测试报告

本项目所有测试都使用 Junit 进行，测试覆盖了所有的方法。

代码覆盖率图

测试通过图

FractionS2FTest 测试字符串与分数互换功能

```java
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;

import java.util.Arrays;
import java.util.Collection;

import static org.junit.Assert.assertEquals;

@RunWith(value=Parameterized.class)
public class FractionS2FTest {
    private String input;
    private String output;

    @Parameterized.Parameters
    public static Collection data() {
        return Arrays.asList(new Object[][] {
                {"0","0"},
                {"1/4","1/4"},
                {"2/4","1/2"},
                {"3/4","3/4"},
                {"4/4","1"},
                {"5/4","1'1/4"},
                {"6/4","1'1/2"},
                {"7/4","1'3/4"},
                {"8/4","2"},
                {"9/4","2'1/4"},
                {"10/4","2'1/2"},
                {"11/4","2'3/4"},
                {"12/4","3"},
                {"13/4","3'1/4"},
                {"14/4","3'1/2"},
                {"15/4","3'3/4"}
        });
    }

    public FractionS2FTest(String input, String output) {
        this.input = input;
        this.output = output;
    }

    @Test
    public void test() {
        System.out.println("Input: " + input + " Output: " + output);
        assertEquals(Fraction.string2Fraction(input).toString(), output);
        assertEquals(Fraction.string2Fraction(output).toString(), output);
    }
}
```

FractionConstructorTest 测试分数构造函数和化简功能

```
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;

import java.util.Arrays;
import java.util.Collection;

import static org.junit.Assert.assertEquals;

@RunWith(value=Parameterized.class)
public class FractionConstructorTest {
    private String expected;
    private int numerator;
    private int denominator;

    @Parameterized.Parameters
    public static Collection data() {
        return Arrays.asList(new Object[][] {
                {0,4,"0"},
                {1,4,"1/4"},
                {2,4,"1/2"},
                {3,4,"3/4"},
                {4,4,"1"},
                {5,4,"1'1/4"},
                {6,4,"1'1/2"},
                {7,4,"1'3/4"},
                {8,4,"2"},
                {9,4,"2'1/4"},
                {10,4,"2'1/2"},
                {11,4,"2'3/4"},
                {12,4,"3"},
                {13,4,"3'1/4"},
                {14,4,"3'1/2"},
                {15,4,"3'3/4"}
        });
    }

    public FractionConstructorTest(int numerator, int denominator, String
expected) {
        this.expected = expected;
        this.numerator = numerator;
        this.denominator = denominator;
    }

    @Test
    public void test() {
        String result = new Fraction(numerator,denominator).toString();
        System.out.println("The input fraction is: " + numerator + "/" +
denominator + " and expression is: " + result);
        assertEquals(expected, result);
    }
}
```

FractionArithmeticTest 测试分数四则运算和算术式整体运算功能

```
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class FractionArithmeticTest {
    @Test
    public void addTest() {
        assertEquals(Fraction.string2Fraction("44/5").toString(),
Fraction.string2Fraction("8/1").add(Fraction.string2Fraction("4/5")).toString())
;
        assertEquals(Fraction.string2Fraction("2/1").toString(),
Fraction.string2Fraction("0/4").add(Fraction.string2Fraction("6/3")).toString())
;
        assertEquals(Fraction.string2Fraction("79/24").toString(),
Fraction.string2Fraction("8/3").add(Fraction.string2Fraction("5/8")).toString())
;
        assertEquals(Fraction.string2Fraction("2/1").toString(),
Fraction.string2Fraction("8/6").add(Fraction.string2Fraction("4/6")).toString())
;
        assertEquals(Fraction.string2Fraction("3/7").toString(),
Fraction.string2Fraction("3/7").add(Fraction.string2Fraction("0/5")).toString())
;
    }

    @Test
    public void subtractTest() {
        assertEquals(Fraction.string2Fraction("47/56").toString(),
Fraction.string2Fraction("9/8").subtract(Fraction.string2Fraction("2/7")).toStri
ng());
        assertEquals(Fraction.string2Fraction("7/6").toString(),
Fraction.string2Fraction("5/3").subtract(Fraction.string2Fraction("1/2")).toStri
ng());
        assertEquals(Fraction.string2Fraction("1/1").toString(),
Fraction.string2Fraction("2/1").subtract(Fraction.string2Fraction("6/6")).toStri
ng());
        assertEquals(Fraction.string2Fraction("1/2").toString(),
Fraction.string2Fraction("2/3").subtract(Fraction.string2Fraction("1/6")).toStri
ng());
        assertEquals(Fraction.string2Fraction("1/2").toString(),
Fraction.string2Fraction("3/6").subtract(Fraction.string2Fraction("0/3")).toStri
ng());
    }

    @Test
    public void multiplyTest() {
        assertEquals(Fraction.string2Fraction("0/1").toString(),
Fraction.string2Fraction("0/9").multiply(Fraction.string2Fraction("8/4")).toStri
ng());
        assertEquals(Fraction.string2Fraction("0/1").toString(),
Fraction.string2Fraction("6/5").multiply(Fraction.string2Fraction("0/4")).toStri
ng());
```

```java
        assertEquals(Fraction.string2Fraction("16/63").toString(),
Fraction.string2Fraction("4/9").multiply(Fraction.string2Fraction("4/7")).toStri
ng());
        assertEquals(Fraction.string2Fraction("7/8").toString(),
Fraction.string2Fraction("6/8").multiply(Fraction.string2Fraction("7/6")).toStri
ng());
        assertEquals(Fraction.string2Fraction("28/27").toString(),
Fraction.string2Fraction("8/9").multiply(Fraction.string2Fraction("7/6")).toStri
ng());
    }

    @Test
    public void divideTest() {
        assertEquals(Fraction.string2Fraction("0/1").toString(),
Fraction.string2Fraction("0/7").divide(Fraction.string2Fraction("13/4")).toStrin
g());
        assertEquals(Fraction.string2Fraction("7/5").toString(),
Fraction.string2Fraction("3/5").divide(Fraction.string2Fraction("3/7")).toString
());
        assertEquals(Fraction.string2Fraction("12/5").toString(),
Fraction.string2Fraction("6/1").divide(Fraction.string2Fraction("5/2")).toString
());
        assertEquals(Fraction.string2Fraction("32/9").toString(),
Fraction.string2Fraction("8/3").divide(Fraction.string2Fraction("3/4")).toString
());
        assertEquals(Fraction.string2Fraction("15/4").toString(),
Fraction.string2Fraction("6/8").divide(Fraction.string2Fraction("1/5")).toString
());
    }

    @Test
    public void isZeroTest() {
        assertEquals(true, Fraction.string2Fraction("0/4").isZero());
    }

    @Test
    public void isGreaterThanTest() {
        assertEquals(true,
Fraction.string2Fraction("5/4").isGreaterThan(Fraction.string2Fraction("1/6")));
        assertEquals(false,
Fraction.string2Fraction("1/2").isGreaterThan(Fraction.string2Fraction("1/2")));
        assertEquals(false,
Fraction.string2Fraction("2/8").isGreaterThan(Fraction.string2Fraction("3/8")));
    }

    @Test
    public void calculateStringExpTest() {
        assertEquals("60'43/60", Fraction.calculateStringExp("(4/5 + 3'1/4) +
(8'8/9 × 6'3/8)").toString());
        assertEquals("0", Fraction.calculateStringExp("(1 × 0) ÷ (6 +
3/8)").toString());
        assertEquals("49'205/294", Fraction.calculateStringExp("2'5/7 × (3'1/2 +
(6'1/7 + 8'2/3))").toString());
        assertEquals("5'1/10", Fraction.calculateStringExp("3'2/3 × 3 - 1'1/2 -
4'2/5").toString());
        assertEquals("0", Fraction.calculateStringExp("(4'1/2 - (1/2 + 4)) ×
0").toString());
```

```
        assertEquals("1", Fraction.calculateStringExp("(5/7 + 1'1/4) - (1'5/7 -
3/4)").toString());
        assertEquals("196/375", Fraction.calculateStringExp("4'1/5 × ((4/5 -
1/3) ÷ 3'3/4)").toString());
        assertEquals("3/7", Fraction.calculateStringExp("(1'3/4 × 1'1/3) ÷
(3'4/9 + 2)").toString());
        assertEquals("15'7/9", Fraction.calculateStringExp("2 × (4'8/9 + 3) -
0").toString());
        assertEquals("23/35", Fraction.calculateStringExp("(1 + 5'4/7) ÷ (4 ×
2'1/2)").toString());
        assertEquals("10/107", Fraction.calculateStringExp("1 ÷ (2'3/4 + 2'3/5)
× 1/2").toString());
        assertEquals("1'1/42",Fraction.calculateStringExp("2'6/7 - 2'2/3 - 1/6 +
1").toString());
        assertEquals("1'26/35", Fraction.calculateStringExp("(3'3/5 - 1'6/7) -
(0 × 1/2)").toString());
        assertEquals("0", Fraction.calculateStringExp("1'1/6 × 5'1/7 × 0 +
0").toString());
        assertEquals("1'17/114", Fraction.calculateStringExp("1'1/5 ÷ 3'4/5 + 4
- 3'1/6").toString());
        assertEquals("44/45", Fraction.calculateStringExp("4'8/9 ÷ 5 × 1 ÷
1").toString());
        assertEquals("65'1/3", Fraction.calculateStringExp("(5'1/3 + 0) × (4'3/8
× 2'4/5)").toString());
        assertEquals("2'4/7", Fraction.calculateStringExp("2 ÷ 3'1/2 +
2").toString());
        assertEquals("24'17/60", Fraction.calculateStringExp("1'1/5 + 5'2/3 ×
3'1/2 + 3'1/4").toString());
        assertEquals("5'7/12", Fraction.calculateStringExp("(4 - 2'1/4) + (4'5/6
- 1)").toString());
    }

}
```

FractionTest 分数测试汇总类，把上方三个测试归一调用

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(value = Suite.class)
@Suite.SuiteClasses(value={
        FractionArithmeticTest.class,
        FractionConstructorTest.class,
        FractionS2FTest.class})
public class FractionMasterTest {
}
```

ExerciseGeneratorTest 测试算术式生成类

```
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
```

```java
public class ExerciseGeneratorTest {
    ExerciseGenerator exGen;

    public ExerciseGeneratorTest() {
        exGen = new ExerciseGenerator(10, 1000);
    }

    @Test
    public void fractionGeneratorTest() {
        Fraction f;
        for(int i = 0;i < 100;i++) {
            f = exGen.fractionGenerator();
            assertTrue(Fraction.string2Fraction("10/1").isGreaterThan(f));
            assertFalse(Fraction.string2Fraction("0/1").isGreaterThan(f));
        }
    }

    @Test
    public void opGeneratorTest() {
        int add = 0;
        int sub = 0;
        int mul = 0;
        int div = 0;
        for(int i = 0;i < 100;i++) {
            switch(exGen.opGenerator()){
                case '+': add++; break;
                case '-': sub++; break;
                case '×': mul++; break;
                case '÷': div++; break;
            }
        }
        assertTrue(sub > 0);
        assertTrue(add > 0);
        assertTrue(mul > 0);
        assertTrue(div > 0);
    }

    @Test
    public void oneOpGeneratorTest() {
        String[] ss;
        for(int i = 0;i < 100;i++) {
            ss = exGen.oneOpGenerator();
            for(String s:ss){
                System.out.println(s);
            }
            System.out.println();
        }
    }

    @Test
    public void twoOpGeneratorTest() {
        String[] ss;
        for(int i = 0;i < 100;i++) {
            ss = exGen.twoOpGenerator();
            for(String s:ss){
                System.out.println(s);
            }
            System.out.println();
        }
    }

    @Test
    public void threeOpGeneratorATest() {
        String[] ss;
        for(int i = 0;i < 100;i++) {
            ss = exGen.threeOpGeneratorA();
            for(String s:ss){
                System.out.println(s);
            }
            System.out.println();
        }
    }

    @Test
    public void threeOpGeneratorBTest() {
        String[] ss;
        for(int i = 0;i < 100;i++) {
            ss = exGen.threeOpGeneratorB();
            for(String s:ss){
                System.out.println(s);
            }
            System.out.println();
        }
    }

    @Test
    public void duplicateCheckTest() {
        String[] ss;
        int saveOne = 0;
        int saveTwo = 0;
        int saveThreeA = 0;
        int saveThreeB = 0;
        for(int i = 0;i < 100000;i++) {
            if(i < 1000) {
                ss = exGen.oneOpGenerator();
                if (exGen.duplicateCheck(ss[0])) {
                    System.out.println("这是第 " + (i + 1) + " 个算术式");
                    for (String s : ss) {
                        System.out.println(s);
                    }
                    saveOne++;
                }
            }
            else if(i < 10000){
                ss = exGen.twoOpGenerator();
                if (exGen.duplicateCheck(ss[0])) {
                    System.out.println("这是第 " + (i + 1) + " 个算术式");
                    for (String s : ss) {
                        System.out.println(s);
                    }
                    saveTwo++;
                }
            }
            else if(i < 50000){
                ss = exGen.threeOpGeneratorA();
                if (exGen.duplicateCheck(ss[0])) {
                    System.out.println("这是第 " + (i + 1) + " 个算术式");
                    for (String s : ss) {
                        System.out.println(s);
                    }
                    saveThreeA++;
                }
            }
            else{
                ss = exGen.threeOpGeneratorB();
                if (exGen.duplicateCheck(ss[0])) {
                    System.out.println("这是第 " + (i + 1) + " 个算术式");
                    for (String s : ss) {
                        System.out.println(s);
                    }
                    saveThreeB++;
                }
            }

        }
        System.out.println("一共保留了 " + saveOne + " 个 1 型表达式");
        System.out.println("一共保留了 " + saveTwo + " 个 2 型表达式");
        System.out.println("一共保留了 " + saveThreeA + " 个 3A 型表达式");
        System.out.println("一共保留了 " + saveThreeB + " 个 3B 型表达式");
    }

    @Test
    public void generateExpTest() {
        HashMap<String,String> result = exGen.generateExp();
        if(result != null) {
            System.out.println("最终生成的题目和答案共 " + result.size() + " 对");
            for(Map.Entry<String, String> entry: result.entrySet())
            {
                System.out.println("式子: "+ entry.getKey()+ " 答案:
"+entry.getValue());
            }
        }
        else {
            System.out.println("range 和 number 不匹配");
        }
    }
}
```

UserInterfaceTest 测试用户接口类

```
import org.junit.Test;

public class UserInterfaceTest {
    @Test
    public void mainTest(){
        UserInterface.mainTest(new String[] {"-r","2","-n","2000"});
        UserInterface.mainTest(new String[] {"-r","1","-n","1"});
        UserInterface.mainTest(new String[] {"-r","10","-n","1000"});
        UserInterface.mainTest(new String[] {"-e","Exercise.txt","-
a","Answer.txt"});
        UserInterface.mainTest(new String[] {"-e","ExercisesF.txt","-
a","AnswersF.txt"});
    }
}
```

Calculate 软件整体测试，为了检查代码覆盖率

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(value = Suite.class)
@Suite.SuiteClasses(value={
        FractionMasterTest.class,
        ExerciseGeneratorTest.class,
        UserInterfaceTest.class})
public class MyAppMasterTest {
}
```