Nota aclaratoria:
Estás realizando un examen, y como tal no puedes manejar ningún material de referencia adicional, ni comunicarte de ninguna forma con nadie. Estás sólo tú con estas instruciones y las herramientas imprescindibles para llevarlas a cabo (Eclipse y Writer). Si se detecta cualquier amago de comunicación, automáticamente suspenderás.

Utilizando el siguiente código que implementa una Pila, sigue las siguientes instrucciones. Al final deberás entregar un fichero comprimido "NombreApellido1Apellido2ExJUnit.rar" con dos ficheros:

1. NombreApellido1Apellido2ExJUnit.pdf (Exportado a pdf desde OpenWriter)
2. NombreApellido1Apellido2ExJUnitProyecto. (Eclipse)

Vamos a diseñar un caso de pruebas para la clase Pila. Para ello trabajaremos sobre una pila de Personas (también se da en el código). Realiza los siguientes pasos:

- Abre Eclipse y crea un proyecto "NombreApellido1Apellido2ExJUnit.
- Añade el paquete pila que contiene Pila.java y TestPila.java.
- En el mismo paquete crea un caso de prueba jUnitPila sobre la clase Pila.
- Para todas las pruebas se van a necesitar tres objetos de la clase Persona. Serán los que apilemos/desapilemos de la Pila. Crea los tres objetos Persona con las siguientes características. Los referenciaremos en el orden indicado y con el nombre indicado.
  - Anacleto Montes, Rigoberta Bosques, Florencio Arenas
- A partir de ahora, y a lo largo de todo el examen, sustituye el nombre de Anacleto Montes por tu nombre.
- Además de los tres objetos necesitaremos un objeto Pila de Personas para cada una de las pruebas. Llámala como tu usuario "**d13apapn**". Cada prueba comenzará con la pila vacía.
- Cada prueba comenzará con una Pila de Personas vacía.
- Debes probar los siguientes métodos:
  - Pila: constructor
    1. Crea una nueva pila
    2. Comprueba que está vacía, lo que devuelve top y su tamaño
  - top: Devuelve la cima de la pila.
    1. Comprueba que top() inicialmente devuelve null
    2. Apila a Anacleto, Rigoberta y a Florencio.
    3. Comprueba que top() devuelve a Florencio
    4. Comprueba que top() no devuelve null
    5. Apila un null
    6. Comprueba que top() devuelve null
  - pop: Desapila
    1. Apila a Anacleto
    2. Comprueba que pop() devuelve a Anacleto
    3. Apila a Rigoberta y a Florencio.
    4. Comprueba que pop() devuelve a Florencio
    5. Comprueba que pop() devuelve a Rigoberta
    6. Comprueba que pop() devuelve null
  - push: apila

1. Comprueba que top() inicialmente devuelve null
2. Apila null
3. Comprueba que top() devuelve null y el tamaño de la pila
4. Apila a Anacleto
5. Comprueba que top() lo devuelve y el tamaño de la pila
6. Apila a Rigoberta
7. Comprueba que top() lo devuelve y el tamaño de la pila

- isEmpty: comprueba si está vacía
  1. Comprueba que isEmpty() devuelve true
  2. Apila a Anacleto
  3. Comprueba que isEmpty() devuelve false
  4. Desapila
  5. Comprueba que isEmpty() devuelve true

Entrega el proyecto, así como los siguientes pantallazos:

1. Código de todo el proceso.
2. Cada uno de los métodos implementados
3. La vista con los tiempos en cada uno de los test. Todos han de estar chequeados correctamente, inlucida la clase.

He implementado para poder realizar los test del Junit.

- import static org.junit.Assert

- import org.junit.Before

- import org.junit.Test;

Metodos:

PilaTest.java

```
1  package pila;
2
3  import static org.junit.Assert.*;
7
8  public class PilaTest {
9      @Before
10     public void pila(){
11         Pila d14pegod;
12     }
13
14     @Test
15     public void testConstructor() {
16         Pila pilaNueva=new Pila<Persona>();
17         assertEquals(pilaNueva.size(),0);
18     }
19     @Test
20     public void testTop() {
21         Persona David =new Persona("David", "Peralvo");
22         Persona Rigoberta=new Persona("Rigoberta","Bosques");
23         Persona Florencio=new Persona("Florencio","Flores");
24         Pila d14pegod=new Pila<Persona>();
25         assertEquals(d14pegod.top(),null);
26         d14pegod.push(David);
27         d14pegod.push(Rigoberta);
28         d14pegod.push(Florencio);
29         Persona top=(Persona)d14pegod.top();
30         assertEquals(d14pegod.top(),Florencio);
31         d14pegod.push(null);
32         assertEquals(d14pegod.top(),null);
33
34     }
35     @Test
36     public void testPop() {
37         Persona David =new Persona("David", "Peralvo");
38         Persona Rigoberta=new Persona("Rigoberta","Bosques");
39         Persona Florencio=new Persona("Florencio","Flores");
40         Pila d14pegod=new Pila<Persona>();
41         d14pegod.push(David);
42         assertEquals(d14pegod.pop(),David);
43         d14pegod.push(Rigoberta);
44         d14pegod.push(Florencio);
45         assertEquals(d14pegod.pop(),Florencio);
46         assertEquals(d14pegod.pop(),Rigoberta);
47         assertEquals(d14pegod.pop(),null);
48
```

```
    }
    @Test
    public void TestPush(){
        Persona David =new Persona("David", "Peralvo");
        Persona Rigoberta=new Persona("Rigoberta","Bosques");
        Pila d14pegod=new Pila<Persona>();
        assertEquals(d14pegod.top(),null);
        d14pegod.push(null);
        assertEquals(d14pegod.top(),null);
        assertEquals(d14pegod.size(),1);
        d14pegod.push(David);
        assertEquals(d14pegod.top(),David);
        assertEquals(d14pegod.size(),2);
        d14pegod.push(Rigoberta);
        assertEquals(d14pegod.top(),Rigoberta);
        assertEquals(d14pegod.size(),3);

    }
    @Test
    public void TestisEmpy(){
        Persona David =new Persona("David", "Peralvo");
        Pila d14pegod=new Pila<Persona>();
        assertEquals(d14pegod.IsEmpty(),true);
        d14pegod.push(David);
        assertEquals(d14pegod.IsEmpty(),false);
        d14pegod.pop();
        assertEquals(d14pegod.IsEmpty(),true);



    }

}
```

Tiempos del test: