软件测试上机报告

第一次上机作业

学　院__智能与计算学部__
专　业____软件工程_____
姓　名_____陈姝宇_____
学　号___3017218119_____
年　级_____2017 级_____
班　级_____3 班_____

1. Experimental Requirements

Tasks:

1. Install Junit(4.12), Hamcrest(1.3) with Eclipse/IDEA
2. Install Eclemma with Eclipse
3. Write a java program for the given problem and test the program with Junit.
   a) Description of the problem:

      There is one 50 yuan, one 20 yuan, one 10 yuan, two 5 yuan bills and three 1 yuan coins in your pocket. Write a program to find out whether you can take out a given number (x) yuan.

Requirements for the experiment:

1. Finish the tasks above individually.
2. Check in your java code and junit test program to github
3. Please send your experiment report to 'Wisdom Tree', the following information should be included in your report:
   a) The brief description that you install junit, hamcrest and eclemma.
   b) The test result and coverage report (print screen) of your tests on the problem.

2. Environmental configuration

[Step1] Add jar of junit, hamcrest for IDEA just like the figure1.

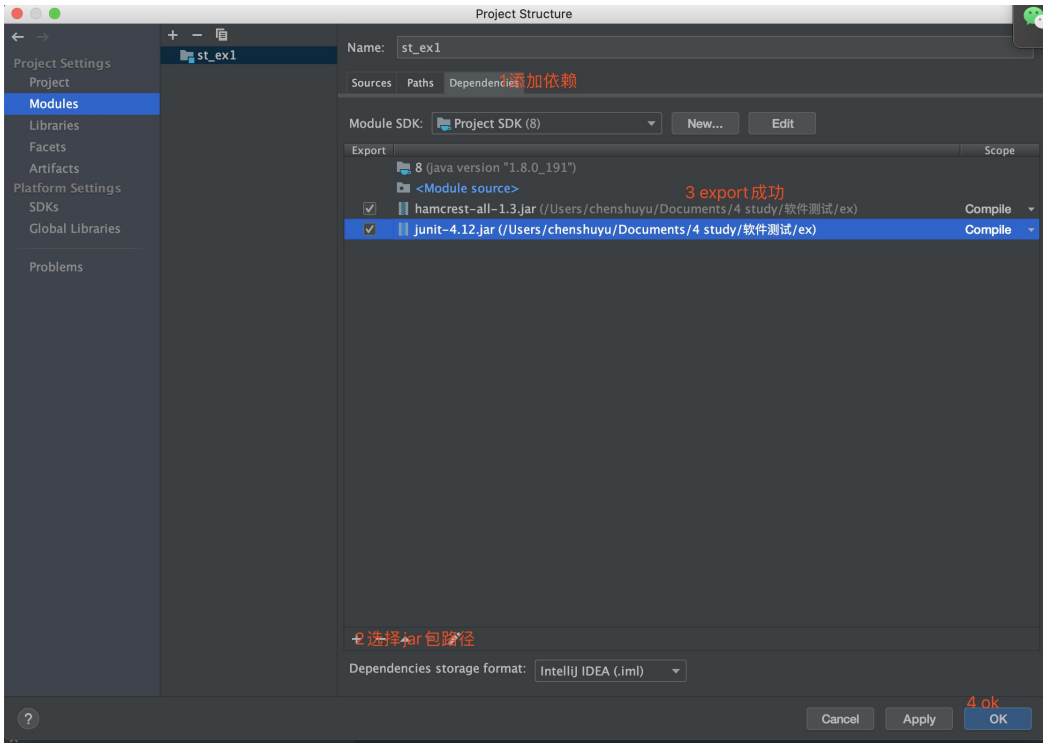


Figure 1 Add Dependencies

[Step2] Set properties for directory: set src directory as Sources Root and set test directory as Test Sources Root, like figure2

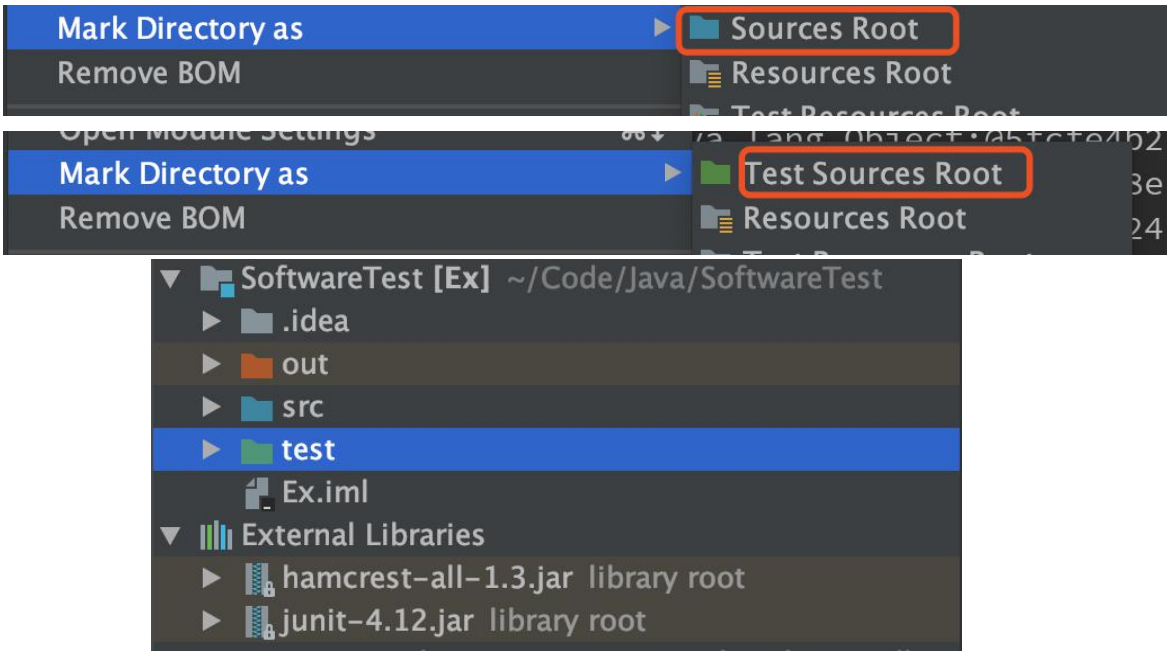


Figure 2 Set Propertities

3. Source Code

[src/com.chenshuyusc.ex1/Money.java]

```
public class Money {

    private static int[] money = {1, 1, 1, 5, 5, 10, 20, 50};// save all money

    /**
     * @param num given number (x) yuan.
     * @return whether can take out a given number (x) yuan.
     */
    public boolean findMoney(int num) {
        boolean result = false; // recored result

        if (num > 93) return false;  // exceed max money
        else if (num >= 50) { // must contain 50
            num = num - 50;
            result = findMoney(num);// num <= 43
        } else if (num > 43) { // 1+1+1+5+5+10+20=43
            return false;
        } else if (num > 23) { // 1+1+1+5+5+10=23
            // must contain 20
            num = num - 20;
            result = findMoney(num); // num <= 23
            // when num <= 23
            // whether 20 is used or not is not important
            // because when num <= 23,only when num >= 20 that 20 has
chance to be used
            // but at this time 20 can be replaced by 10+5+5
        } else {
            // num<=23
            // at this time only the last number of num is important
```

```
            // because 10+5+5=20, 10=10
            num = num % 10; // use this method to get the last number of
num
            // only has three 1, so num cannot be 4 or 9
            return num != 4 && num != 9;
        }
        return result;
    }

    //To find how many case
    public static Set<Integer> allCase() {
        // use class CopyOnWriteArraySet to modify set
        Set<Integer> possible = new CopyOnWriteArraySet<>();

        // initial set
        possible.add(0);
        for (int num : money) {
            for (Object aPossible : possible) {
                // not has
                int temp = (int) aPossible;
                // has
                possible.add(temp + num);
            }
        }
        // possible: 0 1 2 3 5 6 7 8 10 11 12 13 15 16 17 18 20 21 22 23
        // 25 26 27 28 30 31 32 33 35 36 37 38 40 41 42
        // 43 50 51 52 53 55 56 57 58 60 61 62 63 65 66 67 68 70 71 72
73 75 76 77 78 80 81 82 83 85 86 87 88 90 91 92 93
        return possible;
    }
}
```

[test/com.chenshuyusc.ex1/TestMoney.java]

```
@RunWith(Parameterized.class)
public class MoneyTest {
    private boolean expected;
    private int num;
    private static Money money = null;

    // initialized
    public MoneyTest(boolean expected, int num) {
        this.expected = expected;
        this.num = num;
    }

    @Before
    public void before() throws Exception {
        money = new Money();
    }

    /**
     * this function must be static function
     * @return test case
     */
    @Parameterized.Parameters
    public static Collection<Object[]> getData() {

        ArrayList<Object[]> datas = new ArrayList<Object[]>();
        Set<Integer> cases = Money.allCase(); // get all true case
        for (int i = 0; i < 100; i++) {
            datas.add(new Object[]{cases.contains(i), i});// i whether in set
        }
        return datas;
```

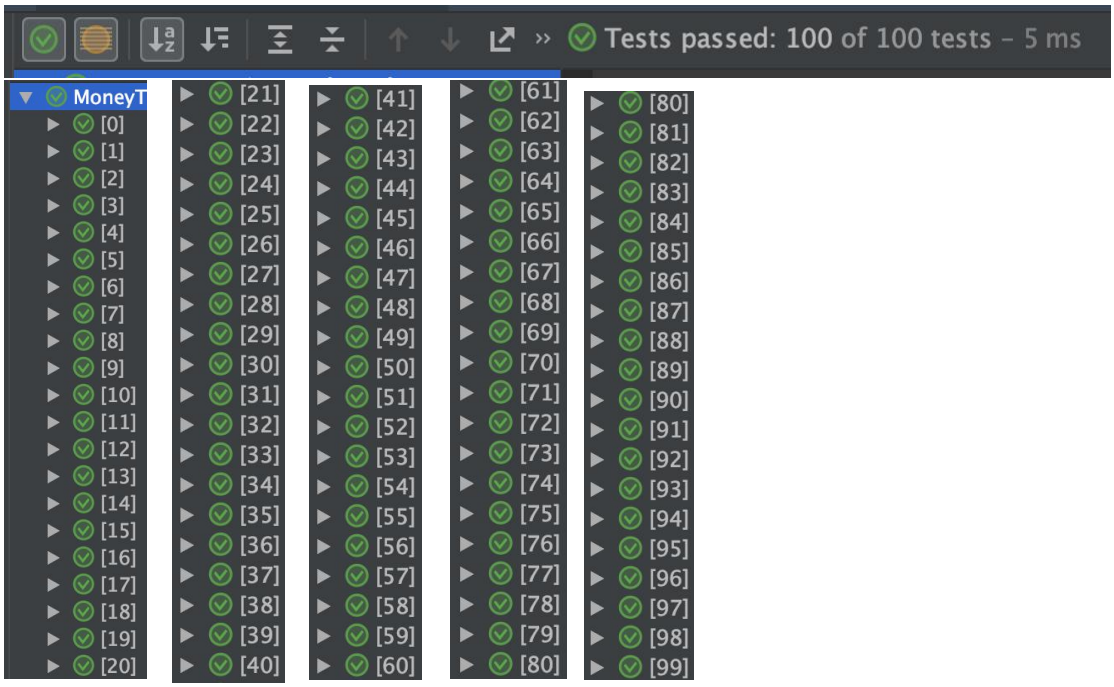
```
    }

    @After
    public void after() throws Exception {
    }

    /**
     * Method: findMoney(int num)
     */
    @Test
    public void testFindMoney() throws Exception {
        Assert.assertEquals(this.expected, money.findMoney(this.num));
    }
}
```

4. Operation Result

Passed all 100 test cases in 5ms.



Run with coverage: Coverage all class, all method and all line.