



## 关于实验课

---

1. 使用腾讯课堂上课，如遇到技术故障将改用腾讯会议；
2. 为方便考勤，请同学们将昵称改成学号-真实姓名；
3. 上课不定时发起签到，请同学们不要迟到早退。



# 面向对象的软件构造导论

## 实验三：JUnit和单元测试

2022春

哈尔滨工业大学（深圳）



# 目录

---

01

实验目的

---

02

实验任务

---

03

实验步骤

---

04

作业提交

---



# 本学期实验总体安排

实验项目	一	二	三	四	五	六
学时数	2	2	2	4	2	4
实验内容	飞机大战 功能分析	单例模式 工厂模式	Junit 单元测试	策略模式 数据访问 对象模式	Swing 多线程	模板模式 观察者模式
分数	4	6	4	6	6	14
提交内容	UML类图、 代码	UML类图、 代码	单元测试 代码	UML类图、 代码	代码	项目代码、 实验报告

实验课程共**16**个学时，**6**个实验项目，总成绩为**40**分。



# 实验目的

---

- 了解单元测试的定义及其重要性，掌握JUnit5的常见用法；
- 理解编码规范的重要性，熟悉阿里编码规约插件的使用方法。



# 实验任务

---

1. 用JUnit5 创建单元测试类，编写测试用例对实验二的代码进行单元测试；
2. 使用阿里编码规约插件扫描代码，并改正问题代码。



# 实验步骤

## 1 了解单元测试&JUnit

**单元测试** (Unit Testing)，是指对软件中的最小可测试单元进行检查和验证。最小单元可以是一个类或是一个方法，包括基类（超类）、抽象类、或者派生类（子类）中的方法。

**JUnit** 是一个Java编程语言的单元测试框架，是xUnit家族中最成功的一个。它促进了测试驱动开发的发展，强调建立测试数据的一段代码，可以先测试，然后再应用。





# 实验步骤

## 2 用JUnit5进行单元测试

假如，有个实现简易计算器功能的类 Calculator，有加、减、乘、除功能。我们用JUnit5对它进行单元测试。



测它→

```
public class Calculator {  
    public int add(int x, int y) { //加法  
        return x + y;  
    }  
  
    public int sub(int x, int y) { //减法  
        return x - y;  
    }  
  
    public int mul(int x, int y) { //乘法  
        return x * y;  
    }  
  
    public int div(int x, int y) { //除法  
        return x / y;  
    }  
  
    public int div2(int x, int y) { //除法 做了异常判断  
        try {  
            int z = x / y;  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
        return x / y;  
    }  
  
    public void unCompleted(int x, int y) { //未完成的模块: 例如  
        //TODO  
    }  
}
```





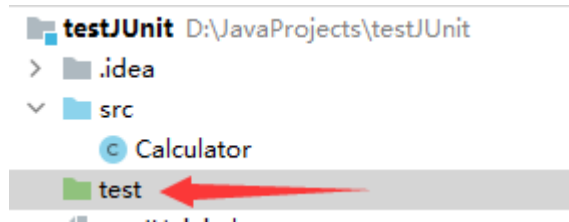


# 实验步骤

## 2 用JUnit5进行单元测试

### 1、创建测试类文件夹

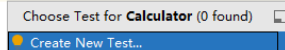
新建test文件夹，右键选择 Mark Directory as → Test Sources Root，标记test文件夹为测试类文件夹。



### 2、创建JUnit单元测试类

在待测试的类中，按下快捷键ctrl + shift + T，选择Create New Test。

```
public class Calculator {  
    public int add(int x, int y) { //加法  
        return x + y;  
    }  
  
    public int sub(int x, int y) { //减法  
        return x - y;  
    }  
  
    public int mul(int x, int y) { //乘法  
        return x * y;  
    }  
  
    public int div(int x, int y) { //除法  
        return x / y;  
    }  
  
    public int div2(int x, int y) { //除法 做了异常判断  
        try {  
            int z = x / y;  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```





# 实验步骤

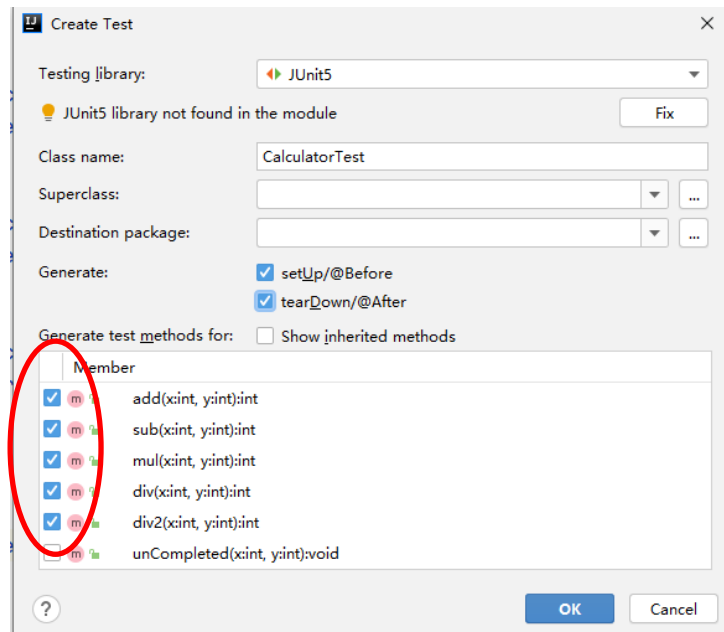
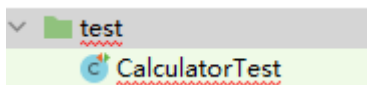
## 2 用JUnit5进行单元测试

### 3、勾选需要测试的方法

在弹出框的Member中勾选  
Calculator类中需要进行单元测试的方法

### 4、确认生成了单元测试类

查看test目录下自动生成的单元测试类CalculatorTest





# 实验步骤

## 2 用JUnit5进行单元测试

### 5、编写单元测试代码

详见实验指导书4.2节

注解

断言

测试异常

假设

参数测试

The screenshot shows an IDE with a project named 'testJUnit' and a test class 'CalculatorTest'. The project structure on the left includes 'src' and 'test' directories. The 'test' directory contains 'CalculatorTest' and 'testUnitIm1'. The 'CalculatorTest' class is highlighted. The code editor on the right shows the following code:

```
1 import org.junit.jupiter.api.*;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class CalculatorTest {
6     private Calculator calculator;
7
8     @BeforeEach
9     void setUp() {
10         calculator = new Calculator();
11     }
12
13     @AfterEach
14     void tearDown() {
15         calculator = null;
16     }
17
18     @Test
19     void add() {
20         System.out.println("*** Test add method executed ***");
21         assertEquals( expected: 10, calculator.add( x: 8, y: 2));
22     }
23
24     @Test
25     void sub() {
26         System.out.println("*** Test sub method executed ***");
27         assertEquals( expected: 6, calculator.sub( x: 8, y: 2));
28     }
29
30     @Test
31     void mul() {
32         System.out.println("*** Test mul method executed ***");
33         assertEquals( expected: 16, calculator.mul( x: 8, y: 2));
34     }
35 }
```





# 实验步骤

## 2 用JUnit5进行单元测试

### 6、运行测试类

右键CalculatorTest类，选择Run CalculatorTest，即可运行该单元测试类。

```
35 }
36
37 @Test
38 void div() {
39     System.out.println("*** Test div method executed ***");
40     assertEquals( expected: 4, calculator.div( 8, 2));
41 }
42
43
44 @Test
45 void div2() { //除法 做了异常判断
46     System.out.println("*** Test div2 method executed ***");
47     Exception exception = assertThrows(ArithmeticException.class, () -> calculator.div
48     assertEquals( expected: "/ by zero", exception.getMessage());
49     assertTrue(exception.getMessage().contains("zero"));
50 }
51 }
```

```
Run: CalculatorTest
Test Results
CalculatorTest
  add() 33 ms
  div() 19 ms
  mul() 1 ms
  sub() 1 ms
  div2() 11 ms
Tests passed: 5 of 5 tests - 33 ms
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" ...
*** Test add method executed ***
*** Test div method executed ***
*** Test mul method executed ***
*** Test sub method executed ***
*** Test div2 method executed ***
java.lang.ArithmeticException: Create breakpoint: / by zero
    at Calculator.div2(Calculator.java:20)
    at CalculatorTest.lambda$div2$0(CalculatorTest.java:47) <3 internal calls>
    at CalculatorTest.div2(CalculatorTest.java:47) <19 internal calls>
```





## 实验步骤

### 3 用JUnit5对飞机大战代码进行单元测试

结合飞机大战实例，在系统中选择英雄机、敌机、子弹和道具类的方法（**包含其父类方法**）作为单元测试的对象，用JUnit5进行单元测试。

**要求至少选择3个类，每个类至少测试2个方法**





# 实验步骤

---

## 4 了解编码规范&阿里编码规约插件

**编码规范**是程序编码所要遵循的规则，要注意代码的正确性、稳定性、可读性。好的编码规范是提高我们代码质量的最有效的工具之一。

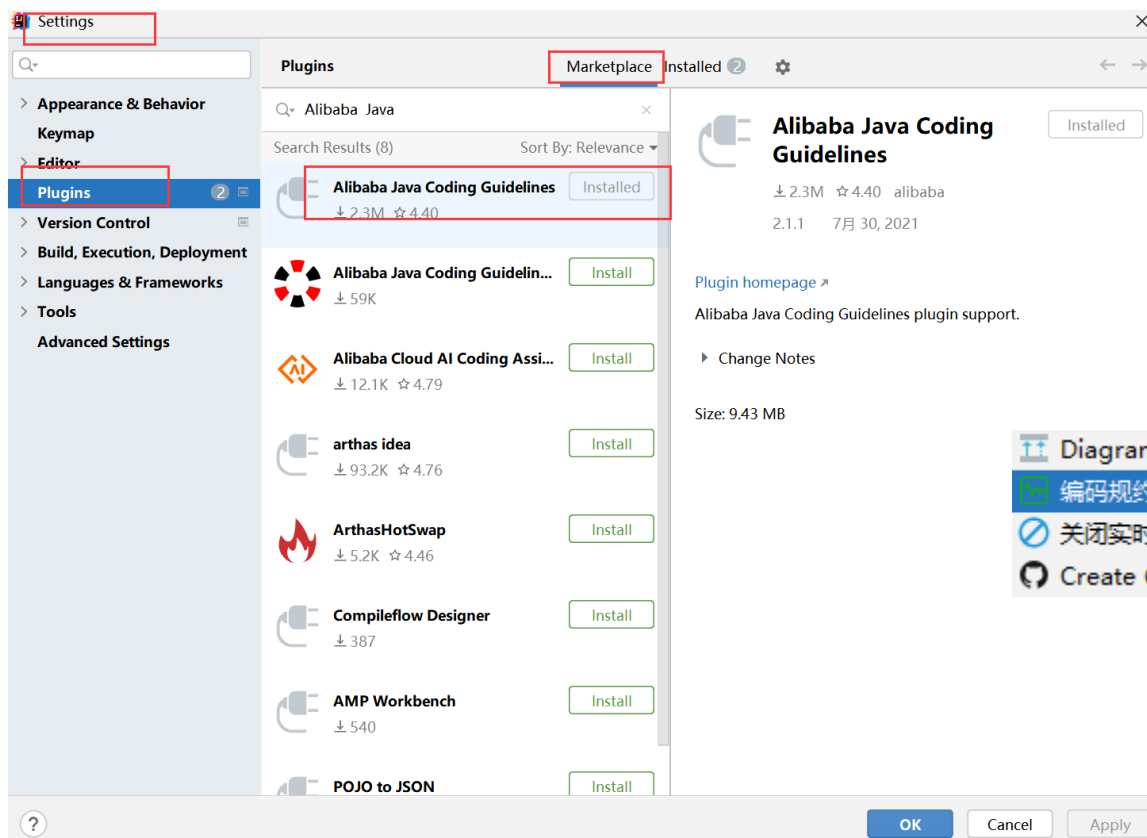
**阿里编码规约插件** (Alibaba Java Coding Guidelines) 是对《阿里巴巴 Java 开发规约》的一个延伸，它以一个 IDE 的插件存在，可以自动对手册中的 Java 不规范的问题进行提示。



# 实验步骤

## 5 安装插件

File --> Settings --> Plugins --> Marketplace, 搜索Alibaba  
安装完后重启IntelliJ IDEA。



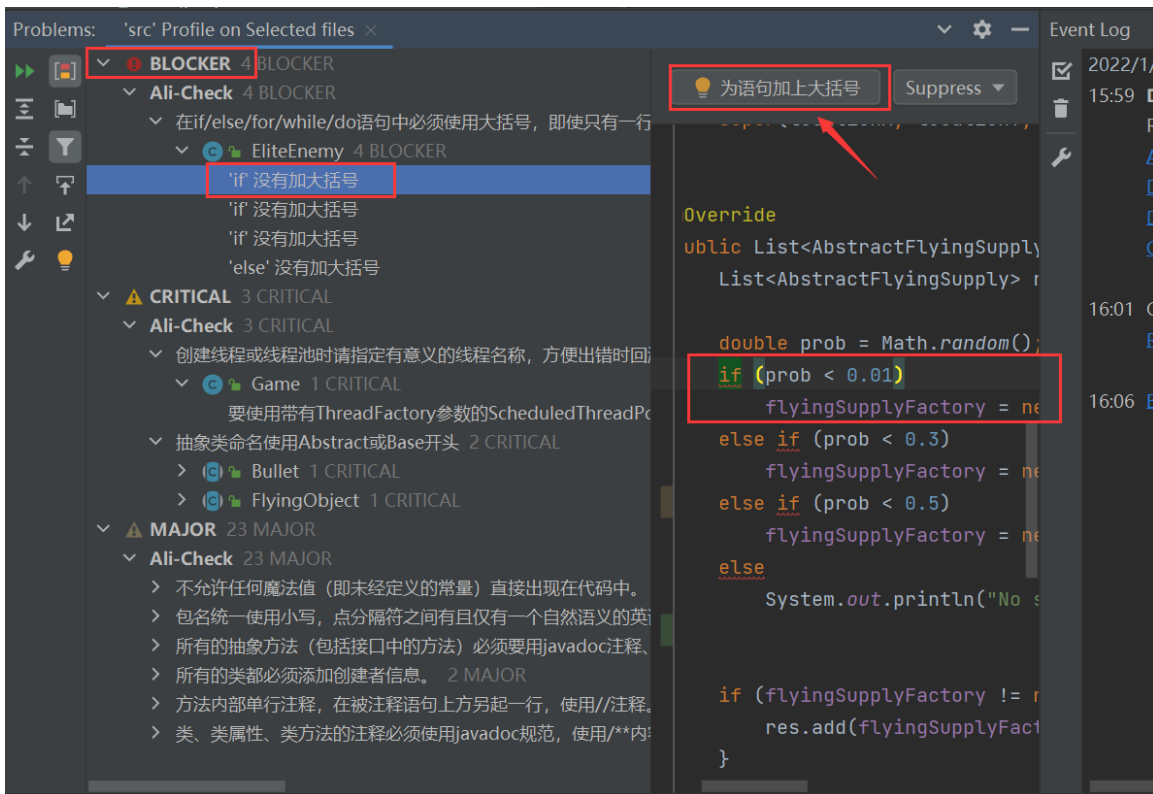


# 实验步骤

## 6 使用阿里编码规约插件扫描代码，修改代码

扫描代码后，按 **Blocker**（崩溃） / **Critical**（严重） / **Major**（重要）三个等级显示问题，双击可以定位至代码处，右侧窗口还有针对代码的批量修复功能。

- 建议处理掉全部 **Blocker/Critical** 级别的问题
- 修改前请备份代码







# 作业提交

---

## • 提交内容

- ① 提交项目代码（包含单元测试代码，压缩成zip包）；
- ② 提交测试报告，内容包括所设计的单元测试用例描述及相应的单元测试结果截图；
- ③ 在测试报告中列举2个用阿里编码规约插件扫描出来的问题，并描述如何解决（截图和文字描述）。

## • 截止时间

实验课后一周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。

登录网址：：<http://grader.tery.top:8000/#/login>



# 作业提交

## 一、单元测试

结合飞机大战实例，在系统中选择英雄机、敌机、子弹和道具类的方法（包含其父类方法）作为单元测试的对象，为每个测试对象编写单元测试代码。要求至少选择3个类，每个类至少2个方法（一个方法一个测试用例），并截图JUnit单元测试的结果。

### 1. 测试用例

用例编号				
待测试类及方法				
测试类及方法				
前提条件（如有）				
用例描述	测试步骤	期望结果	实际输出	测试结果

用例编号：唯一标识测试用例的序号，一般是数字或模块名首字母大写+数字序号。

待测试类及方法：该用例所测试的类名和方法名

测试类及方法：相应的测试代码的类名和方法名

前提条件（如有）：执行该测试用例的前提条件，比如碰撞检测，需已创建英雄机和敌机（或道具）。

用例描述：用一句话简单总结该测试用例的用意和目的。

测试步骤：详细完整的操作过程描述。

期望结果：正常情况下的响应结果。

实际结果：程序通过测试步骤后实际的响应结果。

测试结果：通过或失败

### 2. JUnit 单元测试结果

请把JUnit每个测试类（包含多个方法）的运行结果截图。

## 二、编码规约

列举2个实验中使用到的阿里编码规约插件的例子，截图或文字描述插件扫描出的代码问题以及你是如何解决的。



**同学们  
请开始实验吧！**