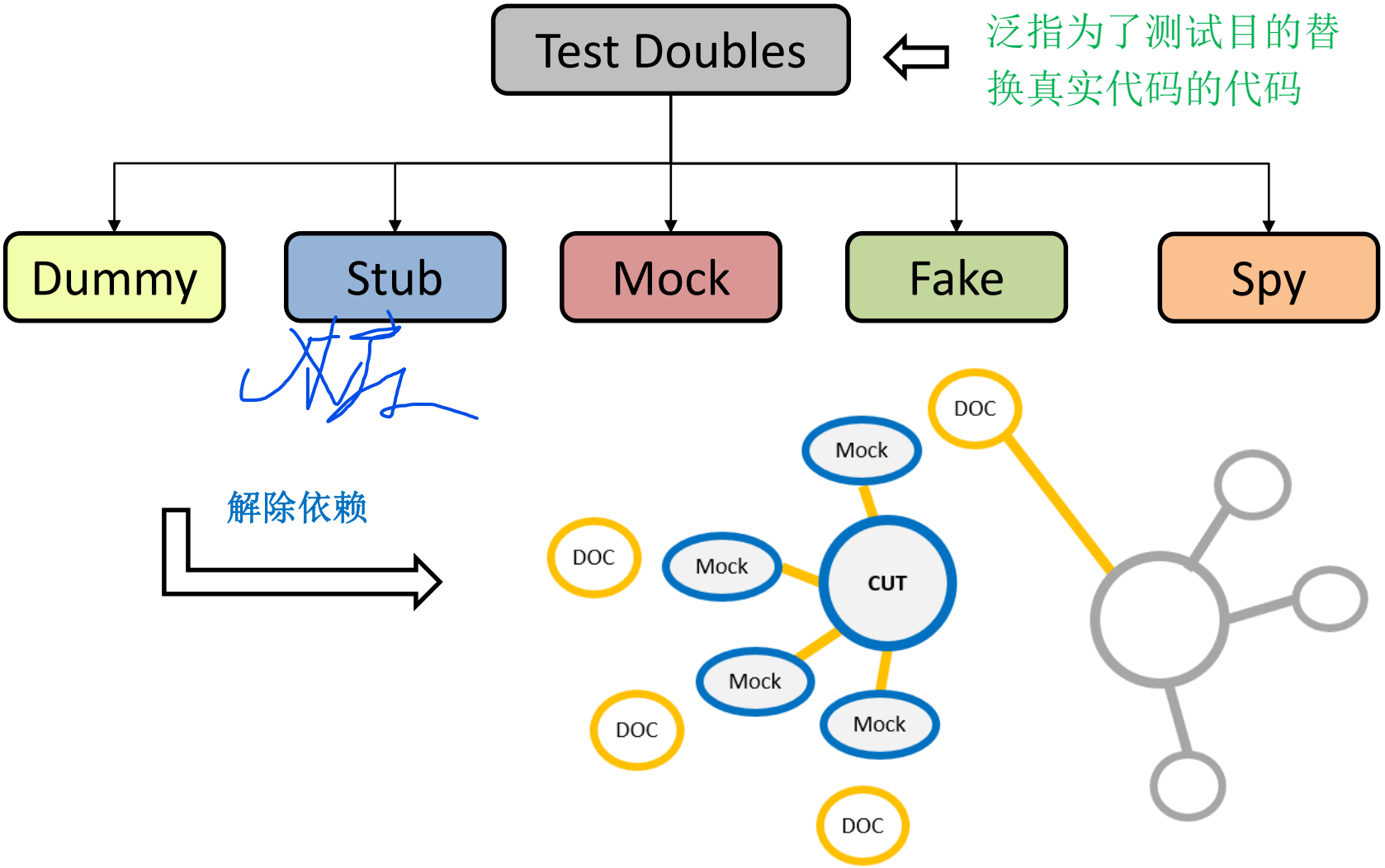
- ✓ 被测代码的输入参数
- ✓ 被测代码内部需要读取的全局变量
- ✓ 被测代码内部需要读取的成员变量
- ✓ 被测代码内部调用的方法/函数获得的数据
- ✓ 被测代码内部调用的方法改写的数据
- ✓ ……

The Complexity of Test Outputs

- 预期输出仅仅只有返回值吗？？
  - ✓ 被测代码的返回值
  - ✓ 被测代码的输出参数（C/C++）
  - ✓ 被测代码内部改写的成员变量和全局变量
  - ✓ 被测代码进行的文件更新、数据库更新、消息队列更新
  - ✓ ……

不变式

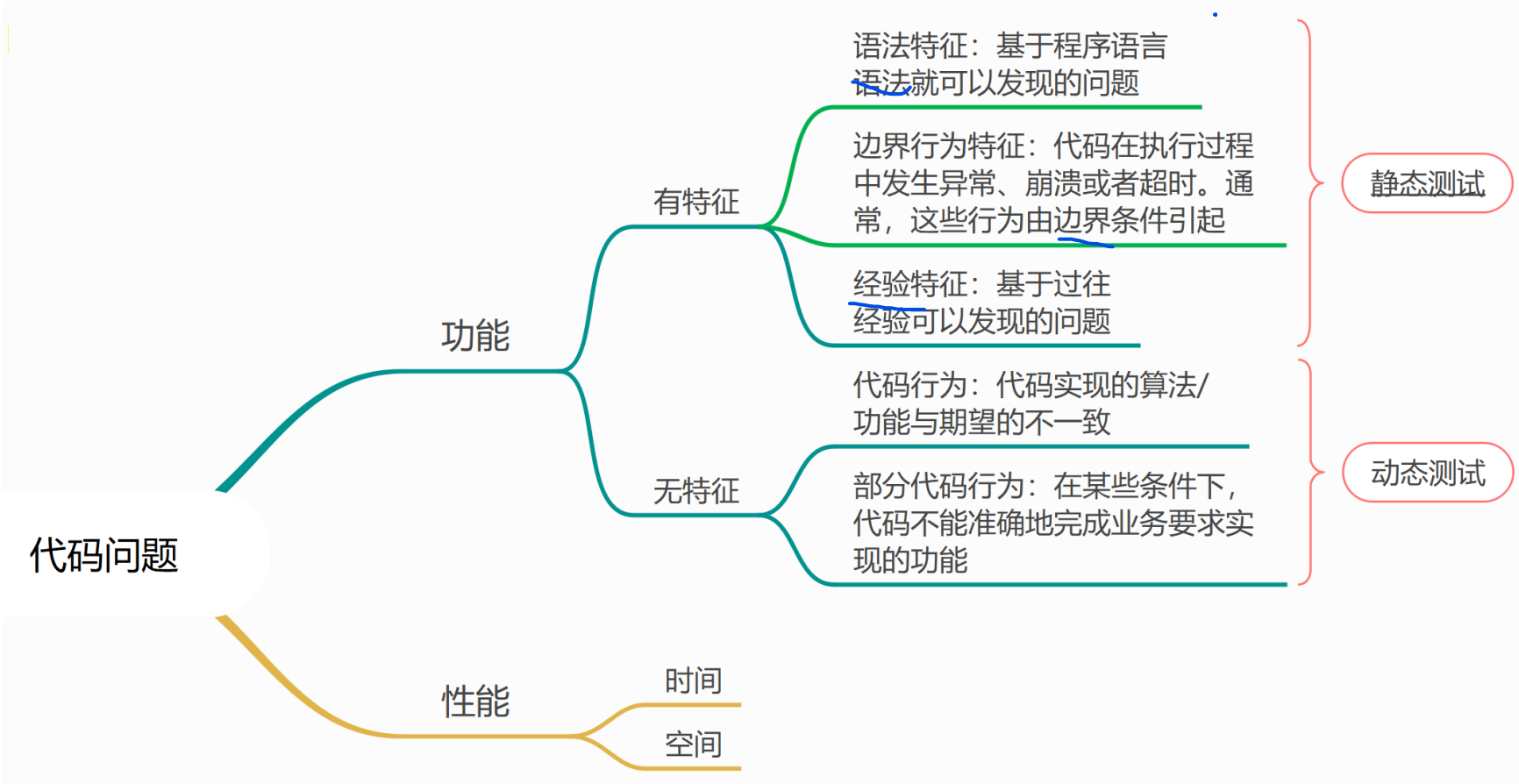
Unavailable of Related Codes



Agenda

- Introduction to Unit Testing
- Common Code Defect Categories
- Unit Tests Design Heuristic Rules
- Unit Tests Implementation
  - Junit & Mockito & Qualified test scripts
- Code Test Adequacy Criteria
  - Control flow based & Jacoco
  - Data flow based
  - Mutation Based
- Code Test Generation

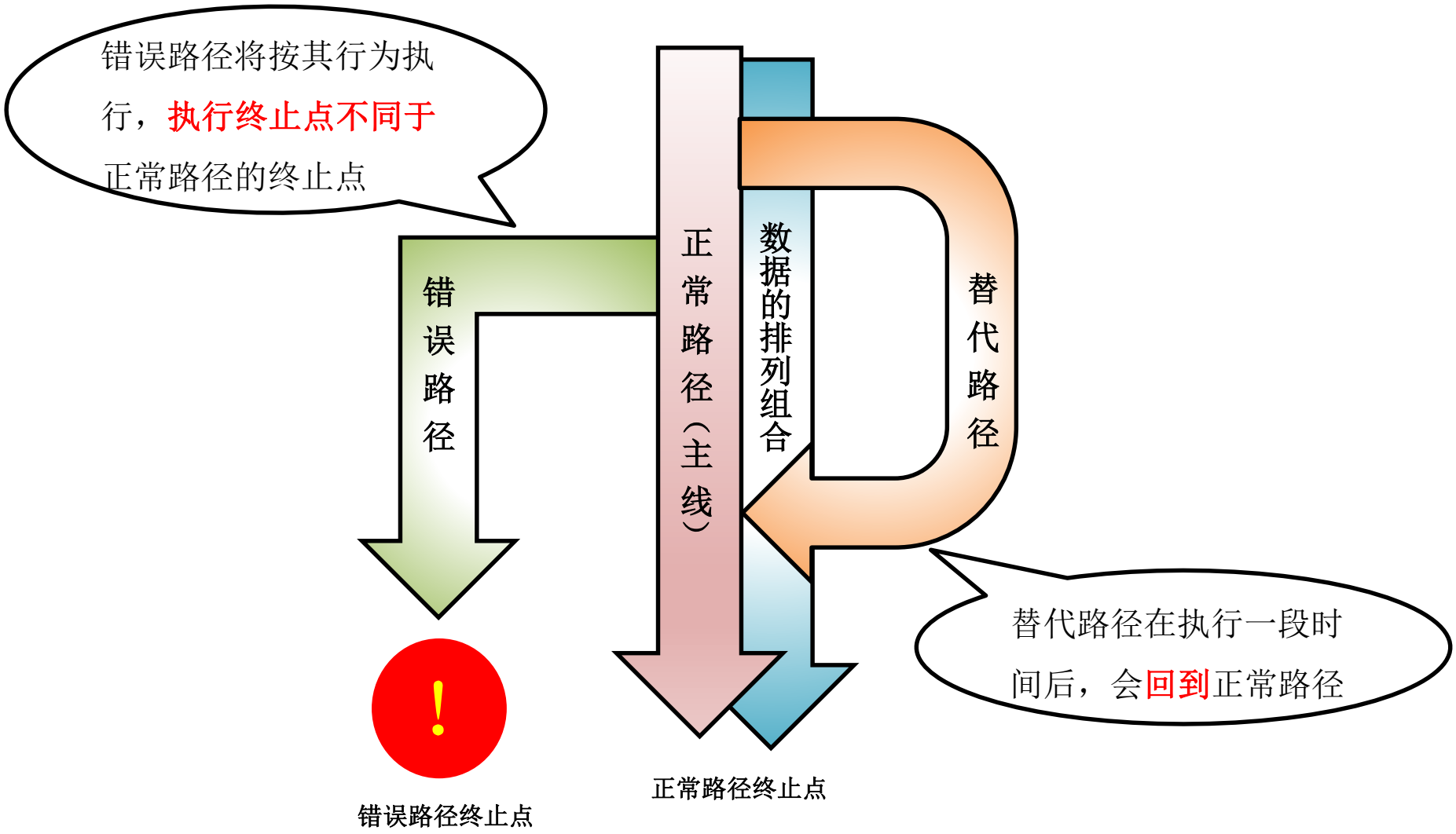
Code Defect Categories



Agenda

- Introduction to Unit Testing
- Common Code Defect Categories
- Unit Tests Design Heuristic Rules
- Unit Tests Implementation
  - Junit & Mockito & Qualified test scripts
- Code Test Adequacy Criteria
  - Control flow based & Jacoco
  - Data flow based
  - Mutation Based
- Code Test Generation

Dynamic Testing Strategy



Practical Heuristic Rules Example

阿里巴巴 Java 开发手册

9.【推荐】编写单元测试代码遵守 BCDE 原则，以保证被测试模块的交付质量。

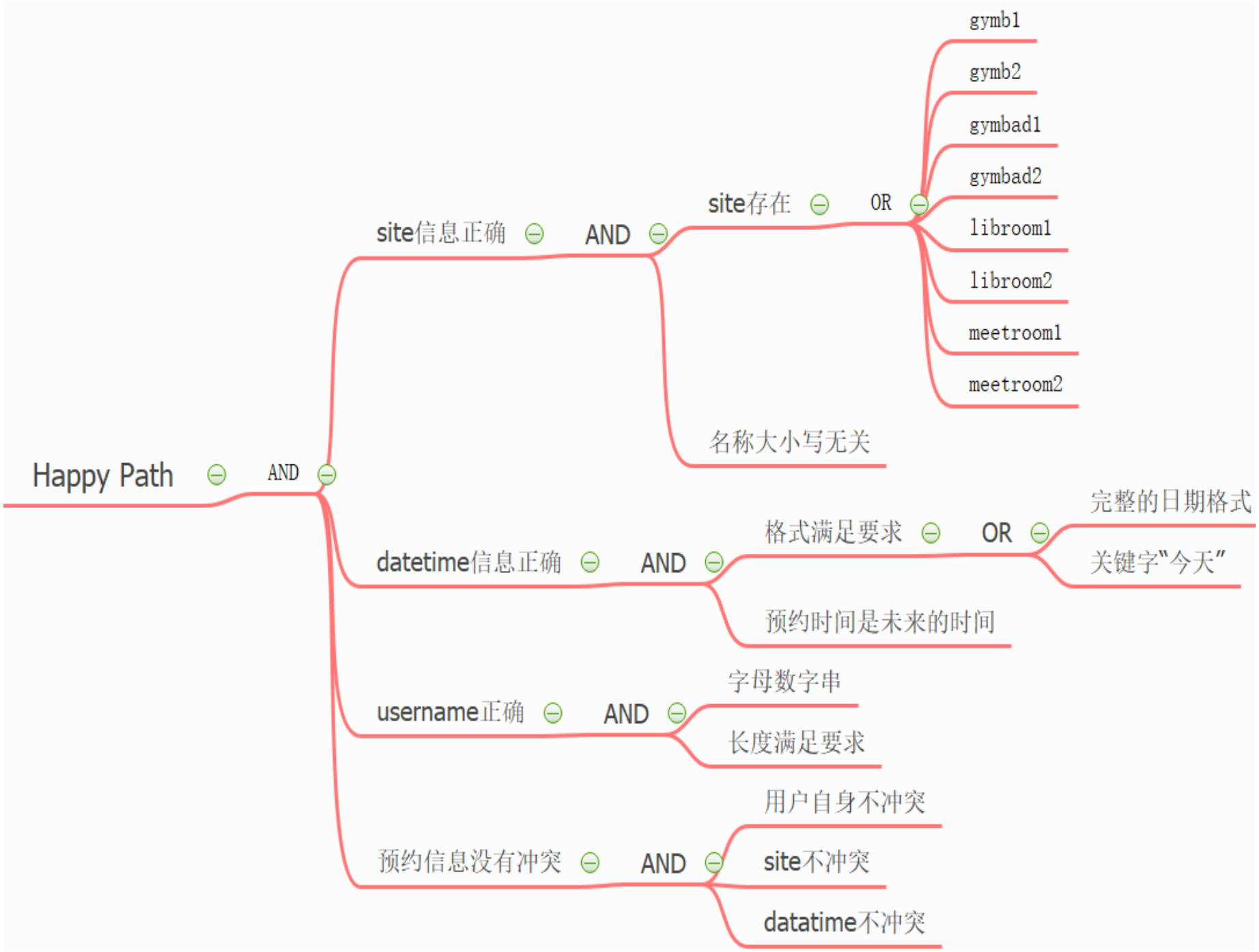
- B：Border，边界值测试，包括循环边界、特殊取值、特殊时间点、数据顺序等。
- C：Correct，正确的输入，并得到预期的结果。
- D：Design，与设计文档相结合，来编写单元测试。
- E：Error，强制错误信息输入（如：非法数据、异常流程、非业务允许输入等），并得到预期的结果。

Testing Strategy

- Right—BICEP
  - Right: Are the results right?
  - B: Are all the boundary conditions correct?
  - I: Can you check inverse relationships?
  - C: Can you cross-check results using other means?
  - E: Can you force error conditions to happen?
  - P: Are performance characteristics within bounds?

Right-BICEP

- Right—BICEP
  - happy path tests: tests should first and foremost validate that the code produces expected results what the users want.
  - If the code ran correctly, how would I know?
- Exercise
  - MeetCalendar.addReservation的Happy Path？


Happy Path
AND
site信息正确
AND
site存在
OR
gymb1
gymb2
gymbad1
gymbad2
libroom1
libroom2
meetroom1
meetroom2
名称大小写无关
datetime信息正确
AND
格式满足要求
OR
完整的日期格式
关键字"今天"
预约时间是未来的时间
username正确
AND
字母数字串
长度满足要求
预约信息没有冲突
AND
用户自身不冲突
site不冲突
datatime不冲突

Right-BICEP

- Right—BICEP: Boundary conditions
  - Bogus or inconsistent input values, a filename like: "!*W:X\&Gi/w$→>$g/h#WQ@.
  - Badly formatted data, bad phone number, hysun@ecnu.edu.cn
  - Computations that can result in numeric overflow.
  - Empty or missing values: 0, 0.0, "", null.

Right-BICEP

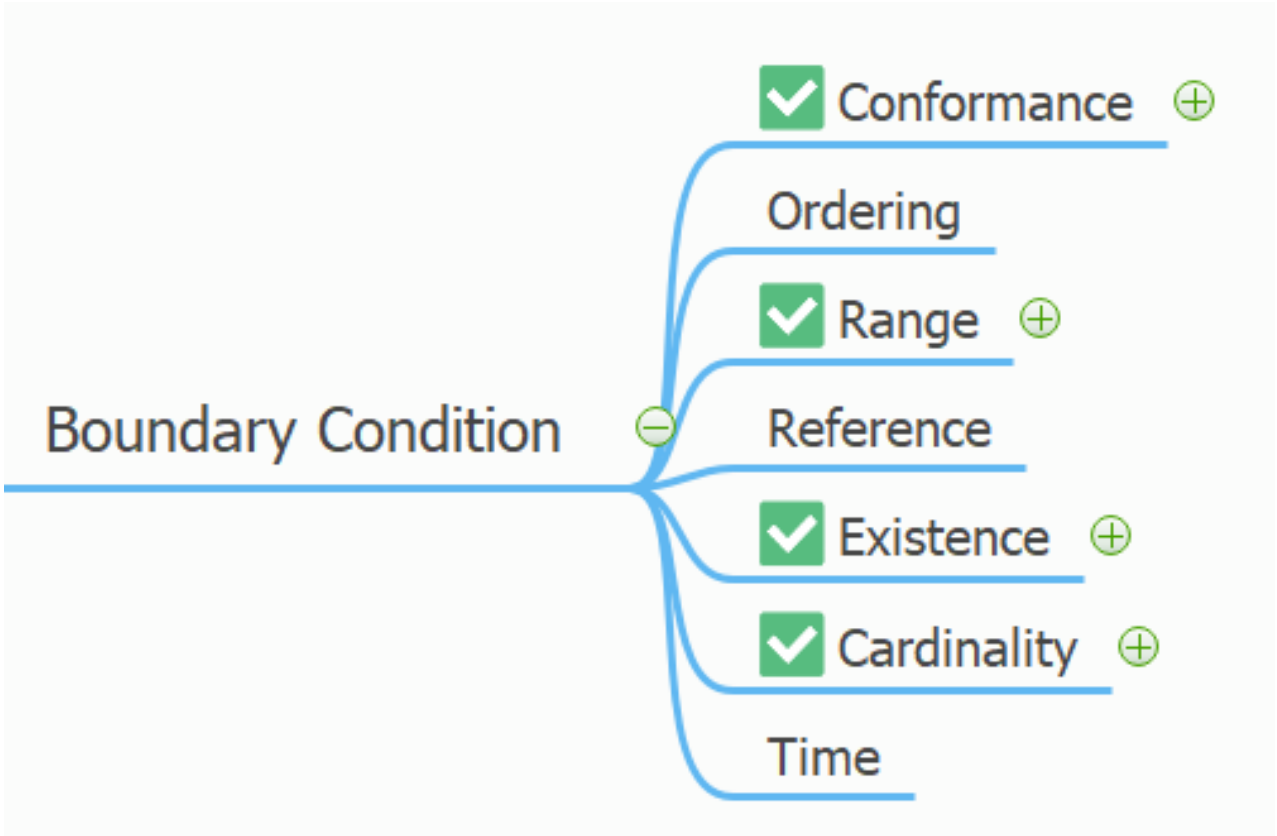
- Values far in excess of reasonable expectations
- Duplicates in lists that shouldn’t have duplicates
- Ordered lists that aren’t, and vice versa. Try handing a presorted list to a sort algorithm.
- Things that happen out of expected chronological order.

Corner Case Testing

- 每个被测对象使用CORRECT启发式规则：
  1. 是否涉及这些条件
  2. 当条件被违背时会发生什么
  - Conformance: 数据格式是否与期望的一致
  - Ordering：数据之间的顺序是否满足要求
  - Range：数据是否在合理的最大值和最小值之间
  - Reference：被测代码是否使用了无法控制的外部引用
  - Existence：数据是否被要求存在，例如非空,非0,必须在集合中
  - Cardinality：数据数量是否满足要求
  - Time：每件事情是否按顺发生？是否在正确的时间发生？发生是否及时？

Corner Case Testing

- Exercise: addReservation涉及哪些规则?

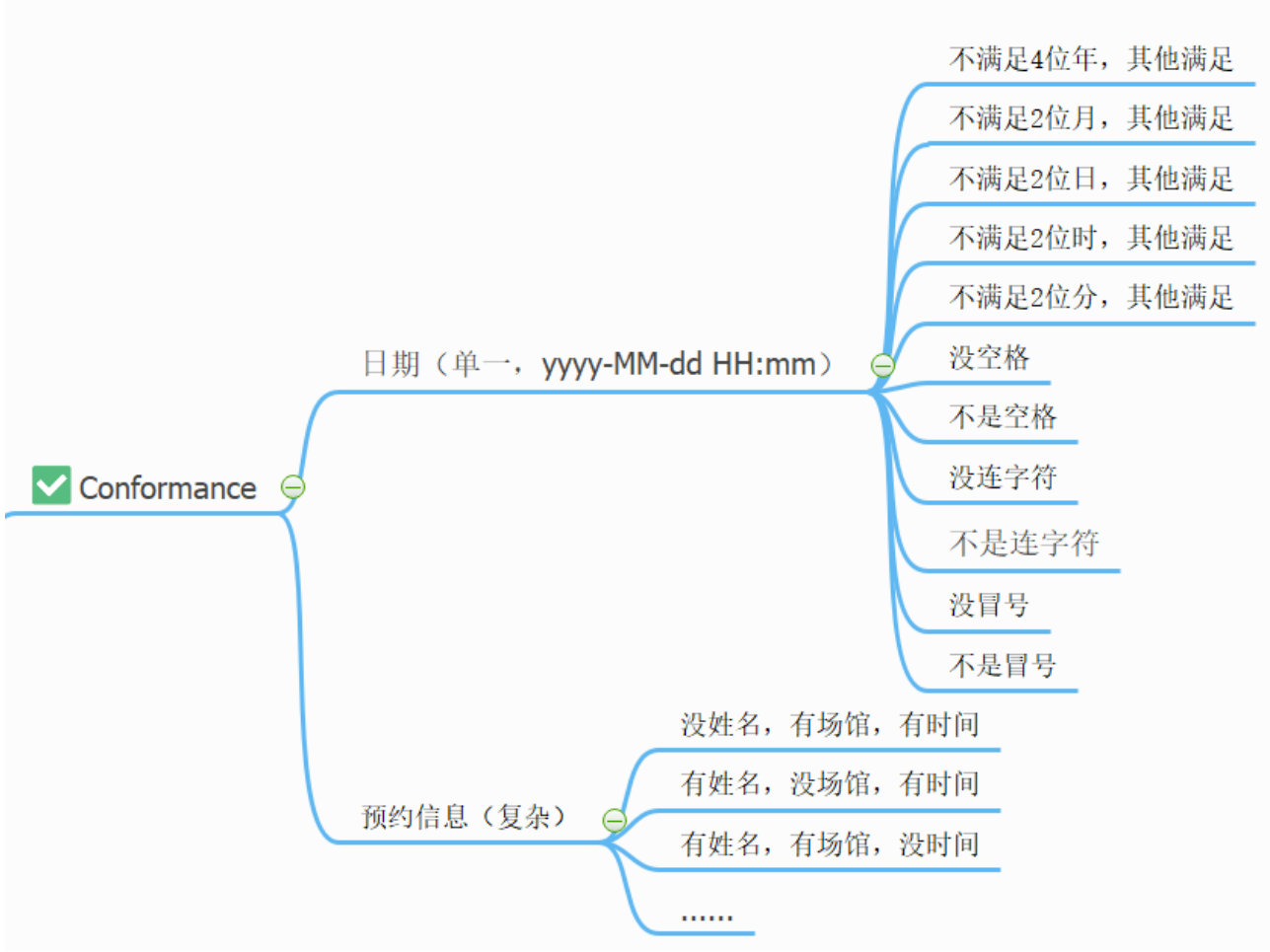


What else can go wrong

- CORRECT: Conformance
  - 单一结构的数据
    - E-mail，phone number.....
  - 复合结构的数据
    - （姓名，场地，时间）：只没有姓名/场地/时间，只有姓名/场地/时间.....
  - 确定数据什么时候进入系统，有利于测试设计
    - UI层就确定了，那么测试就会简单

Corner Case Testing

- Exercise: addReservation的Conformance



What else can go wrong

- CORRECT: Ordering
  - the position of one piece of data within a larger collection
- CORRECT: Range
  - Java primitive types (primitive obsession： age，salary， score)
  - customized range
  - Invariant assertion

Corner Case Testing

- Exercise: addReservation的Range?

What else can go wrong

- CORRECT: Reference
  - May consider:
    1. What method under test（MUT） references outside its scope
    2. What external dependencies MUT has
    3. Whether MUT depends on the object being in a certain state
    4. Any other conditions that must exist
  - precondition
  - postcondition
  - side effect

What else can go wrong

- CORRECT: Existence： Dose some given thing exist?
  - what will happen if the value is null, zero, or otherwise empty, especially check value returned
  - a special case of cardinality
- CORRECT: Cardinality（护栏柱测试）
  - the count of some set of values is only interesting in these three cases (0-1-n rule)
    - Zero
    - One
    - Many (more than one)

Corner Case Testing

- Exercise: addReservation的Existence?

What else can go wrong

- CORRECT: Time
  - Relative time (ordering in time) : if methods are called out of order
  - Absolute time (elapsed and wall clock): test any time-sensitive code on boundary days
  - Concurrency issues

1. what will happen if multiple threads access the same object at the same time?
2. Do you need to synchronize any global or instance-level data or methods?
3. How about external access to files or hardware?

Exercise

- 请完成完整的addReservation测试代码

Right-BICEP

- Right—BICEP: Checking Inverse Relationship
  - Seek an independent means of verification.
    - 用“逆行为”测试被测试代码
    - 在数据库中插入一条记录后，查询该记录
    - 已经使用的款项总数 = 款项总数 – 剩余的款项数
- Right—BICEP: Cross checking
  - 使用不同数据之间的关系进行测试
    - 已经使用的款项总数 = 款项总数 – 剩余的款项数

Exercise

- addReservation中是否存在inverse relationship或者cross-check？
  - 每个场地每天的预定数是有上限的

Right-BICEP

- Right—BICEP: Forcing Error Condition
  - think about what kinds of errors or other environmental constraints :
    1. destroy business rules
    2. Running out of memory
    3. Running out of disk space
    4. Network availability and errors
    5. System load
    6. Very high or very low video resolution

    ……
  - 使用Mock对象模拟各种异常

Right-BICEP

- Right—BICEP: Performance characteristics
  - Usually applied to E2E testing
  - Take as baseline information for changing
  - Junit 5 @Timeout

```
@Test
@Timeout(value = 100, unit = TimeUnit.MILLISECONDS)
void failsIfExecutionTimeExceeds100Milliseconds() {
    // fails if execution time exceeds 100 milliseconds
}
```

Summary

- Unit is defined by function, size and time
- Unit testing is to write code to test code
- There are six common code defect categories. Different defect categories need different testing methods
  1. Syntax related
  2. Boundary behavior related
  3. Experience related
  4. Code behavior related
  5. Partly code behavior related
  6. Performance related
- Right—BICEP are unit test design heuristic rules adopted in practice

The End