

**Unit Test at Range.java:**

**Name:** **Mohamad Abdalmajed Omar Idrees.**

**-------------------------------------------------------**

Describe the issues in the code:

**1-Combine Method:**

**1-First Case:**

Range R1 =new Range (-20.0,10);  
Range R2=new Range (-15.0,15.0);  
Range Real=Range.*combine*(R1, R2);  
*assertTrue*(Real.getLowerBound()==-20.0); // 🡪 Correct   
// failed, the function should return True when the upper is 15.0 but the function return false   
assertTrue (Real.getUpperBound()==15.0); // -> Not Correct

**In this Method all work correct except that combine not**

**Return the larger upper Range right.**

**2-Second Case:**

Range Real\_2=Range.*combine*(null,R1);  
*assertTrue*(Real\_2.getLowerBound()==-20.0); // -> Correct  
*assertTrue*(Real\_2.getUpperBound()==10.0); // -> Correct

**3-Third Case:**

Range Real\_3=Range.*combine*(null,null);  
*assertNull*(Real\_3);->Correct  
*assertTrue*(Real\_3==null);->Correct

**2-Constrain Method:**

**1-First Case:**

*assertEquals*(range.constrain(-5.0),-2.0,1.0E-13);  
*assertEquals*(range.constrain(10.0),9.0,1.0E-13);  
*assertEquals*(range.constrain(6.0),6.0,1.0E-13);  
*assertEquals*(range.constrain(-2.0),-2.0,1.0E-13); -> All Correct  
*assertEquals*(range.constrain(9.0),9.0,1.0E-13);  
*assertEquals*(range.constrain(8.99),8.99,1.0E-13);  
*assertEquals*(range.constrain(-1.99),-1.99,1.0E-13);

**3-Contains Method:**

**1-First Case:**

// Upper Input  
*assertTrue*(range.contains(-1.0)==true);  
*assertFalse*(range.contains(-3.0)==true); -> All Correct  
*assertEquals*(range.contains(-1.0),true);  
*assertNotEquals*(range.contains(-6.0),true);

**2-Second Case:**

//Lower Input  
*assertTrue*(range.contains(8.0)==true);  
*assertFalse*(range.contains(10.0)==true);  
*assertEquals*(range.contains(8.0),true); -> All Correct  
*assertNotEquals*(range.contains(10.0),true);

**3-Third Case:**

// same Range  
// failed, the function should return true when that the lower is -2.0 but in real was return false  
 assertTrue(range.contains(-2.0)==true);  
 assertFalse(range.contains(-2.0)==false);  
  
*assertEquals*(range.contains(9.0),true);  
*assertNotEquals*(range.contains(9.0),false); -> All Correct  
*assertEquals*(range.contains(0.0),true);

**In this Method all work correct except that contains not**

**Return true at the lower range.**

**4-Expand Method:**

**1-First Case:**

//The rule for Lowe Margins is number - length \* param1 ===> 2.0 - (8.0-2.0) \* 0.6 == -1.6  
//The rule for upper Margins is number + length \*param2 ==> 8.0 + (8.0-2.0) \* 5 == 38  
Range Example =new Range(2.0,8.0);   
Range Expand\_Range=Range.*expand*(Example,0.6,5);  
*assertEquals*(Expand\_Range.getUpperBound(),38.0,1.0E-13); -> Correct  
//failed ,the function should return 5.6 but the return -0.64, that mean in the function the lowerMargin is use wrong equation  
assertEquals(Expand\_Range.getLowerBound(),-0.64,1.0E-13); ->not Correct

**In this Method all work correct except that expand not**

**return the correct result at lower Range**

**2-Second Case:**

//test null  
Range expand\_Range= new Range(0.0,0.0);  
*assertThrows*(IllegalArgumentException.class,()->{expand\_Range.*expand*(null,0.0,0.0);}); ->Correct

**5-Expand to include Method:**

**1-First Case:**

Range expand\_Range= new Range(0.0,6.0);  
*assertEquals*(expand\_Range.*expandToInclude*( null,12.0).getUpperBound(),12.0,1.0E-13);  
*assertEquals*(Range.*expandToInclude*( null,0.0).getLowerBound(),0.0,1.0E-13);

Correct

**6-Get central value Method:**

**1-First Case:**

*assertEquals*(range.getCentralValue(),3.5,1.0E-13); ->Correct

**7-Get length Method:**

**1-First Case:**

*assertEquals*(range.getLength(),11.0,1.0E-13); -> Correct

**8- Lower Method:**

**1-First Case:**

*assertEquals*(range.getLowerBound(),-2.0,1.0E-13); ->Correct

**9- Upper Method:**

**1-First Case:**

*assertEquals*(range.getUpperBound(),9.0,1.0E-13); -> Correct

**10-Intersect Method:**

**1-First Case:**

//$offerStartTime < $endTime && $offerEndTime > $startTime  
 // param < 9.0 && param > -2.0  
 Range exp =new Range(-2.0,9.0);  
 *assertTrue*(exp.intersects(4.0,6.0));  
 *assertTrue*(exp.intersects(-4.0,4.0)); -> Correct all  
 *assertTrue*(exp.intersects(-6.0,-1.0));  
 *assertFalse*(exp.intersects(4.0,-3.0));  
 //failed ,the function should return false but the return true, that mean

the function is use wrong equation  
 assertTrue(exp.intersects(10.0,15.0)); -> Not Correct

**In this Method all work correct except that intersects not return false at range not include in hole range.**

**11-Shift\_1 Method:**

**1-First Case:**

Range shift\_range=Range.*shift*(range,5.0);  
*assertTrue*(shift\_range.getUpperBound()==14.0);  
*assertEquals*(shift\_range.getLowerBound(),0.0,1.0E-13);  
*assertThrows*(IllegalArgumentException.class,()->{Range.*shift*(null,0.0);});

**12-Shift\_2 Method:**

**1-First Case:**

Range shift\_range\_2=Range.*shift*(range,6.0,true);  
*assertTrue*(shift\_range\_2.getUpperBound()==15.0);  
*assertEquals*(shift\_range\_2.getLowerBound(),4.0,1.0E-13);  
*assertThrows*(IllegalArgumentException.class,()->{Range.*shift*(null,0.0);});

**13-To\_String Method:**

**1-First Case:**

final String expected = "Range[-2.0,9.0]";  
*assertEquals*(range.toString(),expected );