Баг-репорт

Проект: PR9

Модуль: CashBackCounting

Класс: CashBackServiceTest

Серьезность: Высокая

Приоритет: Высокий

Описание:

В ходе выполнения юнит-тестов для класса `CashBackCounting` было выявлено, что метод `remain` возвращает некорректные результаты для некоторых входных значений. Тесты, проверяющие эти значения, не проходят.

Шаги для воспроизведения:

1. Создайте класс `CashBackCounting` со следующим кодом:

```java

package org.example;

public class CashBackCounting {

private final int boundary = 1000;

public int remain(int amount) {

return boundary - amount % boundary;

}

}

```

2. Создайте тестовый класс `CashBackServiceTest` со следующими тестами:

```java

package org.example;

import org.testng.annotations.Test;

import static org.testng.Assert.assertEquals;

public class CashBackServiceTest {

@Test

public void shouldCalculateForAmount900AndGetExtra100() {

CashBackCounting service = new CashBackCounting();

int expected = 100;

int actual = service.remain(900);

assertEquals(actual, expected);

}

@Test

public void shouldCalculateForAmount1000AndGetExtra0() {

CashBackCounting service = new CashBackCounting();

int expected = 0;

int actual = service.remain(1000);

assertEquals(actual, expected);

}

@Test

public void shouldCalculateForAmount1100AndGetExtra900() {

CashBackCounting service = new CashBackCounting();

int expected = 900;

int actual = service.remain(1100);

assertEquals(actual, expected);

}

@Test

public void shouldCalculateForAmount2000AndGetExtra0() {

CashBackCounting service = new CashBackCounting();

int expected = 0;

int actual = service.remain(2000);

assertEquals(actual, expected);

}

@Test

public void shouldCalculateForAmount0AndGetExtra1000() {

CashBackCounting service = new CashBackCounting();

int expected = 1000;

int actual = service.remain(0);

assertEquals(actual, expected);

}

}

```

3. Запустите тесты с помощью Gradle или другой подходящей системы сборки.

Фактический результат:

Тесты `shouldCalculateForAmount1000AndGetExtra0` и `shouldCalculateForAmount2000AndGetExtra0` завершаются с ошибкой. Метод `remain` возвращает 1000 вместо ожидаемого значения 0.

Ожидаемый результат:

Тесты должны успешно завершаться, а метод `remain` должен возвращать правильные значения в соответствии с ожидаемым результатом.

Логи и сообщения об ошибках:

```

Expected :0

Actual :1000

<Click to see difference>

java.lang.AssertionError: expected [0] but found [1000]

at org.testng.Assert.fail(Assert.java:97)

at org.testng.Assert.assertEqualsImpl(Assert.java:136)

at org.testng.Assert.assertEquals(Assert.java:118)

at org.testng.Assert.assertEquals(Assert.java:839)

at org.testng.Assert.assertEquals(Assert.java:849)

at org.example.CashBackServiceTest.shouldCalculateForAmount1000AndGetExtra0(CashBackServiceTest.java:21)

...

```

```

Expected :0

Actual :1000

<Click to see difference>

java.lang.AssertionError: expected [0] but found [1000]

at org.testng.Assert.fail(Assert.java:97)

at org.testng.Assert.assertEqualsImpl(Assert.java:136)

at org.testng.Assert.assertEquals(Assert.java:118)

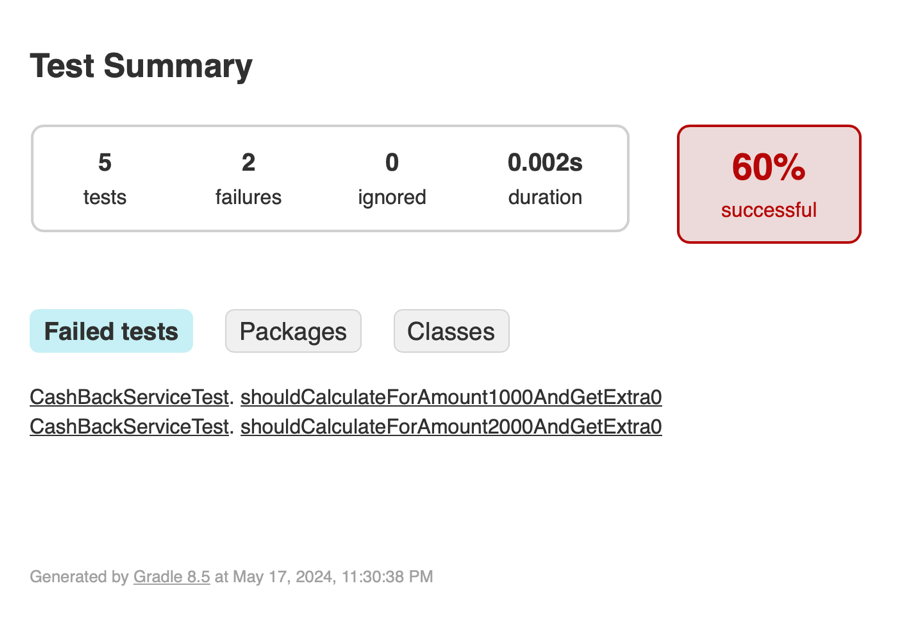
at org.testng.Assert.assertEquals(Assert.java:839)

at org.testng.Assert.assertEquals(Assert.java:849)

at org.example.CashBackServiceTest.shouldCalculateForAmount2000AndGetExtra0(CashBackServiceTest.java:37)

...

```



Возможная причина:

Метод `remain` в текущей реализации некорректно обрабатывает случаи, когда `amount` кратно `boundary` (например, 1000, 2000). В этих случаях результат должен быть 0, но возвращается значение `boundary`.

Рекомендации по исправлению:

Изменить метод `remain`, добавив проверку для случаев, когда `amount` кратно `boundary`:

```java

package org.example;

public class CashBackCounting {

private final int boundary = 1000;

public int remain(int amount) {

int remainder = amount % boundary;

return remainder == 0 ? 0 : boundary - remainder;

}

}

```