Programazio Modularra eta Objektuei Bideratutako Orientazioa- 3. laborategia

IZENA: ENEKO SAMPEDRO-ANDONI BERMO **DATA:**2015-02-24

Lortutako ezagutza eta gaitasunak

Behin 3. laborategia bukatu dudala, hurrengo atazak ondo egin ditudala pentsatzen dut., eta beraz etorkizunean errepikatu beharko banitu gai izango nintzateke. (markatu BAI ala EZ).

	Eneko Sampedro / Andoni Bermo	**BAI**	**EZ**
1	Eclipserekin lan egin Javaz proiektuak, paketeak eta klaseak errez sortzen	✓	
2	Ulertu UML diagrama sinple bat	✓	
3	Ondo inplementatu proba kasuak (*TestCase* klasea) eta proba kasuak batu proba multzoetan *(TestSuite)*:	✓	
4	Arauak betetzen dituzten izenak aukeratu ditut, bai klase, bai proiektu, pakete, atributu, metodo eta aldagaientzat	✓	
5	Enumeratu sinpleak ulertu eta erabili	✓	
6	Ondo ulertu Objektuei Bideratutako paradigma eta Prozedurei Bideratutako paradigma arteko desberdintasuna	✓	
7	Interfaze bat zer den ulertu eta gai izan interfaze bat inplementatzen duten klaseen garapena egiteko gai izan.	✓	

6	BESTELAKOAK: Azaldu		

Ezezko erantzunen bat eman baduzu mesedez jarraian azaldu zergatia, eta ze nolako neurria hartuko duzun sortu zaizun arazoa konpontzeko.

1.Atazaren Soluzioa

Data.java

```
package org.pmoo.packlaborategi3;

import java.util.Calendar;
import java.util.GregorianCalendar;

public class Data implements IData {

	//atributuak
	private int eguna;
	private int hilabetea;
	private int urtea;


	//eraikitzaileak
	public Data(int pUrtea, int pHilabetea, int pEguna) {

		this.eguna = pEguna;
		this.hilabetea = pHilabetea;
		this.urtea = pUrtea;

		if (!this.dataZuzenaDa()) {

			Calendar c = new GregorianCalendar();
			this.eguna = c.get(Calendar.DATE);
			this.hilabetea = c.get(Calendar.MONTH)+1;
			this.urtea = c.get(Calendar.YEAR);
		}
	}

	//beste metodoak

	@Override
	public String toString() {

		String strUrtea = String.format("%04d", this.urtea);
		String strHilabetea = String.format("%02d",
this.hilabetea);
		String strEguna = String.format("%02d", this.eguna);

		return strUrtea + "/" + strHilabetea + "/" + strEguna;
	}

	private boolean dataZuzenaDa() {
		if (this.hilabetea<13 && this.urtea>0) {
			if (this.eguna<=28) {
				return true;
			}
			else if(this.eguna==29 && (this.hilabetea!=2 ||
bisiestuaDa())) {
				return true;
			}
			else if(this.eguna==30 && this.hilabetea!=2) {
				return true;
			}
			else if(this.eguna==31 && (this.hilabetea==1 ||
this.hilabetea==3 || this.hilabetea==5 || this.hilabetea==7 ||
this.hilabetea==8 || this.hilabetea==10 || this.hilabetea==12 )) {
```

```
                        return true;
                  }
                  else {
                        return false;
                  }
            }
            else {
                  return false;
            }
      }

      private boolean bisiestuaDa(){
            if ( (this.urtea %4 ==0 && this.urtea % 100 != 0) ||
this.urtea % 400 == 0) {return true;}
            else{return false;}
      }

      @Override
      public void dataGehitu() {
            this.eguna++;
            if (this.eguna>29 && this.hilabetea==2 &&
this.bisiestuaDa()) {
                  this.eguna=1;
                  this.hilabetea++;
            }
            else if (this.eguna>28 && this.hilabetea==2 &&
!this.bisiestuaDa()) {
                  this.eguna=1;
                  this.hilabetea++;
            }
            else if (this.eguna>30 && (this.hilabetea==4 ||
this.hilabetea==6 || this.hilabetea==9 || this.hilabetea==11)) {
                  this.eguna=1;
                  this.hilabetea++;
            }

            else if (this.eguna>31) {
                  this.eguna=1;
                  this.hilabetea++;
                  if (this.hilabetea>12) {
                        this.hilabetea=1;
                        this.urtea++;
                  }
            }
      }

      @Override
      public void dataKendu() {
            this.eguna--;
            if (this.eguna==0) {
                  this.hilabetea--;
                  if (this.hilabetea==2 && !this.bisiestuaDa()) {
                        this.eguna=28;
                  }
                  else if(this.hilabetea==2 && this.bisiestuaDa()) {
                        this.eguna=29;
                  }
                  else if(this.hilabetea==4 || this.hilabetea==6 ||
this.hilabetea==9 || this.hilabetea==11) {
                        this.eguna=30;
                  }
```

```
                else {
                        this.eguna=31;
                        if (this.hilabetea==0) {
                                this.hilabetea=12;
                                this.urtea--;
                        }
                }
        }
    }
}
```

DataTest.java

```
package org.pmoo.packlaborategi3;

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class DataTest {

    private Data d1;
    private Data d2;

    @Before
    public void setUp() throws Exception {
        d1=new Data(2015,2,24);
        d2=new Data(2015,2,25);
    }

    @After
    public void tearDown() throws Exception {
        d1=null;
        d2=null;
    }


    @Test
    public void dataGehitu(){
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,2,25);
        d2=new Data(2015,2,26);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,2,28);
        d2=new Data(2015,3,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2016,2,28);
        d2=new Data(2016,2,29);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2016,2,29);
        d2=new Data(2016,3,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,1,30);
```

```
d2=new Data(2015,1,31);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,1,31);
d2=new Data(2015,2,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,3,30);
d2=new Data(2015,3,31);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,3,31);
d2=new Data(2015,4,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,4,29);
d2=new Data(2015,4,30);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,4,30);
d2=new Data(2015,5,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,5,30);
d2=new Data(2015,5,31);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,5,31);
d2=new Data(2015,6,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,6,29);
d2=new Data(2015,6,30);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,6,30);
d2=new Data(2015,7,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,7,30);
d2=new Data(2015,7,31);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,7,31);
d2=new Data(2015,8,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,8,30);
d2=new Data(2015,8,31);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,8,31);
d2=new Data(2015,9,1);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,9,29);
d2=new Data(2015,9,30);
d1.dataGehitu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,9,30);
d2=new Data(2015,10,1);
```

```
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,10,30);
        d2=new Data(2015,10,31);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,10,31);
        d2=new Data(2015,11,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,11,29);
        d2=new Data(2015,11,30);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,11,30);
        d2=new Data(2015,12,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,12,30);
        d2=new Data(2015,12,31);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,12,31);
        d2=new Data(2016,1,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(1,12,31);
        d2=new Data(2,1,1);
        d1.dataGehitu();
        assertEquals(d1.toString(),d2.toString());
}


@Test
public void dataKendu(){
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,2,25);
        d2=new Data(2015,2,26);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,2,28);
        d2=new Data(2015,3,1);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2016,2,28);
        d2=new Data(2016,2,29);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2016,2,29);
        d2=new Data(2016,3,1);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,1,30);
        d2=new Data(2015,1,31);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
        d1=new Data(2015,1,31);
        d2=new Data(2015,2,1);
        d2.dataKendu();
        assertEquals(d1.toString(),d2.toString());
```

```
d1=new Data(2015,3,30);
d2=new Data(2015,3,31);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,3,31);
d2=new Data(2015,4,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,4,29);
d2=new Data(2015,4,30);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,4,30);
d2=new Data(2015,5,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,5,30);
d2=new Data(2015,5,31);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,5,31);
d2=new Data(2015,6,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,6,29);
d2=new Data(2015,6,30);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,6,30);
d2=new Data(2015,7,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,7,30);
d2=new Data(2015,7,31);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,7,31);
d2=new Data(2015,8,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,8,30);
d2=new Data(2015,8,31);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,8,31);
d2=new Data(2015,9,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,9,29);
d2=new Data(2015,9,30);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,9,30);
d2=new Data(2015,10,1);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,10,30);
d2=new Data(2015,10,31);
d2.dataKendu();
assertEquals(d1.toString(),d2.toString());
d1=new Data(2015,10,31);
```

```
            d2=new Data(2015,11,1);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
            d1=new Data(2015,11,29);
            d2=new Data(2015,11,30);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
            d1=new Data(2015,11,30);
            d2=new Data(2015,12,1);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
            d1=new Data(2015,12,30);
            d2=new Data(2015,12,31);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
            d1=new Data(2015,12,31);
            d2=new Data(2016,1,1);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
            d1=new Data(1,12,31);
            d2=new Data(2,1,1);
            d2.dataKendu();
            assertEquals(d1.toString(),d2.toString());
        }

}
```

2.Atazaren Soluzioa

Zatikia.java

```
package org.pmoo.packlaborategi3;

public class Zatikia implements IZatikia{

	private int zenbakitzailea;
	private int izendatzailea;

	public Zatikia(int pZenb, int pIzen) {
		this.zenbakitzailea = pZenb;
		this.izendatzailea = pIzen;

		if (this.izendatzailea == 0) {
			System.out.println("Ezin daiteke izendatzailearen
balioa 0 izan dezakeen zatikirik sortu...");
		}
	}

	public int getZenbakitzailea() {
		return this.zenbakitzailea;
	}

	public int getIzendatzailea() {
		return this.izendatzailea;
	}

	private int ikt() {
		int aux1 = Math.abs(this.zenbakitzailea);
		int aux2 = Math.abs(this.izendatzailea);

		if(aux2 == 0) {
			return aux1;
		}

		int aux3;

		while(aux2 != 0) {
			aux3 = aux1 % aux2;
			aux1 = aux2;
			aux2 = aux3;
		}

		return aux1;
	}

	public void sinplifikatu() {
		int zatitu = ikt();

		this.zenbakitzailea /= zatitu;
		this.izendatzailea /= zatitu;
	}

	public Zatikia gehitu(Zatikia pZatikia) {
		int emaitzaZenbakitzailea = (this.zenbakitzailea *
pZatikia.izendatzailea)+(this.izendatzailea *
pZatikia.zenbakitzailea);
		int emaitzaIzendatzailea = (this.izendatzailea *
pZatikia.izendatzailea);
```

```
            Zatikia emaitza = new Zatikia(emaitzaZenbakitzailea,
emaitzaIzendatzailea);

            emaitza.sinplifikatu();

            return emaitza;
      }

      public Zatikia kendu(Zatikia pZatikia) {
            int emaitzaZenbakitzailea = (this.zenbakitzailea *
pZatikia.izendatzailea)-(this.izendatzailea *
pZatikia.zenbakitzailea);
            int emaitzaIzendatzailea = (this.izendatzailea *
pZatikia.izendatzailea);
            Zatikia emaitza = new Zatikia(emaitzaZenbakitzailea,
emaitzaIzendatzailea);

            emaitza.sinplifikatu();

            return emaitza;
      }

      public Zatikia biderkatu(Zatikia pZatikia) {
            int emaitzaZenbakitzailea = (this.zenbakitzailea *
pZatikia.zenbakitzailea);
            int emaitzaIzendatzailea = (this.izendatzailea *
pZatikia.izendatzailea);
            Zatikia emaitza = new Zatikia(emaitzaZenbakitzailea,
emaitzaIzendatzailea);

            emaitza.sinplifikatu();

            return emaitza;
      }

      public Zatikia zatitu(Zatikia pZatikia) {
            int emaitzaZenbakitzailea = (this.zenbakitzailea *
pZatikia.izendatzailea);
            int emaitzaIzendatzailea = (this.izendatzailea *
pZatikia.zenbakitzailea);
            Zatikia emaitza = new Zatikia(emaitzaZenbakitzailea,
emaitzaIzendatzailea);

            emaitza.sinplifikatu();

            return emaitza;
      }

      public boolean berdinaDa(Zatikia pZatikia){
            float bat = (float)this.zenbakitzailea /
(float)this.izendatzailea;
            float bi = (float)pZatikia.zenbakitzailea /
(float)pZatikia.izendatzailea;

            if (bat == bi) {return true;}
            else {return false;}

      }

      public boolean handiagoaDa(Zatikia pZatikia) {
```

```
            float bat = (float)this.zenbakitzailea /
(float)this.izendatzailea;
            float bi = (float)pZatikia.zenbakitzailea /
(float)pZatikia.izendatzailea;

            if (bat > bi) {return true;}
            else {return false;}
      }

      public boolean txikiagoaDa(Zatikia pZatikia) {
            float bat = (float)this.zenbakitzailea /
(float)this.izendatzailea;
            float bi = (float)pZatikia.zenbakitzailea /
(float)pZatikia.izendatzailea;

            if (bat < bi) {return true;}
            else {return false;}
      }

}
```

ZatikiaTest.java

```
package org.pmoo.packlaborategi3;

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class ZatikiaTest {

      private Zatikia zatiki1;
      private Zatikia zatiki2;
      private Zatikia zatiki3;



      @Before
      public void setUp() throws Exception {
            zatiki1 = new Zatikia(10,10);
            zatiki2 = new Zatikia(10,10);
            zatiki3 = new Zatikia(0, 0);
      }

      @After
      public void tearDown() throws Exception {
            zatiki1 = null;
      }

      @Test
      public void testGetZenbakitzailea() {
            assertSame(10 , zatiki1.getZenbakitzailea());
            assertNotSame(9 , zatiki1.getZenbakitzailea());
      }

      @Test
      public void testGetIzendatzailea() {
            assertSame(10 , zatiki1.getIzendatzailea());
            assertNotSame(9 , zatiki1.getIzendatzailea());
      }
```

```
@Test
public void testSinplifikatu() {
	zatiki1.sinplifikatu();
	assertSame(1 , zatiki1.getZenbakitzailea());
	assertNotSame(10 , zatiki1.getZenbakitzailea());
	assertSame(1 , zatiki1.getIzendatzailea());
	assertNotSame(10 , zatiki1.getIzendatzailea());

	zatiki1 = new Zatikia(2,3);
	zatiki1.sinplifikatu();
	assertSame(2 , zatiki1.getZenbakitzailea());
	assertNotSame(10 , zatiki1.getZenbakitzailea());
	assertSame(3 , zatiki1.getIzendatzailea());
	assertNotSame(10 , zatiki1.getIzendatzailea());

	zatiki1 = new Zatikia(3,2);
	zatiki1.sinplifikatu();
	assertSame(3 , zatiki1.getZenbakitzailea());
	assertNotSame(10 , zatiki1.getZenbakitzailea());
	assertSame(2 , zatiki1.getIzendatzailea());
	assertNotSame(10 , zatiki1.getIzendatzailea());

	zatiki1 = new Zatikia(1,0);
	zatiki1.sinplifikatu();
	assertSame(1 , zatiki1.getZenbakitzailea());
	assertNotSame(10 , zatiki1.getZenbakitzailea());
	assertSame(0 , zatiki1.getIzendatzailea());
	assertNotSame(10 , zatiki1.getIzendatzailea());

	zatiki1 = new Zatikia(-6,4);
	zatiki1.sinplifikatu();
	assertSame(-3 , zatiki1.getZenbakitzailea());
	assertNotSame(10 , zatiki1.getZenbakitzailea());
	assertSame(2 , zatiki1.getIzendatzailea());
	assertNotSame(10 , zatiki1.getIzendatzailea());
}

@Test
public void testGehitu() {
	zatiki3 = zatiki1.gehitu(zatiki2);
	assertSame(zatiki3.getZenbakitzailea() , 2);
	assertNotSame(zatiki3.getZenbakitzailea() , 20);
	assertSame(zatiki3.getIzendatzailea() , 1);
	assertNotSame(zatiki3.getIzendatzailea() , 20);

	zatiki1 = new Zatikia(1,2);
	zatiki2 = new Zatikia(1,3);
	zatiki3 = zatiki1.gehitu(zatiki2);
	assertSame(zatiki3.getZenbakitzailea() , 5);
	assertNotSame(zatiki3.getZenbakitzailea() , 20);
	assertSame(zatiki3.getIzendatzailea() , 6);
	assertNotSame(zatiki3.getIzendatzailea() , 20);

	zatiki1 = new Zatikia(1,3);
	zatiki2 = new Zatikia(1,2);
	zatiki3 = zatiki1.gehitu(zatiki2);
	assertSame(zatiki3.getZenbakitzailea() , 5);
	assertNotSame(zatiki3.getZenbakitzailea() , 20);
	assertSame(zatiki3.getIzendatzailea() , 6);
	assertNotSame(zatiki3.getIzendatzailea() , 20);
```

```
        zatiki1 = new Zatikia(2,-3);
        zatiki2 = new Zatikia(2,6);
        zatiki3 = zatiki1.gehitu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 1);
        assertNotSame(zatiki3.getZenbakitzailea() , 20);
        assertSame(zatiki3.getIzendatzailea() , -3);
        assertNotSame(zatiki3.getIzendatzailea() , 20);
    }

    @Test
    public void testKendu() {
        zatiki3 = zatiki1.kendu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 0);
        assertNotSame(zatiki3.getZenbakitzailea() , 20);
        assertSame(zatiki3.getIzendatzailea() , 1);
        assertNotSame(zatiki3.getIzendatzailea() , 20);
    }

    @Test
    public void testBiderkatu() {
        zatiki3 = zatiki1.biderkatu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 1);
        assertNotSame(zatiki3.getZenbakitzailea() , 100);
        assertSame(zatiki3.getIzendatzailea() , 1);
        assertNotSame(zatiki3.getIzendatzailea() , 100);

        zatiki2 = new Zatikia(1,2);
        zatiki3 = zatiki1.biderkatu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 1);
        assertNotSame(zatiki3.getZenbakitzailea() , 100);
        assertSame(zatiki3.getIzendatzailea() , 2);
        assertNotSame(zatiki3.getIzendatzailea() , 100);

        zatiki2 = new Zatikia(1,-2);
        zatiki3 = zatiki1.biderkatu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 1);
        assertNotSame(zatiki3.getZenbakitzailea() , 100);
        assertSame(zatiki3.getIzendatzailea() , -2);
        assertNotSame(zatiki3.getIzendatzailea() , 100);

    }

    @Test
    public void testZatitu() {
        zatiki3 = zatiki1.zatitu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 1);
        assertNotSame(zatiki3.getZenbakitzailea() , 100);
        assertSame(zatiki3.getIzendatzailea() , 1);
        assertNotSame(zatiki3.getIzendatzailea() , 100);

        zatiki2 = new Zatikia(1,2);
        zatiki3 = zatiki1.zatitu(zatiki2);
        assertSame(zatiki3.getZenbakitzailea() , 2);
        assertNotSame(zatiki3.getZenbakitzailea() , 100);
        assertSame(zatiki3.getIzendatzailea() , 1);
        assertNotSame(zatiki3.getIzendatzailea() , 100);

        zatiki1 = new Zatikia(-1, 2);
        zatiki2 = new Zatikia(6,-3);
        zatiki3 = zatiki1.zatitu(zatiki2);
```

```
            assertSame(zatiki3.getZenbakitzailea() , 1);
            assertNotSame(zatiki3.getZenbakitzailea() , 100);
            assertSame(zatiki3.getIzendatzailea() , 4);
            assertNotSame(zatiki3.getIzendatzailea() , 100);
      }

      @Test
      public void testBerdinaDa() {
            assertTrue(zatiki1.berdinaDa(zatiki2));
            zatiki2 = new Zatikia(1,2);
            assertFalse(zatiki1.berdinaDa(zatiki2));
      }

      @Test
      public void testHandiagoaDa() {
            zatiki2 = new Zatikia(1,2);
            assertTrue(zatiki1.handiagoaDa(zatiki2));
            zatiki2 = new Zatikia(11,10);
            assertFalse(zatiki1.handiagoaDa(zatiki2));
      }

      @Test
      public void testTxikiagoaDa() {
            zatiki2 = new Zatikia(11,10);
            assertTrue(zatiki1.txikiagoaDa(zatiki2));
            zatiki2 = new Zatikia(1,2);
            assertFalse(zatiki1.txikiagoaDa(zatiki2));
      }

}
```