Software Testing HW#4

Student Name:
Pouya Haji Mohammadi Gohari

SID:810102113

Date of deadline
Thursday 11th July, 2024

Dept. of Computer Engineering

University of Tehran

Contents

1 Question 1

We will use Randoop for generating test for two different classes in distinguish directories ’impl1’ and ’impl2’. Through investigating randoop manual, and download latest version of randoop from it’s github we can generate tests.
Let us explain it step by step:

1. Create an maven project with JDK version 11.
2. Download codes from this link and copy the main directory into source of created project.
3. Create a ’libs’ directory and copy the downloaded randoop(randoop-all-4.3.3.jar or any jar version you have downloaded) to ’libs’.
4. Before jumping to generate tests, we should compile codes in order to create target/classes directory.
5. In ’intelij’ IDE, open the terminal and use following code to generate tests:

   ```
   java -classpath target/classes:libs/randoop-all-4.3.3.jar:~/.m2/repository/org/junit/platform/junit-platform-console-standalone/1.8.2/junit-platform-console-standalone-1.8.2.jar randoop.main.Main gentests --testclass=ir.ramtung.impl1.Library --junit-output-dir=src/test/java/tests_impl1 --output-limit=30 --regression-test-basename=RegressionTestImpl1
   ```

   We should specify the class paths which is target/classes and specify junit and randoop jar file itself.The class we want to generate tests for will be specified with testclass variable.
   The generated tests will be written to junit output directory and we set an limit that randoop do not create tests more than 30.

6. Randoop usually create two test classes where one is for error(if any error appears while testing), and other one is for regression which can be used as an oracle.

We have obtain multiple property from provided specification’s link.
Based on the following property achieved by the specification, we will change some tests obtained by randoop for each library implemented in impl1 and impl2 directory with maven.
Properties:

1. Each professor can borrow at most 5 documents.
2. For students, this number is 2.
3. Each document can has multiple copies therefore can be borrowed form multiple persons.
4. If all copies of same document were borrowed, no one can borrow this document.
5. Due time of returning a book is 10.
6. Due time of returning a reference is 5.

7. Due time of returning a magazine is 2.
8. Members of library can extend this duration at most 2 times if they borrowed either book or reference.
9. Magazines can not be extended.
10. Can not extend a document in the same day that was borrowed.
11. Can not extend a document that due time is passed.
12. For example, if a member borrows a regular book on 1/9/1399, the return deadline will be 11/9/1399. This means that if the book is returned on that day, no penalty will be applied. If the member extends the borrowing period on or before 11/9/1399, the new return deadline will be 21/9/1399.
13. Penalty of passing due time has been reported in table 1:

Reference Book	For the first 3 days overdue, the fine is 5000 tomans per day and for subsequent days, it is 7000 tomans per day (e.g., for 5 days overdue, the total fine will be 29000 tomans).
Regular Book	For the first 7 days overdue, the fine is 2000 tomans per day and for days 8 to 21, it is 3000 tomans per day and for subsequent days, it is 5000 tomans per day.
Magazine	For magazines published before 1390, the fine is 2000 tomans per day and for those published after 1390, it is 3000 tomans per day.

Table 1: Fine Schedule

14. Adding Student/Prof Member: If either member name of sid were empty, then exception must be thrown.Even if either student or professor with the same name will expected to throw some exception.
15. Adding Book/Magazine/Reference:Duplicate name for documents are not allowed.If name is empty, it consider to throw an exception.For magazine year and number must be positive number.
16. Time Pass: Days must be non-negative number.

Now lets examine test cases obtained by Randoop for each implementation. Note that We have generated 23 and 13 test cases for first and second implementation respectively.

1.1 Implementation 'Impl1'

It seems like when Randoop was generating tests with first elements in its repository, there were so many exception has been made which was obvious for first implementation since this code handles so many exceptions. Unfortunately, the catch was not filled by this tool. Thus we will complete them and if possible add to each test asserts to have more powerful Oracle.

1. Test Case 1: Tries to return a document which was not created at all.

```
    @Test
    public void test01() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test01");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
            ↪ ();
        // The following exception was thrown during execution in test
            ↪ generation
        try {
            library0.returnDocument("", "hi!");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
                ↪ impl1.InvalidArgumentEx; message: The document is not in
                ↪ member's loan");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
        }
        org.junit.Assert.assertNotNull(strList1);
    }
```

Completed the catch with true value:

```
        org.junit.Assert.assertEquals(e.getMessage(), "The document is not in
            ↪  member's loan");
```

In this case we can not even add a person with empty string since it would give us another exception!

2. Test Case 2: This test is trying to get a penalty for a member that does not exist:

```
    @Test
    public void test02() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test02");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
            ↪ ();
        // The following exception was thrown during execution in test
            ↪ generation
        try {
            int int3 = library0.getTotalPenalty("");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
                ↪ impl1.InvalidArgumentEx; message: Cannot find member");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
        }
        org.junit.Assert.assertNotNull(strList1);
    }
```

Add this to catch:

```
org.junit.Assert.assertEquals(e.getMessage(), "Cannot find member");
```

Note that in this test case we could not add an empty string member since it would give us another exception.

3. Test case 3: This test is trying to extend a document for a member that both does not exist therefore we must get one of this exception either member or document is not found(From now on we just say what we would add to catch for this kind of test cases). We must get document is not in member's loan therefore:

```
org.junit.Assert.assertEquals(e.getMessage(), "The document is not in
   ↪ member's loan");
```

4. Test case 4: This test is trying to add a book with empty string as name. Therefore it must throw an exception ”Documents with empty title are not allowed”:

```
org.junit.Assert.assertEquals(e.getMessage(), "Documents with empty title
   ↪  are not allowed");
```

5. Test case 5: Same as test case number 2.

6. Test case 6: Same as test case number 3.

7. Test case 7: This test case is assume an empty string name person want an empty named doc. Therefore it must throw ”can not find document to borrow”.

```
org.junit.Assert.assertEquals(e.getMessage(), "Documents with empty title
   ↪  are not allowed");
```

8. Test case 8: This case will check the instantiated class is not null and we can finally add something to this test case.The generated test is:

```
@Test
public void test08() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "RegressionTestImpl10.test08");
    ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
    library0.addProfMember("hi!");
    java.lang.Class<?> wildcardClass3 = library0.getClass();
    org.junit.Assert.assertNotNull(wildcardClass3);
}
```

Now we can add a reference book and see if penalty is true or not:

```
@Test
public void test08() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "RegressionTestImpl10.test08");
    ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
```

```
        library0.addProfMember("hi!");
        library0.addReference("asd", 100);
        library0.borrow("hi!", "asd");
        library0.timePass(5);
        int expected = library0.getTotalPenalty("hi!");
        org.junit.Assert.assertEquals(expected, 0);
        library0.timePass(1);
        expected = library0.getTotalPenalty("hi!");
        org.junit.Assert.assertEquals(expected, 2000);
        java.lang.Class<?> wildcardClass3 = library0.getClass();
        org.junit.Assert.assertNotNull(wildcardClass3);
    }
```

This test case was failed since as we know in the previous homework, there is a bug in calculating reference penalty.

9. Test Case 9: Same as test case number 4.
10. Test Case 10: Same as test case number 3.
11. Test Case 11: Same as test case number 4.
12. Test Case 12: This test case can be modified and see if other aspect of code will be covered. The generated test case is:

```
    @Test
    public void test12() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test12");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        library0.addProfMember("hi!");
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.borrow("hi!", "");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
              ↪ impl1.InvalidArgumentEx; message: Cannot find document to
              ↪ borrow");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
        }
    }
```

The modified test is designed to see if magazine will be extended or not (Specification said the Magazine documents must not be extended)

```
    @Test
    public void test12() throws Throwable {
        if (debug)
```

```
            System.out.format("%n%s%n", "RegressionTestImpl10.test12");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        library0.addProfMember("hi!");
        library0.addMagazine("see", 2000, 10, 100);
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.borrow("hi!", "see");
            library0.timePass(1);
            library0.extend("hi!", "see");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: Can not extend Magazine");
        } catch (ir.ramtung.sts01.LibraryException e) {
            org.junit.Assert.assertEquals(e.getMessage(), "Can not Extend
               ↪ Magazine");
            // Expected exception.
        }
    }
```

13. Test Case 13: Since we have already multiple tests for exceptions like:

    - The document is not in member's loan
    - Documents with empty title are not allowed
    - Documents with empty title are not allowed

    Let's modify this test in order to cover more of our code. The generated test was:

```
   @Test
    public void test13() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test13");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        library0.addProfMember("hi!");
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.returnDocument("hi!", "");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: The document is not in
               ↪ member's loan");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
        }
    }
```

    The modified code:

```java
    @Test
    public void test13() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test13");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        library0.addProfMember("hi!");
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.addStudentMember("810", "hi!");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: Member with the same name
               ↪ exists");
        } catch (ir.ramtung.sts01.LibraryException e) {
            org.junit.Assert.assertEquals(e.getMessage(), "Member with the
               ↪ same name exists");
            // Expected exception.
        }
    }
```

This test is for see if adding a person with same name will throw and exception or not.

14. Test case 14: Same as test case 3.

15. Test case 15: Same as test case 7.

16. Test case 16: Same as test case 7.

17. Test case 17: This test case has been generated in various ways.The generated test:

```java
    @Test
    public void test17() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test17");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        library0.addProfMember("hi!");
        library0.timePass(100);
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.addProfMember("");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: Empty member name not
               ↪ allowed");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.

        }
    }
```

We can modify the test to see negative values for passing time is throwing an exception or not:

```
    @Test
    public void test17() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test17");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.timePass(-100);
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: Cannot go back in time");
        } catch (ir.ramtung.sts01.LibraryException e) {
            org.junit.Assert.assertEquals(e.getMessage(), "Cannot go back in
               ↪ time");
            // Expected exception.

        }
    }
```

18. Test Case 18: This test is designed to just see if creating instance is not null.

```
    @Test
    public void test18() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test18");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        java.lang.Class<?> wildcardClass2 = strList1.getClass();
        org.junit.Assert.assertNotNull(strList1);
        org.junit.Assert.assertNotNull(wildcardClass2);
    }
```

We can add something like title to it too see if available titles has been implemented correctly or not.The modified test is:

```
    @Test
    public void test18() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test18");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        java.lang.Class<?> wildcardClass2 = strList1.getClass();
        org.junit.Assert.assertNotNull(strList1);
```

```java
        org.junit.Assert.assertNotNull(wildcardClass2);
        library0.addBook("title1", 100);
        library0.addBook("title2", 200);
        library0.addBook("title3", 300);
        java.util.List<String> titles = library0.availableTitles();
        org.junit.Assert.assertNotNull(titles);
        org.junit.Assert.assertTrue(titles.contains("title1"));
        org.junit.Assert.assertTrue(titles.contains("title2"));
        org.junit.Assert.assertTrue(titles.contains("title3"));
    }
```

19. Test Case 19: This scenario wants to add a person with an empty string for name.Therefore the ”Empty member name not allowed” must be thrown therefore we add the following line in catch block:

```java
    @Test
    public void test19() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl10.test19");
        ir.ramtung.impl1.Library library0 = new ir.ramtung.impl1.Library();
        // The following exception was thrown during execution in test
           ↪ generation
        try {
            library0.addProfMember("");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ impl1.InvalidArgumentEx; message: Empty member name not
               ↪ allowed");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
            org.junit.Assert.assertEquals(e.getMessage(), "Empty member name
               ↪ not allowed");
        }
    }
```

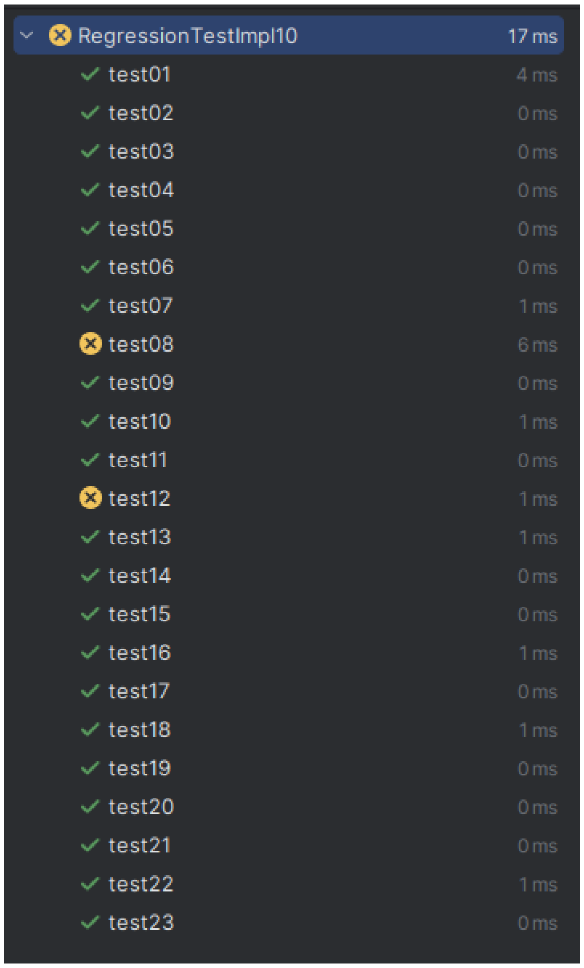
20. Test Case 20: This test case is good enough.

21. Test Case 21: This test case has same behaviour of test case number 19.

22. Test Case 22: Same as test case number 3.

23. Test Case 23: Same as test case number 18.

Note that Randoop using some basic elements to create tests therefore if we want to have good tests we might let this tool generate more tests. As you see the quality of tests are being better and better.Furthermore, code itself handles so many exceptions thus Randoop when creating test at first, it will face with a lot of exceptions and slowly will generate some high-quality tests because each test will update these elements in repo.(Note that generating 1000 tests and examine all of them and also change the oracle is so time consuming and it is not reasonable for this homework)
Total number of changed tests has passed are illustrated in Figure 1:



```
⊗ RegressionTestImpl10   17 ms
  ✓ test01   4 ms
  ✓ test02   0 ms
  ✓ test03   0 ms
  ✓ test04   0 ms
  ✓ test05   0 ms
  ✓ test06   0 ms
  ✓ test07   1 ms
  ⊗ test08   6 ms
  ✓ test09   0 ms
  ✓ test10   1 ms
  ✓ test11   0 ms
  ⊗ test12   1 ms
  ✓ test13   1 ms
  ✓ test14   0 ms
  ✓ test15   0 ms
  ✓ test16   1 ms
  ✓ test17   0 ms
  ✓ test18   1 ms
  ✓ test19   0 ms
  ✓ test20   0 ms
  ✓ test21   0 ms
  ✓ test22   1 ms
  ✓ test23   0 ms
```

Figure 1: Modified Test results

1.2 Implementation ’Impl2’

We have generated 13 test cases for this implementation since it does not handling so much exceptions.Thus there are so many bugs lies with this implementation.These generated test and modified are as follow:

1. Test Case 1: This test case is totally wrong since we can not create a book with a empty name therefore it must have a exception which is not provided in generated test:

```java
    @Test
    public void test01() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test01");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
            ↪ ();
        int int3 = library0.getTotalPenalty("");
        library0.addBook("", 100);
        // The following exception was thrown during execution in test
            ↪ generation
        try {
            library0.addReference("", (int) 'a');
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: the reference has already
               ↪ added");
        } catch (ir.ramtung.sts01.LibraryException e) {
            // Expected exception.
        }
        org.junit.Assert.assertNotNull(strList1);
        org.junit.Assert.assertTrue("'" + int3 + "' != '" + 0 + "'", int3 ==
           ↪ 0);
    }
```

Add an try catch block:

```
        try {
            library0.addBook("", 100);
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: document with empty title")
               ↪ ;
        }catch (ir.ramtung.sts01.LibraryException e){
            org.junit.Assert.assertEquals(e.getMessage(), "document with
               ↪ empty title is not allowed");
        }
```

This test will not passed since developer did not fall an exception and violated specification number 15.

2. Test Case 2: This test case is also must not passed since it added an empty titled book and but other aspect of this test is true.(Since we change test case 1 for same reason, we avoid to modify this test case)

3. Test Case 3: This test case is testing adding a magazine with 0(as number) would throw an exception which is true.

4. Test Case 4: This test case shows a person who does not added to library can a document that does not exist while this situation must throw an exception. The generated test case is:

```
    @Test
    public void test04() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test04");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        library0.extend("", "hi!");
        library0.addBook("", (int) 'a');
        library0.borrow("hi!", "");
        library0.returnDocument("hi!", "");
        org.junit.Assert.assertNotNull(strList1);
    }
```

Modifying to throw an exception(Adding following exception):

```
        try{
            library0.extend("", "hi!");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: Member is not in a loan");
        }catch (ir.ramtung.sts01.LibraryException e){
            org.junit.Assert.assertEquals(e.getMessage(), "Member is not in a
               ↪  loan");
        }
```

5. Test Case 5: In this scenario a person called "hi" are borrowing a document with "hi" title which both does not exist!Therefore we adding following exception:

```
        try{
            library0.borrow("hi!", "hi!");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: Can not be borrowed");
        }catch (ir.ramtung.sts01.LibraryException e){
            org.junit.Assert.assertEquals(e.getMessage(), "Can not be
               ↪ borrowed");
        }
```

6. Test Case 6: This case behaviour is not correct thus it must added a exception.

7. Test Case 7: This scenario(except behaviour still wrong) is trying to add a magazine with negative copy which already has this.

8. Test Case 8: This test case is designed for adding duplicate book names will throw an exception or not(Note that empty titled book is not allowed and we avoid to modify it since we already modify a test case based on this situation)

9. Test Case 9: Here is not a well-quality test. The generated test is:

```
    @Test
    public void test09() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test09");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        library0.extend("", "");
        library0.borrow("", "hi!");
        org.junit.Assert.assertNotNull(strList1);
    }
```

As you can see it is a very bad test therefore we will modify it:

```java
    @Test
    public void test09() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test09");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        library0.addBook("hi", 10);
        library0.addProfMember("asd");

        library0.borrow("asd", "hi");
        try{
            library0.extend("asd", "asd");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: Can not extent on same day"
               ↪ );
        }catch (ir.ramtung.sts01.LibraryException e){
            org.junit.Assert.assertEquals(e.getMessage(), "Can not extent on
               ↪ same day");
        }
        org.junit.Assert.assertNotNull(strList1);
    }
```

This scenario is saying you can not extent a document on the same day and test is passed.

10. Test Case 10: Like previous test is not high-quality enough and we will modify it:

```java
    @Test
    public void test10() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test10");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        library0.borrow("", "hi!");
        org.junit.Assert.assertNotNull(strList1);
    }
```

We will modify it as follow:

```java
    @Test
    public void test10() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test10");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        library0.addBook("hi", 10);
        library0.addProfMember("asd");
        library0.borrow("asd", "hi");
        library0.timePass(2);
        int expected = library0.getTotalPenalty("asd");
        org.junit.Assert.assertEquals(expected, 0);
        library0.timePass(8);
        org.junit.Assert.assertEquals(expected, 0);
        int days = 25;
        library0.timePass(days);
        expected = library0.getTotalPenalty("asd");
        org.junit.Assert.assertEquals(expected, 7*2000 + 14*3000 + (days -
           ↪ 21)*5000);
        org.junit.Assert.assertNotNull(strList1);
    }
```

11. Test Case 11: This test case is kind of duplicated with test case number but it will check if year is greater than 0.

12. Test Case 12: As you can see in test below this test cases are getting better(But note that since bad elements added to the repo to create advanced tests, are wrong. For example have a book with empty title!)
    We will modify the generated test below to check the specification number 8.The generated test is:

```
    @Test
    public void test12() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test12");
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        int int3 = library0.getTotalPenalty("");
        library0.addBook("", 100);
        int int8 = library0.getTotalPenalty("hi!");
        library0.borrow("", "");
        library0.extend("hi!", "");
        library0.timePass(10);
        org.junit.Assert.assertNotNull(strList1);
        org.junit.Assert.assertTrue("'" + int3 + "' != '" + 0 + "'", int3 ==
           ↪ 0);
        org.junit.Assert.assertTrue("'" + int8 + "' != '" + 0 + "'", int8 ==
           ↪ 0);
    }
```

Modified test:

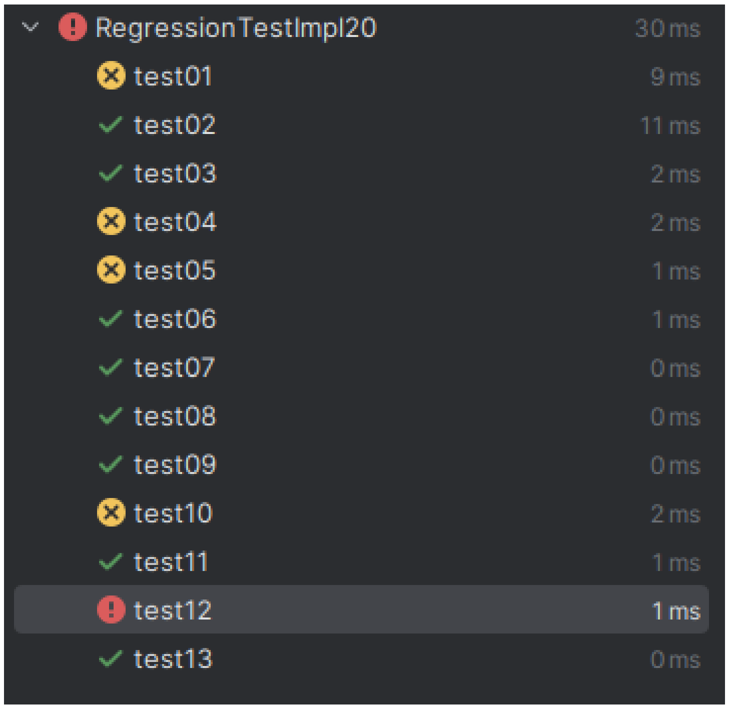
```
    @Test
    public void test12() throws Throwable {
        if (debug)
            System.out.format("%n%s%n", "RegressionTestImpl20.test12");
```

```java
        ir.ramtung.impl2.Library library0 = new ir.ramtung.impl2.Library();
        java.util.List<java.lang.String> strList1 = library0.availableTitles
           ↪ ();
        int int3 = library0.getTotalPenalty("");
        library0.addBook("", 100);
        int int8 = library0.getTotalPenalty("hi!");
        library0.borrow("", "");
        library0.extend("hi!", "");
        library0.timePass(10);
        library0.addBook("abs", 10);
        library0.addProfMember("abcd");
        library0.borrow("abcd", "abs");
        library0.timePass(2);
        library0.extend("abcd", "abs");
        library0.timePass(2);
        library0.extend("abcd", "abs");
        try{
            library0.extend("abcd", "abs");
            org.junit.Assert.fail("Expected exception of type ir.ramtung.
               ↪ sts01.LibraryException; message: Can not be extended more
               ↪ than two");
        }catch (ir.ramtung.sts01.LibraryException e) {
            org.junit.Assert.assertEquals(e.getMessage(), "Can not be
               ↪ extended more than two");
        }
        org.junit.Assert.assertNotNull(strList1);
        org.junit.Assert.assertTrue("'" + int3 + "' != '" + 0 + "'", int3 ==
           ↪ 0);
        org.junit.Assert.assertTrue("'" + int8 + "' != '" + 0 + "'", int8 ==
           ↪ 0);
    }
```

This test reveals a bug! Every time a member just can extend one time since it got an error try block.(The line before try block)

13. Test Case 13: This test case is good!

The result of modified tests are depicted in Figure 2:



```
RegressionTestImpl20   30 ms
    test01   9 ms
    test02   11 ms
    test03   2 ms
    test04   2 ms
    test05   1 ms
    test06   1 ms
    test07   0 ms
    test08   0 ms
    test09   0 ms
    test10   2 ms
    test11   1 ms
    test12   1 ms
    test13   0 ms
```

Figure 2: Modified tests result

1.3 Summary

Out of 13 test cases for this class, 5 tests has been failed. In summary, the Randoop is a great tool to generate test cases with following conditions:

- First elements in repository is good enough.
- If our code has so many bugs(e.g, 'impl2'), this tool will add bad stuffs into repository as new elements.It is better to fix the code and then re-generate tests with this tool.
- If our code has less bugs(e.g, 'impl1'), this tool will create good repository based on added elements. But we should get it time to generate more tests to reveal bugs(since bugs are fewer)

According to investigating generated tests, based on my opinion, it is good to someway guide this tool to generate good test cases. Since this will trace the execution and based on that will generate test cases, it would be good to consider using exception to avoid having bad elements added to repository. But for codes which has fewer bug, we should consider this tool have enough time to generate test cases.However, to generate like 1000 tests and investigating for this homework was very time consuming and indeed not reasonable thus we decide to generate fewer tests and modify them a little bit as description said.