Java code coverage 代码覆盖率

https://www.jacoco.org/jacoco/

实验说明

在本实验中, 总共包括 4 个类文件:

- `TestRun.java` Junit5 测试入口文件, 包含两个被 `@Test` 注解的方法
- `TestRun2.java` 测试入口文件, 包含两个方法, 一个被 `@Test` 注解, 另外一个没有.
- `UtilPerson.java` 工具人
- UtilMan.java 工具男人

其关系如下:

- TestRun 中调用了 UtilMan 中的两个方法
- UtilPerson 中的两个类均没有被调用过

TestRun2.java

```
package utils;

import org.junit.Test;

public class TestRun2 {
    @Test
    public  void  testGetName(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

    public  void  testGetName2(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

}
```

TestRun.java

```java
package utils;

import org.junit.Test;

public class TestRun {
    @Test
    public  void  testGetName(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

    @Test
    public  void  testGetSex(){
        System.out.println("testGetSex");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getSex());
    }

}
```

UtilPerson.java

```java
package utils;

public class UtilPerson {
    public  String getName(){
        return "xxx";
    }
    public  String getSex(){
        int i=1;
        int b=3;
        b++;
        String s="absdfsd";
        return "xxx";
    }

    public  String getHeight(){
        return "xxx";
    }
}
```

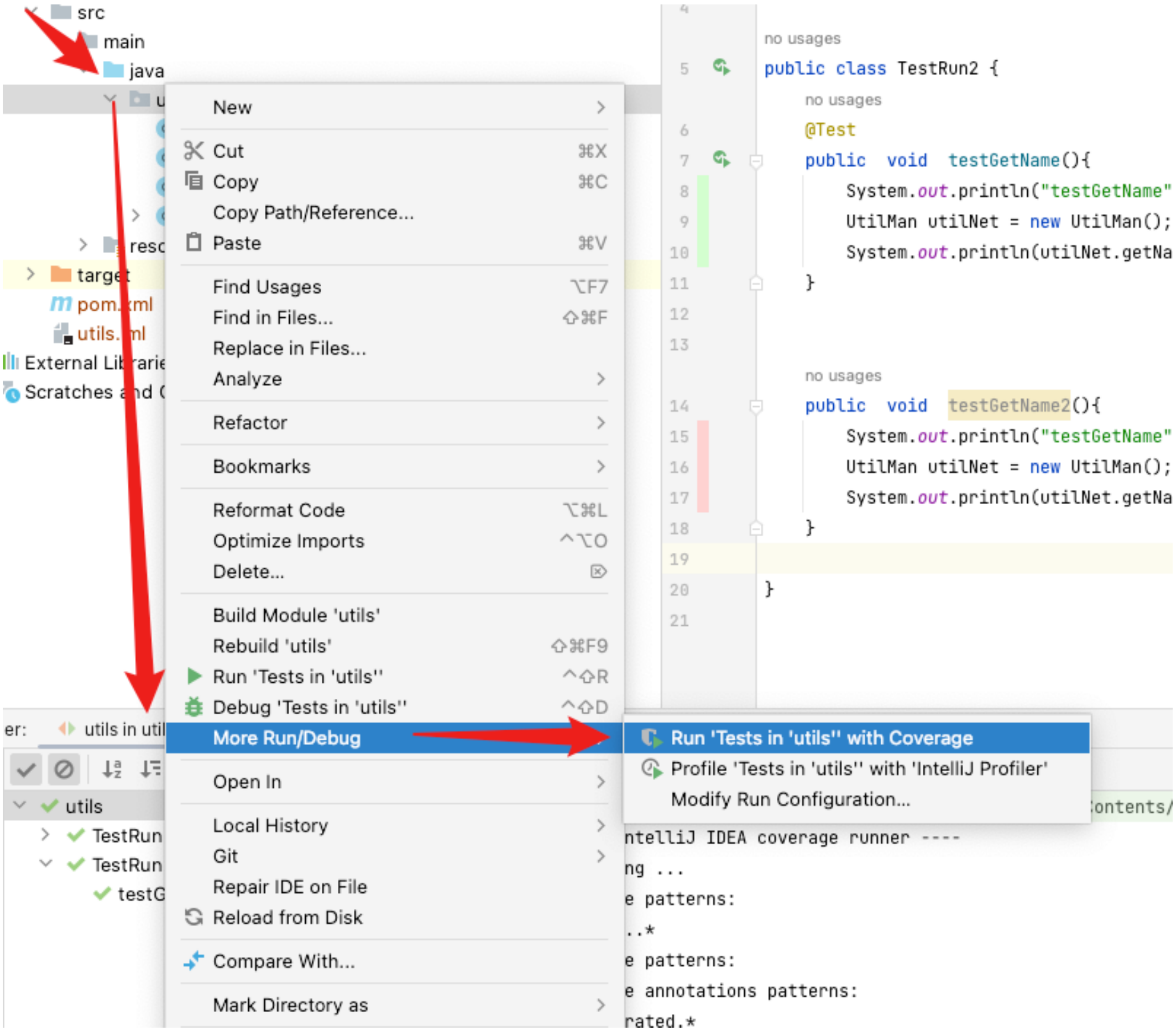
```
class Hand5Method{
    public  String getName(){
        return "xxx";
    }
    public  String getName2(){
        return "xxx";
    }
    public  String getName3(){
        return "xxx";
    }
    public  String getName4(){
        return "xxx";
    }
    public  String getName5(){
        return "xxx";
    }
}
```

UtilMan.java

```
package utils;

public class UtilMan {
    public  String getName(){
        int a=0;
        a++;
        return "xxx";
    }

    public  String getSex(){
        return "xxx";
    }

    public  String getHeight(){
        return "xxx";
    }
}
```

使用 coverage 运行

Idea 默认已经集成了 coverage, 运行方式如下:



运行效果

通过 `code coverage` 运行后, 结论:

Element	Class, %	Method, %	Line, %
utils	50% (2/4)	30% (4/13)	59% (19/32)
Hand5Method	0% (0/1)	0% (0/5)	0% (0/5)
TestNet	100% (1/1)	100% (2/2)	100% (7/7)
UtilMan3Method	100% (1/1)	66% (2/3)	92% (12/13)
UtilPerson	0% (0/1)	0% (0/3)	0% (0/7)

其中:

1. Class 表示类, `2/4` 表示 4 个类中有 2 个类被测试覆盖了. 只要类中的一个方法被测试了, 那么这个类就就视为被测试过了.

2. Method 表示类的方法, `2/4` 表示 4 个方法中有两个方法被测试到了. **被测试的定义是**: 该方法被 JUnit 执行到了, 也就是有 `@Test` 的方法, 如下面的代码的代码覆盖率是 `1/2`, 因为只有 `testGetName()` 被 `@Test` 标注了, 也就是只有 `testGetName` 被执行 (被 `JUnit` 执行), 因此是代码覆盖率 `1/2`.

```
package utils;

import org.junit.Test;

public class TestRun2 {
    @Test
    public  void  testGetName(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

    public  void  testGetName2(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

}
```

而下面的代码的覆盖率是 `2/2`:

```
package utils;

import org.junit.Test;

public class TestRun {
    @Test
    public  void  testGetName(){
        System.out.println("testGetName");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getName());
    }

    @Test
    public  void  testGetSex(){
```

```
        System.out.println("testGetSex");
        UtilMan utilNet = new UtilMan();
        System.out.println(utilNet.getSex());
    }

}
```

因为两个方法都被执行了, 都被 `@Test` 标注了.

3. Line 表示有多少行代码被执行 (测试) 到, 有效语句值实际进行计算的地方 (声明,注释,导包不算在行内)

运行结果

运行结果是针对每个测试类来说的,运行不同的类, 得到的结果是不同的, 如下:

运行 `TestRun` 的结果:

Element	Class, %	Method, %	Line, %
utils	40% (2/5)	26% (4/15)	38% (12/31)
Hand5Method	0% (0/1)	0% (0/5)	0% (0/5)
TestRun	100% (1/1)	100% (2/2)	100% (7/7)
TestRun2	0% (0/1)	0% (0/2)	0% (0/6)
UtilMan	100% (1/1)	66% (2/3)	83% (5/6)
UtilPerson	0% (0/1)	0% (0/3)	0% (0/7)

运行 `TestRun2` 的结果

Element	Class, %	Method, %	Line, %
utils	40% (2/5)	13% (2/15)	25% (8/31)
Hand5Method	0% (0/1)	0% (0/5)	0% (0/5)
TestRun	0% (0/1)	0% (0/2)	0% (0/6)
TestRun2	100% (1/1)	50% (1/2)	57% (4/7)
UtilMan	100% (1/1)	33% (1/3)	66% (4/6)
UtilPerson	0% (0/1)	0% (0/3)	0% (0/7)

在目录 `utils` 上运行的结果

Element	Class, %	Method, %	Line, %
utils	60% (3/5)	33% (5/15)	50% (16/32)
Hand5Method	0% (0/1)	0% (0/5)	0% (0/5)
TestRun	100% (1/1)	100% (2/2)	100% (7/7)
TestRun2	100% (1/1)	50% (1/2)	57% (4/7)
UtilMan	100% (1/1)	66% (2/3)	83% (5/6)
UtilPerson	0% (0/1)	0% (0/3)	0% (0/7)

当运行整个文件夹时, **会统计每个入口类 (JUnit) 的平均结果**.