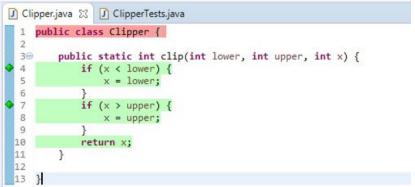
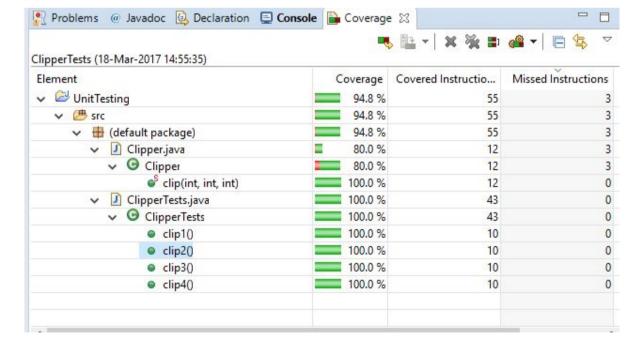
```
☑ Clipper.java
☑ ClipperTests.java
  1⊕ import static org.junit.Assert.*;[]
  4 public class ClipperTests {
  60
               @Test
               public void clip1() {
    int x = Clipper.clip(1, 9, 0);
  8
                        assertEquals(1, x);
 10
 11
               @Test
 120
               public void clip2() {
    int x = Clipper.clip(1, 9, 10);
    assertEquals(9, x);
 14
 15
 16
189
               @Test
               public void clip3() {
    iht x = Clipper.clip(1, 9, 5);
 19
 20
                        assertEquals(5, x);
22
 23
 240
               @Test
               public void clip4() {
    int x = Clipper.clip(9, 1, 5);
 25
 26
                        assertEquals(1, x);
 27
 28
 29 }
 30
```

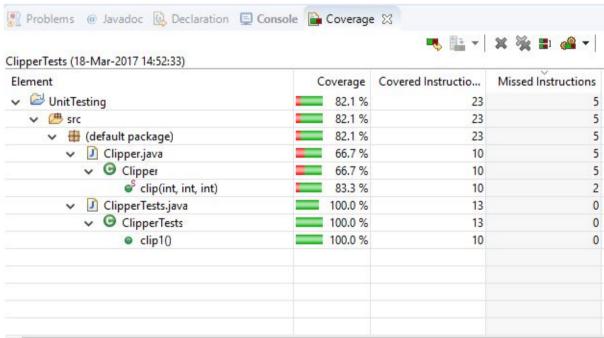




```
J Clipper.java
 1⊕ import static org.junit.Assert.*;[]
 4 public class ClipperTests {
 5
 60
           @Test
 7
           public void clip1() {
 8
                   int x = Clipper.clip(1, 9, 0);
 9
                   assertEquals(1, x);
10
11
12
    }
13

    Clipper.java 
    □ ClipperTests.java

      public class Clipper {
           public static int clip(int lower, int upper, int x) {
   30
   4
               if (x < lower) {
                    x = lower;
   5
   6
   7
               if (x > upper) {
   8
                    x = upper;
   9
  10
               return x;
  11
           }
  12
  13
      }
```



```
public class MorgageTests {
   MTest
    public void Test1() {
            assertEquals((int) (7* 1000), Mortgage.calculateMortgage(25, 1000, Mortgage.Gender.FEMALE));
   @Test
    public void Test2() {
           assertEquals((int) (5* 1000), Mortgage.calculateMortgage(39, 1000, Mortgage.Gender.FEMALE));
   @Test
    public void Test3() {
           assertEquals((int) (3* 1000), Mortgage.calculateMortgage(48, 1000, Mortgage.Gender.FEMALE));
   @Test
   public void Test4() {
           assertEquals(0, Mortgage.calculateMortgage(59, 1000, Mortgage.Gender.FEMALE));
   @Test
    public void Test5() {
           assertEquals(0, Mortgage.calculateMortgage(17, 1000, Mortgage.Gender.FEMALE));
    public void Test6() {
           assertEquals((int) (7.5* 1000), Mortgage.calculateMortgage(25, 1000, Mortgage.Gender.MALE));
   @Test
    public void Test7() {
           assertEquals((int) (5.5* 1000), Mortgage.calculateMortgage(39, 1000, Mortgage.Gender.MALE));
   @Test
    public void Test8() {
           assertEquals((int) (3.5* 1000), Mortgage.calculateMortgage(48, 1000, Mortgage.Gender.MALE));
   }
   @Test
    public void Test9() {
           assertEquals(0, Mortgage.calculateMortgage(59, 1000, Mortgage.Gender.MALE));
   @Test
    public void Test10() {
           assertEquals(0, Mortgage.calculateMortgage(17, 1000, Mortgage.Gender.MALE));
   @Test
    public void Test11() {
            assertEquals(0, Mortgage.calculateMortgage(17, 1000, null));
```

```
public class Mortgage {
    public enum Gender {
       MALE, FEMALE;
    public static int calculateMortgage(int age, int salary, Gender gender) {
        double factor = 0;
        if (gender == Mortgage.Gender.FEMALE) {
           if (18 <= age & age <= 35) {
               factor = 7;
            } else if (36 <= age & age < 45) {
               factor = 5;
            } else if (45 < age & age <= 55) {
               factor = 3;
        } else if (gender == Mortgage.Gender.MALE) {
           if (18 <= age & age <= 35) {
               factor = 7.5;
            } else if (35 < age & age <= 45) {
               factor = 5.5;
            } else if (45 < age & age <= 55) {
               factor = 3.5;
        return (int)(salary * factor);
    }
}
```

Element		Coverage	Covered Instructio	Missed Instructions	Total Instructions
>	☑ ClipperTests.java	0.0 %	0	43	43
	☑ Mortgage.java	85.7 %	144	24	168
	→ O Mortgage ■	97.5 %	115	3	118
	> 😉 Gender	58.0 %	29	21	50
	calculateMortgage(int, int, Ger	100.0 %	115	0	115
)	☑ Clipper.java	0.0 %	0	15	15
· ·	☑ MorgageTests.java	100.0 %	91	0	91
	→ O MorgageTests ■	100.0 %	91	0	91
	Test1()	100.0 %	8	0	8
	Test10()	100.0 %	8	0	8
	Test11()	100.0 %	8	0	8
	Test2()	100.0 %	8	0	8
	Test3()	100.0 %	8	0	8
	Test4()	100.0 %	8	0	8
	● Test5()	100.0 %	8	0	8
	● Test6()	100.0 %	8	0	8
	Test7()	100.0 %	8	0	8
	Test8()	100.0 %	8	0	8
	Test9()	100.0 %	8	0	8