# Args

## 技能点

1.单元测试

2.任务拆分

3.测试驱动开发

4.面向对象设计

5.重构

## 能力目标

1.用测试用例描述和沟通需求

2.把需求拆解成可以逐步开发的任务列表

3.基本的面向对象设计

4.识别代码坏味道

5.重构的基本节奏和原则

## 做题要求

  1. 看懂题目，开始编码之前先花10分钟拆解任务，把任务清单写下来

// Items

# 实际参数列表 - list

· 元素为item

# 参数结构 - scheme

// TODO

> 根据list的特征（“-”），拆分成项items；

[

\* 测试用例，测试拆分函数；

> 定义Parser类，转换函数splitArgs()

]

> 循环遍历list

针对每个item（如”p 8080”）的特征(空格“ ”)，取flag标记，转换值（或默认值）及值类型

> 非法item，提示错误信息

[

\* 测试用例，非法item

]

> 重构scheme枚举

[

\* 重新运行测试

]

  2. 计时，开始编码

  3. 要求用TDD的方式实现：先写测试，后写代码

如果超过90分钟还没完成，就先暂停。我们还需要留出一点时间来反思。

## 反思

  - 做完整道题用了多长时间？

  - 代码质量怎么样？

  - TDD的方式顺畅吗？

  - 实际做的步骤，和一开始拆分的任务是否相同？

>1.1 TEST参数列表（list）拆分的items数量：

@Test  
public void isMatchNumWhenListExist() {  
 *assertEquals*(new Parser(testNormalArgs).splitArgs().size(), 5);  
}

# 创建Parser类以及splitArgs() 函数

\* RUN TEST

List<String> rawList = new ArrayList<>(Arrays.*asList*(args.trim().split("-")));  
return rawList.stream()  
 .collect(Collectors.*toList*());

TESTRESULT: FAILED

测试数目 - 实际数目 = 1

# 增加filter()去除空项item

List<String> rawList = new ArrayList<>(Arrays.*asList*(args.trim().split("-")));  
return rawList.stream()  
 .filter(elem -> StringUtils.*isNotBlank*(elem))  
 .collect(Collectors.*toList*());

\* RUN TEST

TESTRESULT: OK

>1.2 在ParserTest中重构new Parsar().splitArgs()方法

private List<String> splitArgs(String testArgs) {  
 return new Parser(testArgs).splitArgs();  
}

TEST参数列表项（item）字符序列是否准确

@Test  
public void isMatchItemWhenListExist() {  
 *assertTrue*(new Parser(testNormalArgs).splitArgs().contains("l true"));  
}

\* RUN TEST

TESTRESULT: FAILED

# 增加map()去除多余空格

\* RUN TEST

TESTRESULT: OK

>1.3 TEST 辅助函数isBlank()

@Test  
public void isMatchTrueWhenListIsNull() {  
 *assertTrue*(new Parser(null).isBlank());  
}

TESTRESULT: FAILED

# 增加isBlank()方法

\* RUN TEST

TESTRESULT: OK

TEST args为null或者blank异常

@Test  
public void isMatchBlankExceptionWhenListIsBlank() {  
 *assertThrows*(ParserArgsException.class, () -> splitArgs(blankArgs));  
 *assertThrows*(NullPointerException.class, () -> splitArgs(null));  
}

TESTRESULT: FAILED

# 在splitArgs()中增加isBlank()的判断

\* RUN TEST

TESTRESULT: OK

>1.4 重构isBlank()为checkArgs()

RUN TEST AND OK

TEST 不含”-”的args

@Test  
public void isMatchParserExceptionWhenListNotContainFlag() {  
 ParserArgsException ex = *assertThrows*(ParserArgsException.class, () -> new Parser("p 90").checkArgs());  
 *assertTrue*(ex.getMessage().contains("args doesn't contain -"));  
}

RUN TEST BUT FAIED

# 在checkArgs()中增加if判断

RUN TEST AND OK

//--------------

获取到map结构，如何映射为返回对象，大量时间思考

1.1、1.2 TEST合并为assertArrayEquals()

考虑由外及内：

先写parse()函数，由其推动调用函数的编写

# 【讨论区】

做这道题，需要弄清楚这两个不同的角色，用户，用户的用户

用户是谁，怎么用

用户是谁？用户怎么用？

后台工程师

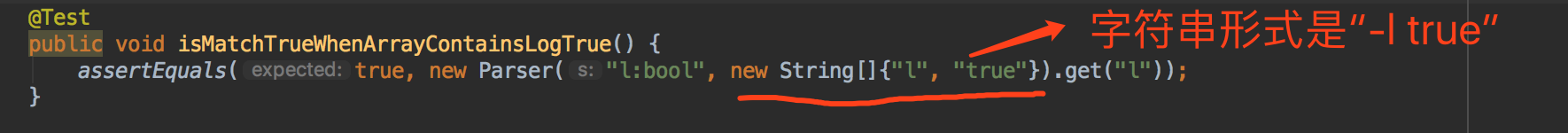
用户有多少个？

多个

每个用户在用的时候，他打算用一个参数名（例如“-x”）代表什么意思、什么类型的参数，他会跟其他用户交流吗？

-x在用户扩展里做，用户灵活可配

用代码表示一下，用户怎么用？



想个名字呢？

极限编程中所说的“隐喻”

找到合适的名字，理解各个东西是怎么彼此关联的

