1.Experimental requirements

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.

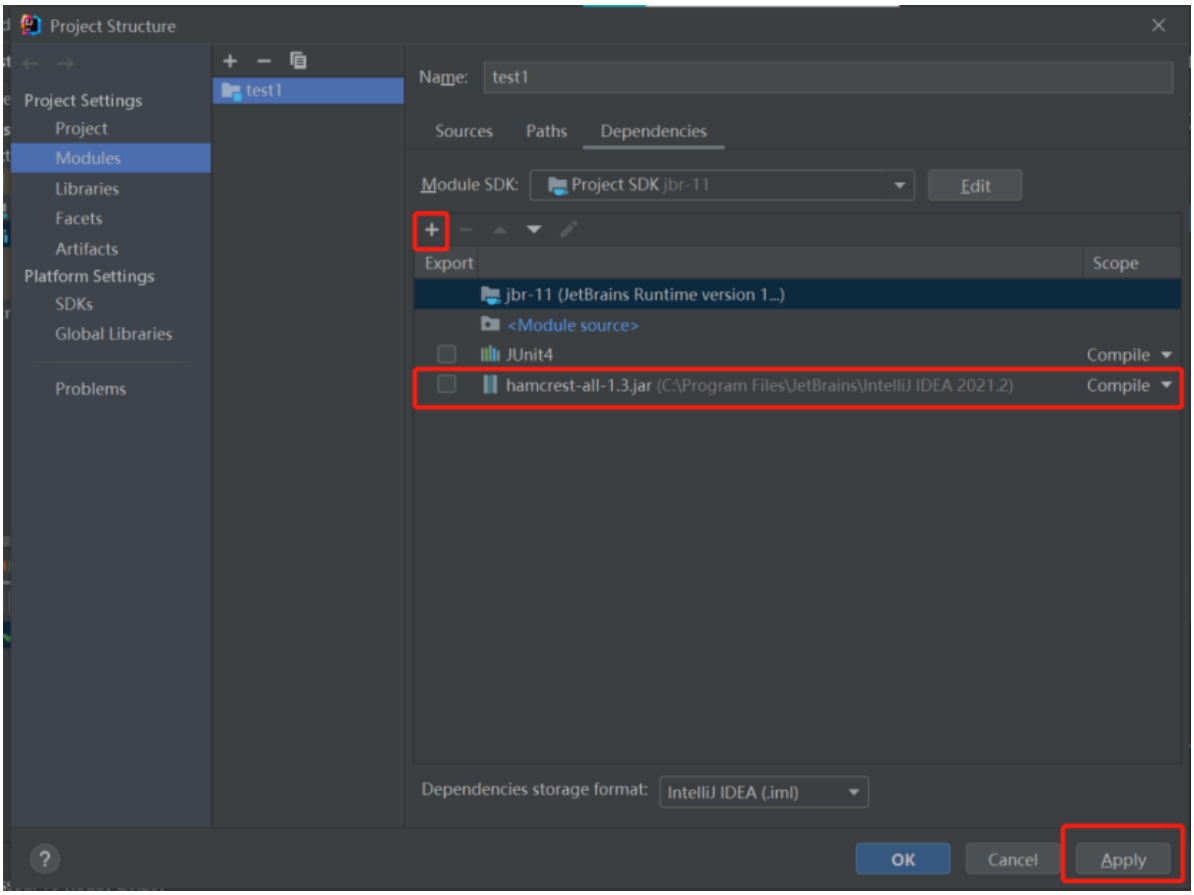
   a) Description of the problem:

   There are 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.

2. Configuration

在IDEA上安装Junit，建立好项目后，输入@Test，会报错，然后根据提示安装JUnit，选择4.12版本，同时hamcrest-core-1.3也被安装，如图所示

File->Project Structure->Modules



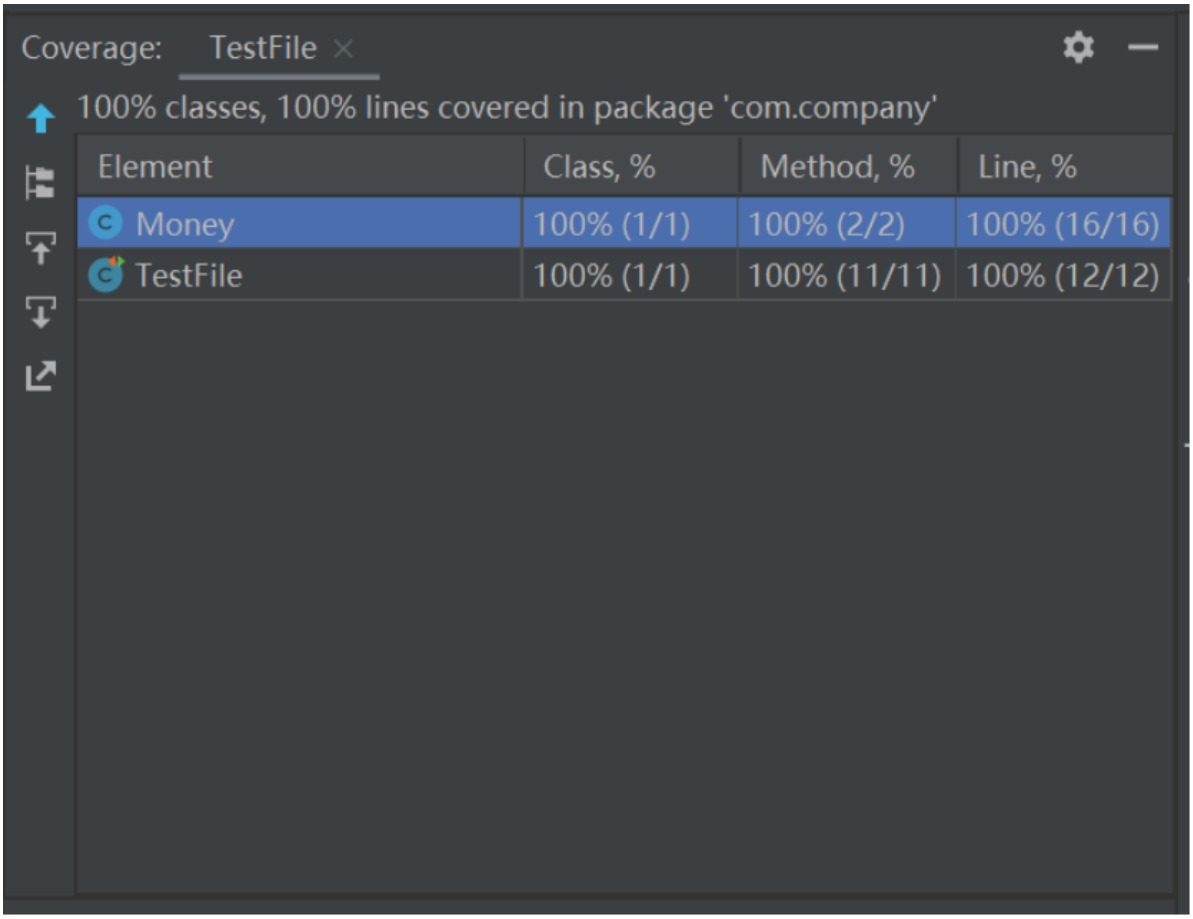
选择hamcrest-all-1.3的位置，添加到环境中。

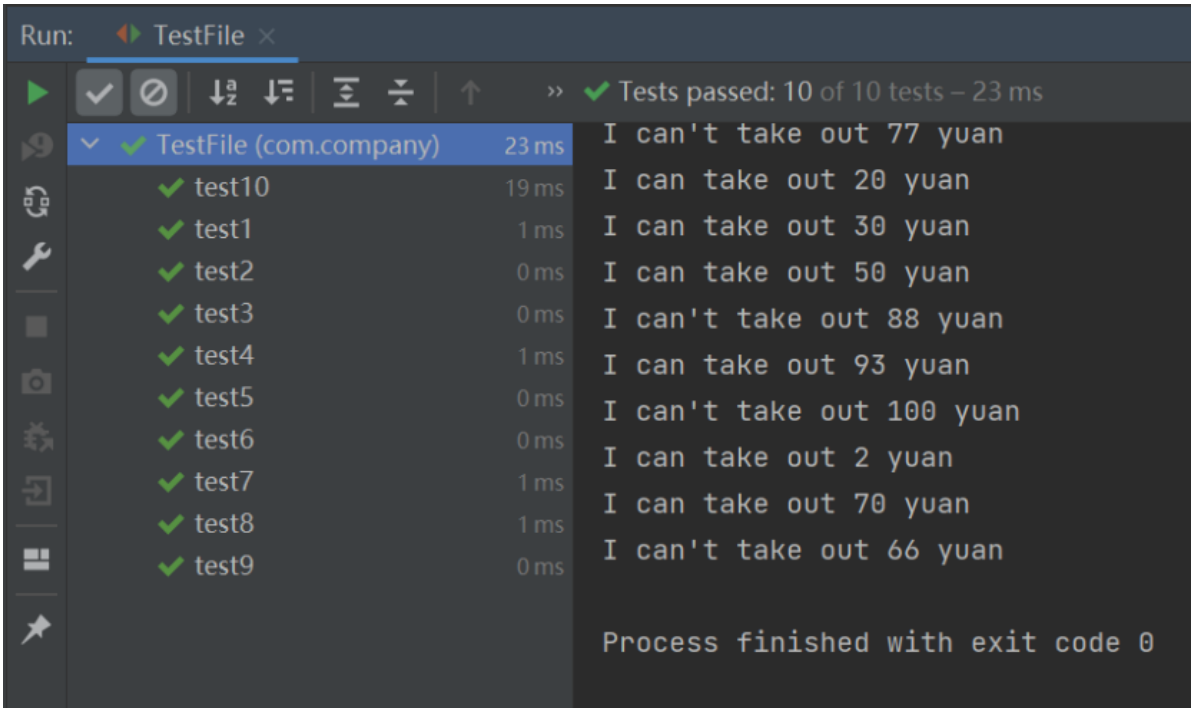
任务1，Junit(4.12), Hamcrest(1.3)安装完毕

3. Result analysis

题目分析：对于能拿出的金额，将零钱存在数组pocket中，通过双层for循环，将所有的金额依次相加，再存入另一个数组中，若重复则不存入，从而得到了可以拿出的金额的所有可能情况。然后写一个判断函数，如果测试的金额在结果数组中，则返回true，否则但会false。然后写测试类，进行测试即可。

结果截图如下：





4. Source code

Money.java

```java
package com.company;

import java.util.ArrayList;

public class Money {
    public int[] pocket ={1,1,1,5,5,10,20,50};
    public ArrayList<Integer> sum = new ArrayList<>();

    public ArrayList<Integer> getMoney(){
        for(int i = 0;i<pocket.length;i++){
            int m=pocket[i];
            if(!sum.contains(m))
            {
                sum.add(m);
            }
            for(int j=i+1;j< pocket.length;j++){
                m += pocket[j];
                if(!sum.contains(m))
                {
                    sum.add(m);
                }
            }
        }
        return sum;
//        System.out.println(sum);
    }

    public boolean canTakeMoney(int money){
        if(getMoney().contains(money)){
            System.out.println("I can take out "+money+" yuan");
            return true;
        }
        else {
            System.out.println("I can't take out "+money+" yuan");
            return false;
        }
    }
}

```

TestFile.java

```java
package com.company;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;


public class TestFile {

    public Money money;

```

```
    @Before
    public void setup() {
        money = new Money();
    }

    @Test
    public void test1(){
        Assert.assertTrue(money.canTakeMoney(20));
    }

    @Test
    public void test2(){
        Assert.assertTrue(money.canTakeMoney(30));
    }


    @Test
    public void test3(){
        Assert.assertTrue(money.canTakeMoney(50));
    }

    @Test
    public void test4(){
        Assert.assertFalse(money.canTakeMoney(88));
    }

    @Test
    public void test5(){
        Assert.assertTrue(money.canTakeMoney(93));
    }

    @Test
    public void test6(){
        Assert.assertFalse(money.canTakeMoney(100));
    }

    @Test
    public void test7(){
        Assert.assertTrue(money.canTakeMoney(2));
    }

    @Test
    public void test8(){
        Assert.assertTrue(money.canTakeMoney(70));
    }

    @Test
    public void test9(){
        Assert.assertFalse(money.canTakeMoney(66));
    }

    @Test
    public void test10(){
        Assert.assertFalse(money.canTakeMoney(77));
    }
}

```