摘要

实验5-6的实验目的是使用工具对一个小项目生成测试。分析这些测试，以了解测试了什么以及遗漏了什么。

**关键词：**软件测试；Evosuite；ArrayList

目录

[1实验内容 5](#_Toc42543270)

[1.1实验工具 5](#_Toc42543271)

[1.2 测试项目 5](#_Toc42543272)

[2实验工具安装 5](#_Toc42543273)

[简介 5](#_Toc42543274)

[安装 5](#_Toc42543275)

[3 源码和测试分析 8](#_Toc42543276)

[3.1 源码 8](#_Toc42543277)

[3.2 测试覆盖率及分析 9](#_Toc42543278)

[3.3源码和Junit测试 12](#_Toc42543279)

[参考文献 13](#_Toc42543280)

# 1实验内容

## 1.1实验工具

我选择的实验工具是Evosuite

## 1.2 测试项目

我使用的测试项目是java.util.ArrayList，和理论课程选择的源码是一样的。在使用Evosuite对该项目生成测试后，对测试覆盖内容进行分析。

# 2实验工具安装

## 简介

EvoSuite是由Sheffield等大学联合开发的一种开源工具，用于自动生成测试用例集，生成的测试用例均符合Junit的标准，可直接在Junit中运行。

## 安装

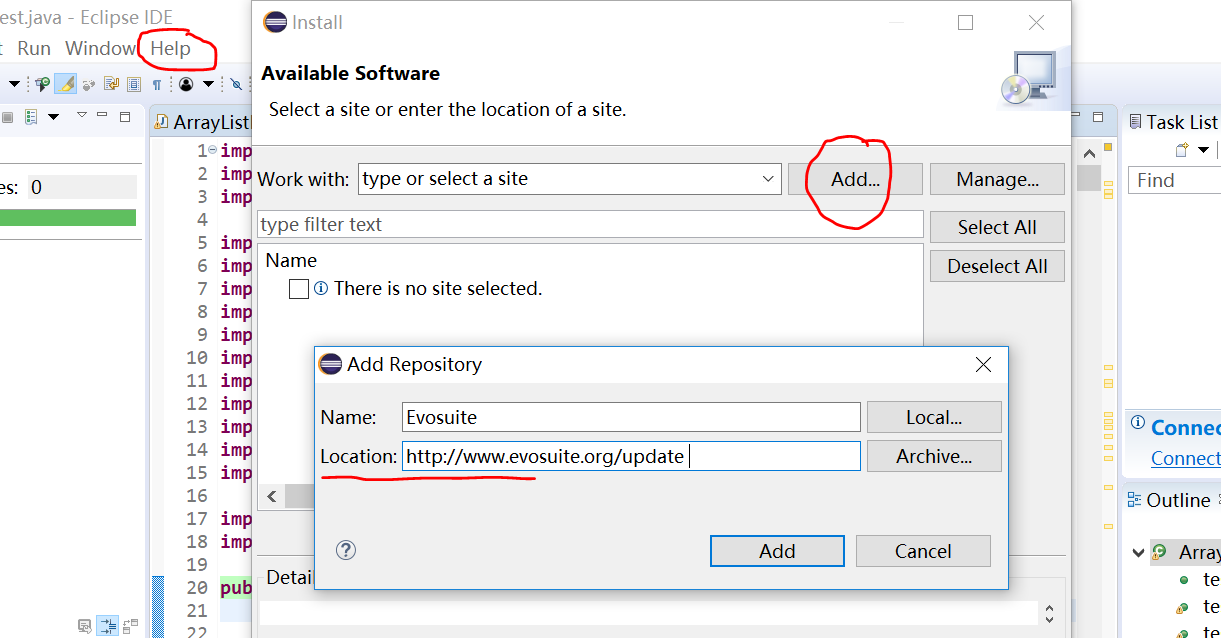
EvoSuite 插件需要Java 8 的运行环境，我在Java10的环境下安装完插件，无法成功生成测试，所以不是Java10的运行环境需要卸载重装jdk1.8。

安装过程如下：

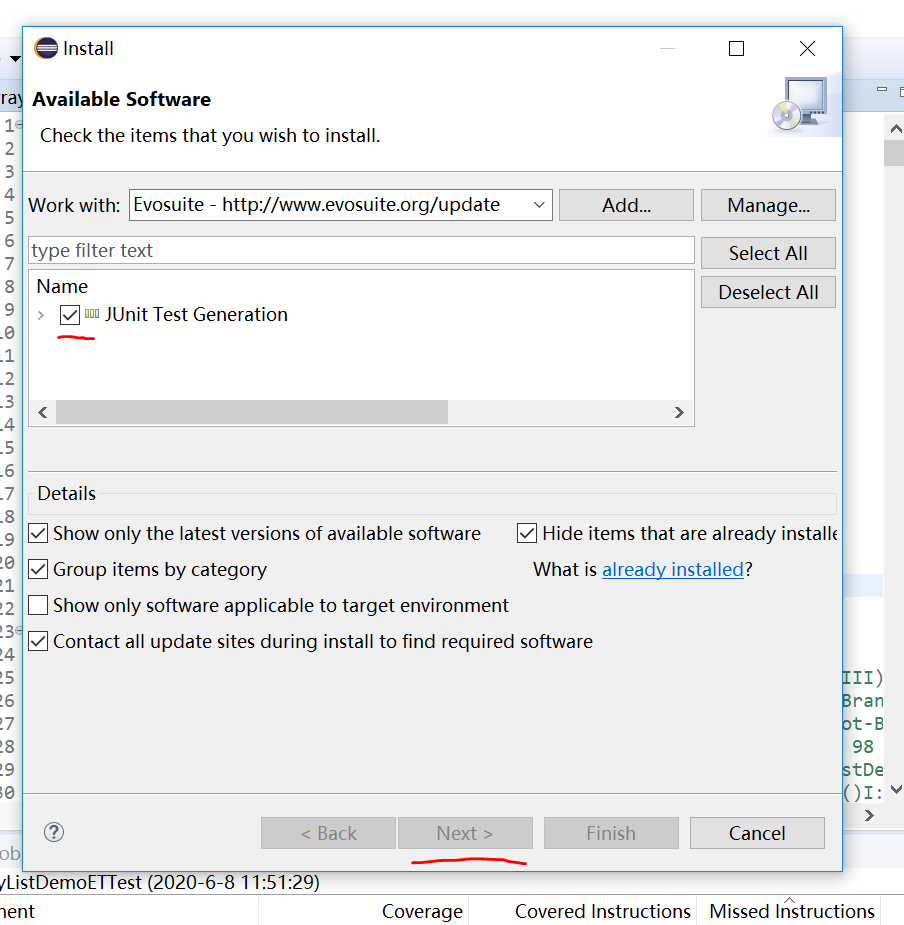
打开Eclipse 依次点击菜单栏的help->Install New Software…

点击Add…按钮

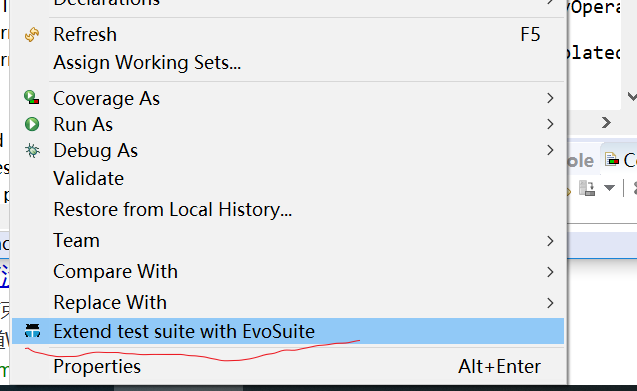
在Location栏中输入EvoSuite Eclipse插件的地址：http://www.evosuite.org/update 后，点击Add按钮



选中 Junit Test Generation 后，点击Next按钮



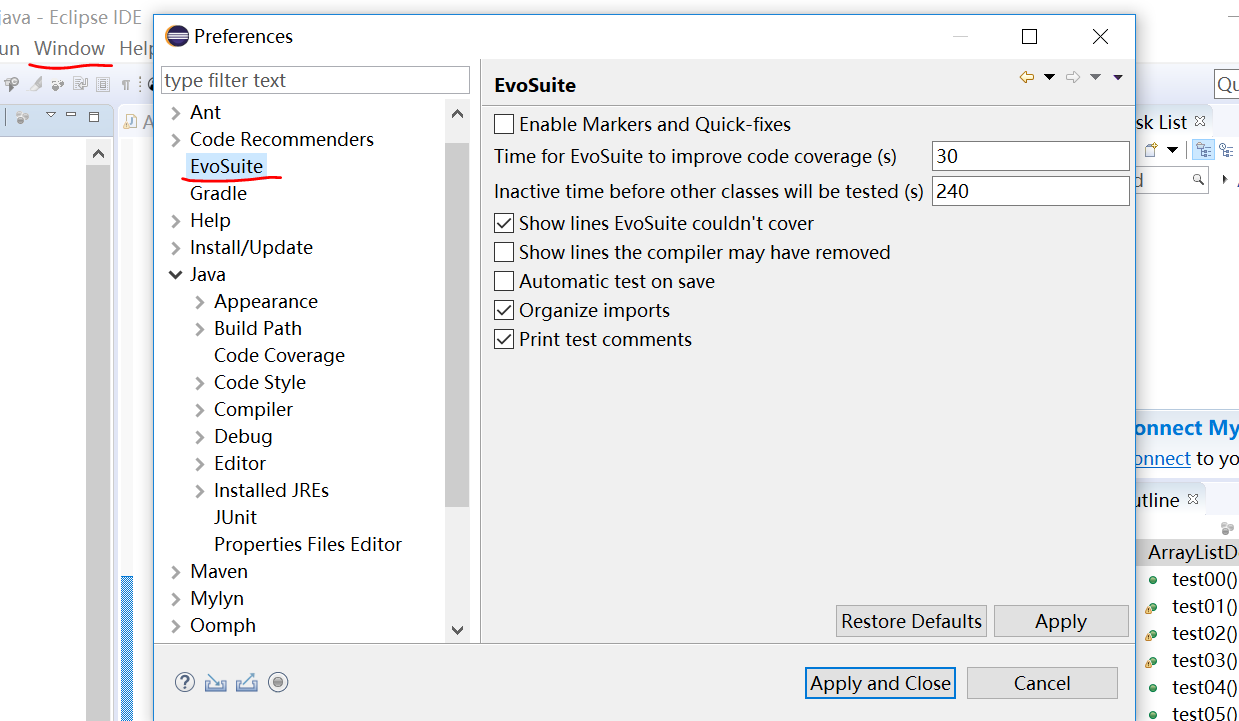
运行时间可能过长，等待运行完毕, 点击Next按钮，选中 I accept the terms of license agreement后，点击Finish按钮后，显示 Installing Software对话框，等待一段时间后，会弹出SecurityWarning对话框，点击Ok，一路同意下去，重启eclipse后，右键一个java文件后，出现如下选项：



则安装成功。

选中需要测试的类，右击鼠标，选择Generate tests with EvoSuite ，则会生成测试用例，测试用例生成在项目中的evosuite-tests文件夹内。生成的测试类为标准的Junit 4 测试类，可以完全按照Junit 4 的操作规范对其测试。

点击Windows->Preferences ，查找到EvoSuite即可对其参数进行设置，设置界面如下：



由于某些原因，生成的测试无法直接运行，我将测试代码粘贴至另一个junit测试文件中运行。

# 3 源码和测试分析

## 3.1 源码

由于要测试覆盖率，我将ArrayList<E>类的代码复制为本地文件，ArrayListDemo.java，由于remove(), add()等多个方法涉及访问parent.modCount，而AbstractList<E>类的成员变量**protected** **transient** **int** modCount = 0;在包外无法访问，我新建了AbstractListDemo.java文件, 将AbstractList<E>类的代码复制到此文件中。在ArrayListDemo.java中引入了需要使用一些方法所属的类如下。

**import** java.util.\*;

**import** java.util.function.Consumer;

**import** java.util.function.Predicate;

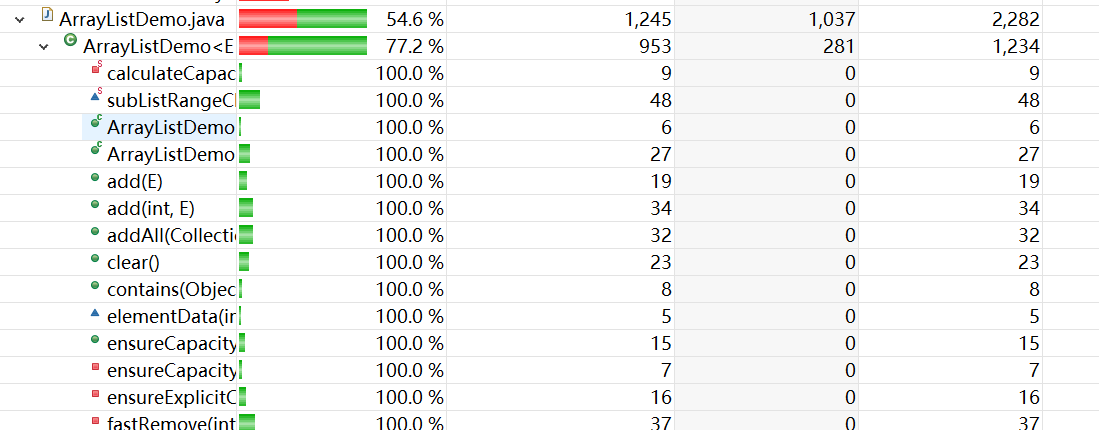
**import** java.util.function.UnaryOperator;

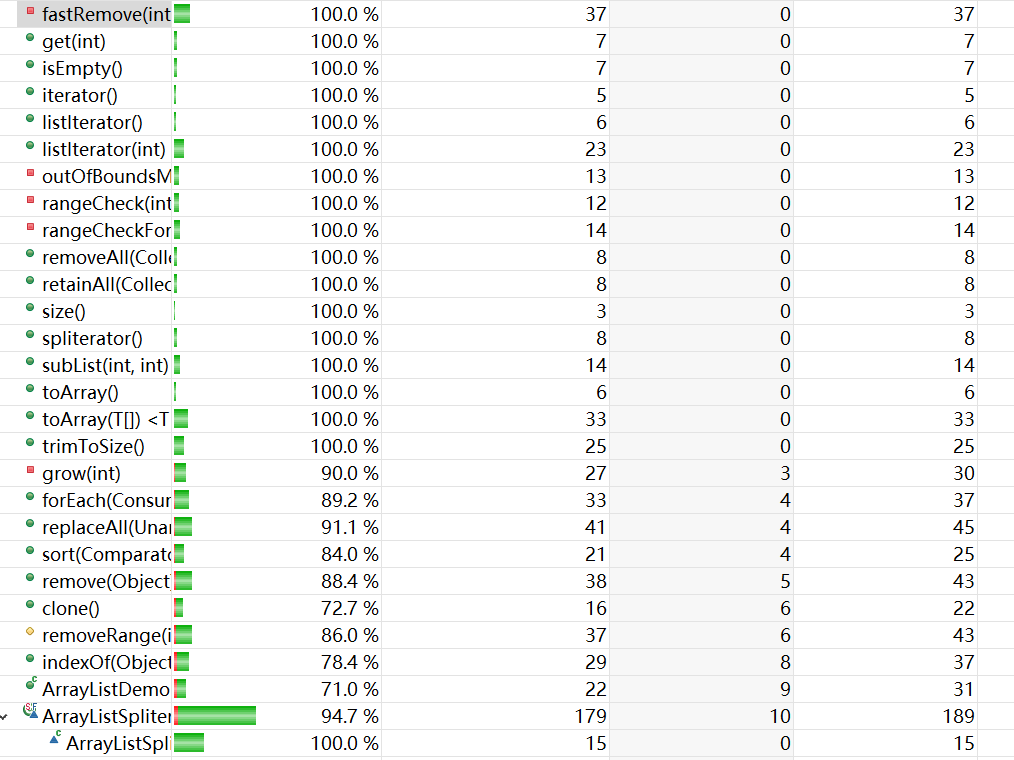
**import** sun.misc.SharedSecrets;

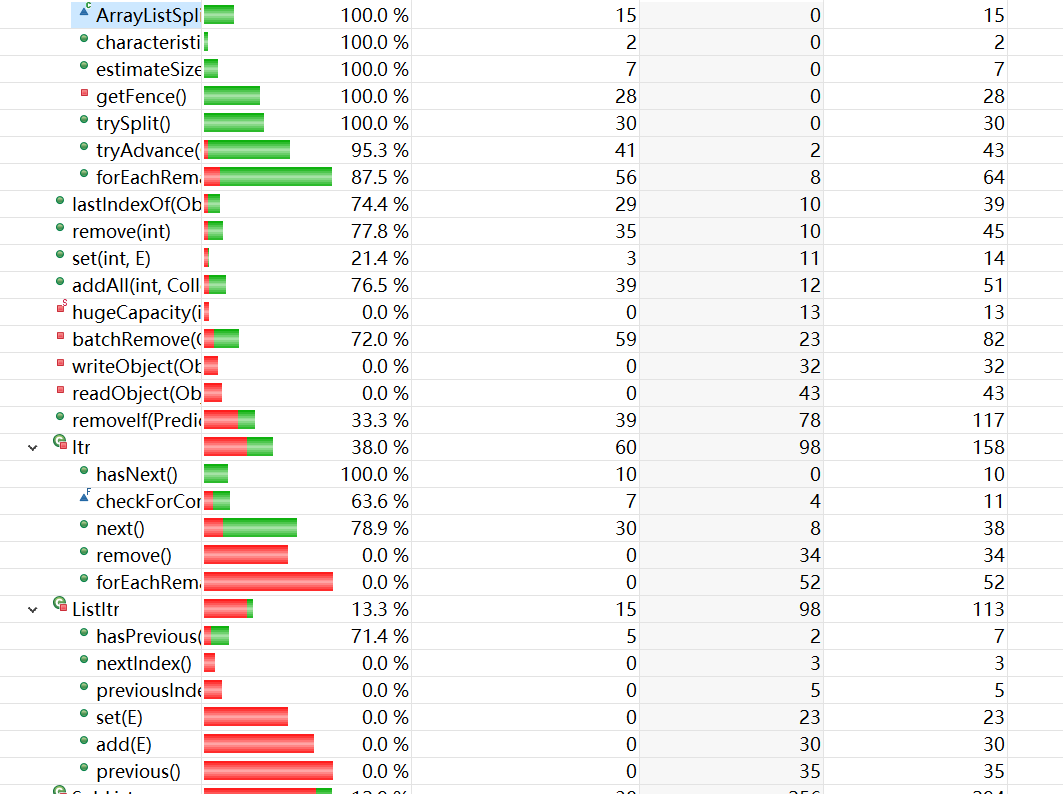
## 3.2 测试覆盖率及分析

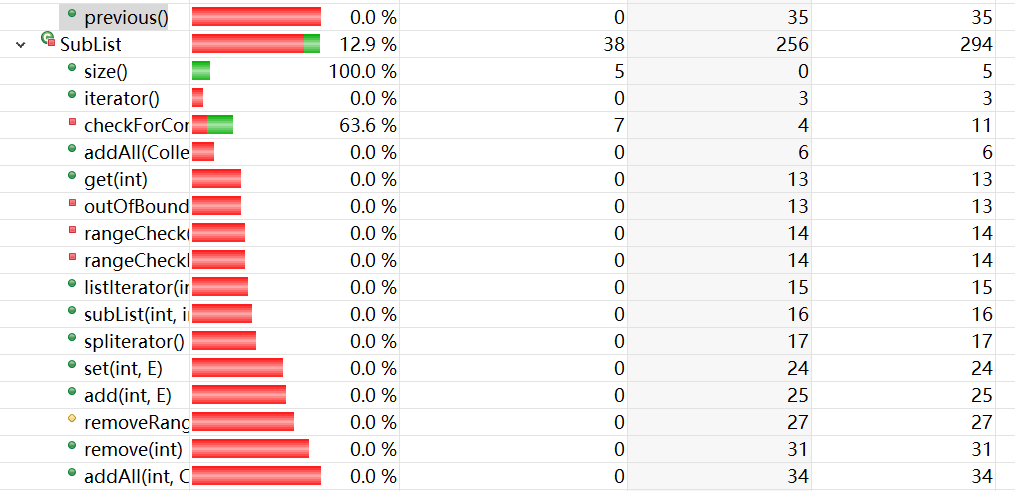
总概率：54.6%

ArrayList<E>类的方法覆盖率较高，类中的私有类测试覆盖率较低



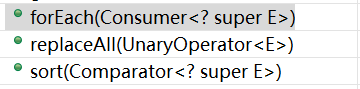






鉴于它测试到的方法太多，我这里分析一些它未测试到的方法

ArrayList的grow方法的第1个分支，条件newCapacity - minCapacity < 0为false未测到，第2个分支条件newCapacity - MAX\_ARRAY\_SIZE > 0为true未测到，这个方法的指令覆盖率：90%



上图这3个方法都是

**if** (modCount != expectedModCount) {

**throw** **new** ConcurrentModificationException();

}

这段检查列表结构性变化的分支没有被测试到，前两个的循环的条件表达式没有被完全执行；

clone()方法捕捉CloneNotSupportedException代码块没有被测试，因为java实现了支持克隆，所以这个代码块是无法被执行到的；

removeRange(**int** fromIndex, **int** toIndex)方法的将被移除的元素置为null的循环条件为true的分支没有被测试到，生成的测试测试了两次条件为false的分支；

indexOf(Object o)方法的if分支里的循环条件为true的分支没有被测试，生成了一个列表不包含任何元素，导致循环的条件为false，需要add一些元素；

ArrayListDemo(Collection<? extends E> c)构造方法的分支

**if** (elementData.getClass() != Object[].**class**)

elementData = Arrays.*copyOf*(elementData, size, Object[].**class**);

没有被测试到，让列表持有的数组不返回为Object[]很难做到；

lastIndexOf(Object o)方法的第1个循环的条件为true的分支没有被测到，因为列表中不包含元素，第2个循环的if分支没有被测到，因为列表没有包含要找寻下标的元素；

remove(int index)方法的if分支没有被测到，因为列表只有1个元素；

set(int index, E element)方法rangeCheck(index)后面的代码都没有被测到，因为列表不包含元素，或传的参数下标为负数或超过列表的size；

其他增删改的方法未被测试到的部分，原因和上述类似

writeObject，readObject完全没有被测试到，因为它们的修饰符是private;

类中的类ArrayListSpliterator<E>覆盖率较高，为94.7%，下面介绍未被测试到的部分：

tryAdvance方法的**if** (i < hi)分支的条件list.modCount != expectedModCount为false的分支没有被测到，这是因为用**new** ArrayListDemo.ArrayListSpliterator<Object>(arrayListDemo0, 0, 1, 1);使结构性改变次数与记录的不一致，只测了条件为true的分支；

forEachRemaining(Consumer<? super E> action)没有测试循环条件为true的分支, 因为所有对ArrayListDemo<Object> list, int origin, int fence, int expectedModCount)传的参数的origin>=fence，是条件为false；

类中的类Itr覆盖率为38.0%，下面介绍被被测试到的部分：

hasNext()方法

checkForComodification()方法的if条件为false的分支

next()方法的两个if分支会抛出异常的没有被测到，其他的实现功能的代码都被测到了

类中的类ListItr只有hasPrevious()方法被测到了，且只测到cursor == 0，也就是没有前一项的分支；

类中的类SubList的测试覆盖率较低，为12.9%，但是它的所有方法ArrayList也都有，因此可以用ArrayList的测试用例稍作修改后用于测试，这样就大大提高了整个源码的覆盖率

总的来说，生成的测试覆盖率比较可观，在源码结构较简单时可以达到更高的覆盖率，可以作为良好的辅助测试的软件，生成的测试结构清晰，可以有效减轻测试的工作，在工作量适中的修改后就可以达到较高的测试覆盖率。

## 3.3源码和Junit测试

源码：<ArrayListDemo.java>

Junit测试：<ArrayListDemoETTest.java>

# 参考文献

[1]Hugh修正的博客 . 自动化单元测试工具 EvoSuite 的简单使用[EB/OL］.博客园，<https://www.cnblogs.com/hughding/p/evosuite.html．2016-08-31>.