Trabajo Clean Code 4-5-6-7 Sergio Sanchez Ramirez
url github :https://github.com/sergiosanram1/Practica-clean-code-4-5-6-7.git

Apartado 4

```java
 */
public class CleanCode4567 {

    public static void main(String[] args) {
        ArrayList<Persona> arrayPersonas=new ArrayList<>();
        Persona pepe = new Persona("12345678Y", "Pepe", "Perez", 21);
        Persona juan = new Persona("84723930T", "Juan", "Fernandez", 12);
        Persona luis = new Persona("01927461A", "Luis", "Lopez", 18);
        arrayPersonas.add(pepe);
        arrayPersonas.add(juan);
        arrayPersonas.add(luis);
        for (Persona i:arrayPersonas){
            if (i.verificarMayoriaEdad()){
                System.out.println(i.getNombre() + " es mayor de edad");
            }else{
                System.out.println(i.getNombre() + " no es mayor de edad");
            }
        }
    }
}
```

```java
public class Persona {
    String dni;
    String nombre;
    String apellido;
    int edad;

    public Persona(String dni, String nombre, String apellido, int edad) {
        this.dni = dni;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
    }

    public Persona() {
        dni ="";
        nombre="";
        apellido="";
        edad=0;
    }
    public boolean verificarMayoriaEdad() {
        return edad >= 18;
    }
```

Cumple el apartado 4 ya que los objetos tiene funciones logicas y cumple la ley de demeter ya que solo un objeto de la calse o derivado de la clase accede a sus datos y la estructura de datos que en este ejemplo es un arrayList no tiene funciones logicas

Apartado 5

```
try{
System.out.println("Introduce DNI de la primera persona");
    dni = teclado.nextLine();
System.out.println("Introduce nombre de la primera persona");
    nombre = teclado.nextLine();
System.out.println("Introduce apellido de la primera persona");
    apellido = teclado.nextLine();
System.out.println("Introduce edad de la primera persona");
    edad = teclado.nextInt();
Persona pepe = new Persona(dni, nombre, apellido, edad);

System.out.println("Introduce DNI de la segunda persona");
    dni = teclado.nextLine();
System.out.println("Introduce nombre de la segunda persona");
    nombre = teclado.nextLine();
System.out.println("Introduce apellido de la segunda persona");
    apellido = teclado.nextLine();
System.out.println("Introduce edad de la segunda persona");
    edad = teclado.nextInt();
Persona juan = new Persona(dni, nombre, apellido, edad);

System.out.println("Introduce DNI de la tercera persona");
    dni = teclado.nextLine();
System.out.println("Introduce nombre de la tercera persona");
    nombre = teclado.nextLine();
```

```
    arrayPersonas.add(pepe);
    arrayPersonas.add(juan);
    arrayPersonas.add(luis);
    for (Persona i:arrayPersonas){
        if (i.verificarMayoriaEdad()){
            System.out.println(i.getNombre() + " es mayor de edad");
        }else{
            System.out.println(i.getNombre() + " no es mayor de edad");
        }
    }
    } catch (InputMismatchException ex) {
        System.out.println("Has introducido algun dato incorrecto");
    }

}
```

Usamos una excepción para comprobar que si al insertar datos por teclado salta un error de metida de datos saque la frase en vez del error.
Por lo que no devuelve nunca null

Apartado 6

```java
public class PersonaTest {

    Persona pruebamayor;
    Persona pruebamenor;
    Persona pruebacero;
    Persona pruebanegativo;


    @BeforeEach
    public void setUp() {
        pruebamayor = new Persona("11111111A", "Prueba1", "Prueba1", 19);
        pruebamenor = new Persona("11111112A", "Prueba2", "Prueba2", 9);
        pruebacero = new Persona("11111113A", "Prueba3", "Prueba3", 0);
        pruebanegativo = new Persona("11111114A", "Prueba4", "Prueba4", -4);
    }

    @AfterEach
    public void tearDown() {
        pruebamayor = new Persona();
        pruebamenor = new Persona();
        pruebacero = new Persona();
        pruebanegativo = new Persona();
    }

    @Test
    public void testVerificarMayoriaEdad() {
        assertAll(() -> {
            assertTrue(pruebamayor.verificarMayoriaEdad());
            assertFalse(pruebamenor.verificarMayoriaEdad());
            assertFalse(pruebacero.verificarMayoriaEdad());
            assertFalse(pruebanegativo.verificarMayoriaEdad());
        });
    }
```

Hacemos test para el metodo de verificarMAyoriaEdad cumpliendo las reglas F.I.R.S.T, solo hay un assert y solo tiene un concepto

Apartado 7

```java
*/
public class Persona {
    String dni;
    String nombre;
    String apellido;
    int edad;

    public Persona(String dni, String nombre, String apellido, int edad) {
        this.dni = dni;
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
    }

    public Persona() {
        dni ="";
        nombre="";
        apellido="";
        edad=0;
    }
    public boolean verificarMayoriaEdad() {
        return edad >= 18;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```java
public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}

@Override
public String toString() {
    return "Persona{" + "dni=" + dni + ", nombre=" + nombre + ", apellido=" + apellido + ", edad=" + edad + '}';
}
```

La clase esta organizada, la clase se centraq en solo un objetivo que es almacenar personas y comprobar la mayoria de edad, tiene pocas variables