

CONCEPTOS RELACIONADOS CON LA POO **PRÁCTICA**



CONCEPTOS RELACIONADOS CON LA POO - PRÁCTICA

1.CREACIÓN Y USO DE UNA CLASE GENÉRICA

Crear una clase genérica, en un paquete **claseGenerica**, llamada **Operaciones** con cuatro métodos para las cuatro operaciones básicas: suma, resta, producto y división.

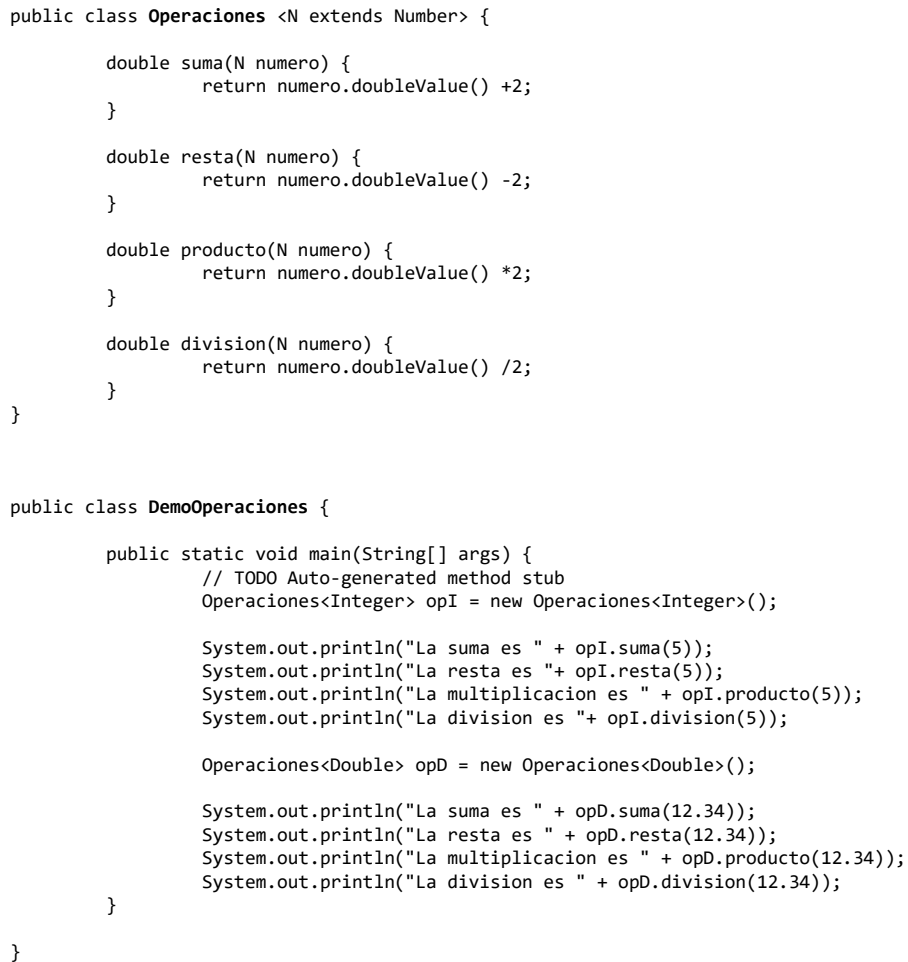
Especificaciones:

- ✓ Se escribe la clase Operaciones con un tipo **genérico N**.
- ✓ Todos los **métodos** de esta clase **reciben un objeto de tipo N y devuelven un double**.
- ✓ Cada función con cada una de las cuatro operaciones, lo que tiene que hacer es incrementar en dos, decrementar en dos, multiplicar por dos o dividir por dos, respectivamente, el argumento pasado al método y retornar el resultado de dicha operación.

Posteriormente, para probar la clase genérica Operaciones, debemos crear una clase llamada **DemoOperaciones**, en la cual debemos crear:

- Una instancia de Operaciones de **tipo Integer** y realizar las cuatro operaciones y mostrar el resultado. El valor entero con el que vamos a probar es 5.
- Una segunda instancia de Operaciones de **tipo Double** y realizar las cuatro operaciones y mostrar el resultado. El valor con el que vamos a probar es 12.34.

Solución:



```
public class Operaciones <N extends Number> {

    double suma(N numero) {
        return numero.doubleValue() +2;
    }

    double resta(N numero) {
        return numero.doubleValue() -2;
    }

    double producto(N numero) {
        return numero.doubleValue() *2;
    }

    double division(N numero) {
        return numero.doubleValue() /2;
    }
}

public class DemoOperaciones {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Operaciones<Integer> opI = new Operaciones<Integer>();

        System.out.println("La suma es " + opI.suma(5));
        System.out.println("La resta es " + opI.resta(5));
        System.out.println("La multiplicacion es " + opI.producto(5));
        System.out.println("La division es "+ opI.division(5));

        Operaciones<Double> opD = new Operaciones<Double>();

        System.out.println("La suma es " + opD.suma(12.34));
        System.out.println("La resta es " + opD.resta(12.34));
        System.out.println("La multiplicacion es " + opD.producto(12.34));
        System.out.println("La division es " + opD.division(12.34));
    }
}
```

CONCEPTOS RELACIONADOS CON LA POO - PRÁCTICA

2.USO DE INTERFAZ DE COLECCIÓN Y ORDENACIÓN

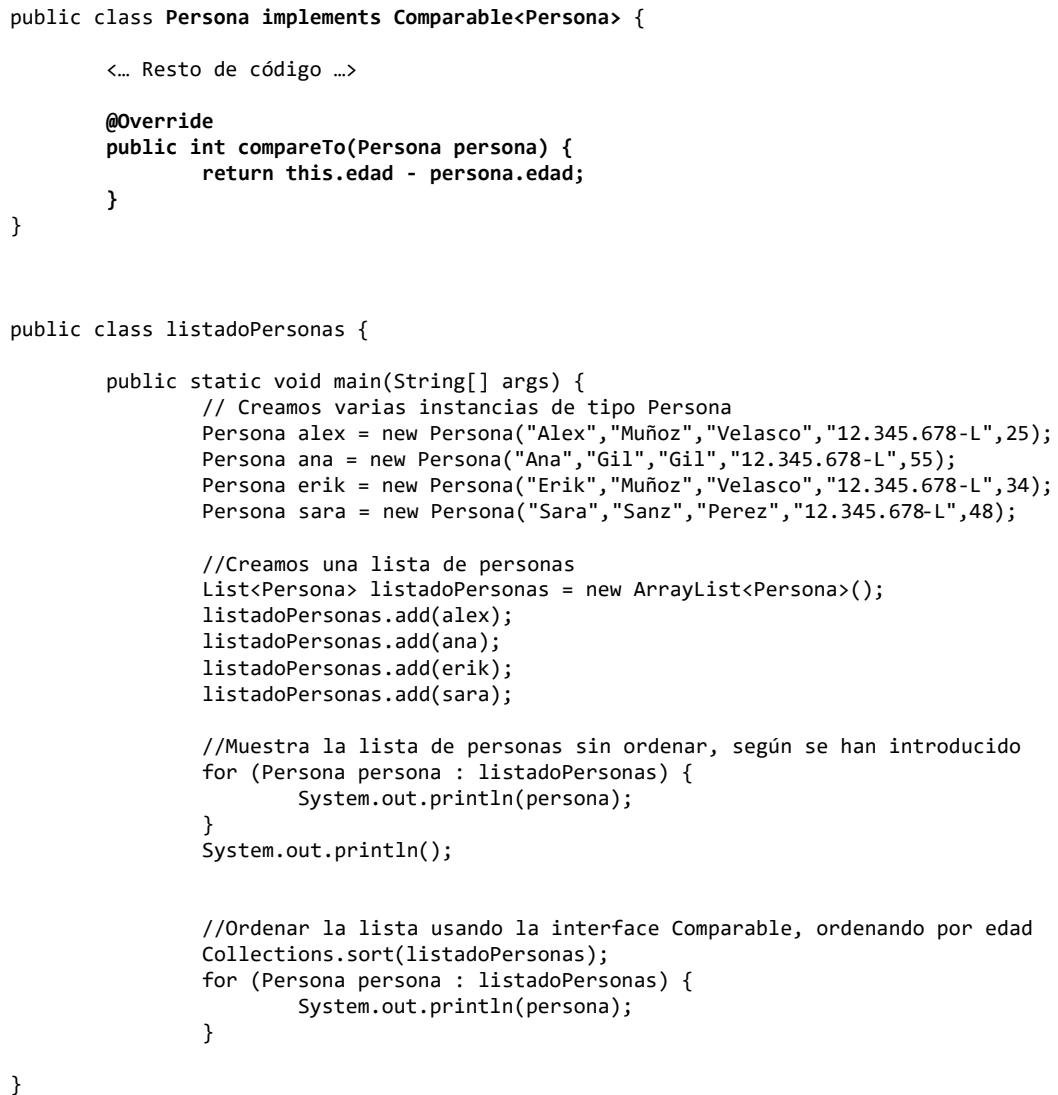
Para este ejercicio, vamos a reutilizar la clase **Persona** creada anteriormente.

En primer lugar, vamos a definir el criterio de orden natural (en este caso por la edad de menor a mayor) para la clase Persona, para ello vamos a modificar la clase Persona para que implemente la **interfaz Comparable**, para lo que tendremos que sobrescribir el método **compareTo** e indicar la ordenación por edad, ya que es uno de los atributos de los que tenemos en la clase Persona.

Por último, debemos crear una clase llamada **listadoPersonas**, en el mismo paquete, en la cual debemos crear el método main con lo siguiente:

- Varias instancias de tipo Persona usando el constructor de parámetros y asignando distintos valores para cada instancia.
- Los valores (nombre, primer apellido, segundo apellido, dni, edad) con los que vamos a probar son los siguientes:
 - "Alex", "Muñoz", "Velasco", "12.345.678-L", 25
 - "Ana", "Gil", "Gil", "12.345.678-L", 55
 - "Erik", "Muñoz", "Velasco", "12.345.678-L", 34
 - "Sara", "Sanz", "Pérez", "12.345.678-L", 48
- Creamos una lista de tipo Persona con todas las instancias creadas anteriormente.
- Mostrar por consola la lista de personas de la siguientes formas:
 - Sin ordenar, es decir, según se han introducido.
 - Ordenar la lista usando la interface Comparable, es decir, ordenando por edad y mostrando dicha lista tras su ordenación.

Solución:



```
public class Persona implements Comparable<Persona> {  
  
    <... Resto de código ...>  
  
    @Override  
    public int compareTo(Persona persona) {  
        return this.edad - persona.edad;  
    }  
}  
  
public class listadoPersonas {  
  
    public static void main(String[] args) {  
        // Creamos varias instancias de tipo Persona  
        Persona alex = new Persona("Alex", "Muñoz", "Velasco", "12.345.678-L", 25);  
        Persona ana = new Persona("Ana", "Gil", "Gil", "12.345.678-L", 55);  
        Persona erik = new Persona("Erik", "Muñoz", "Velasco", "12.345.678-L", 34);  
        Persona sara = new Persona("Sara", "Sanz", "Perez", "12.345.678-L", 48);  
  
        //Creamos una lista de personas  
        List<Persona> listadoPersonas = new ArrayList<Persona>();  
        listadoPersonas.add(alex);  
        listadoPersonas.add(ana);  
        listadoPersonas.add(erik);  
        listadoPersonas.add(sara);  
  
        //Muestra la lista de personas sin ordenar, según se han introducido  
        for (Persona persona : listadoPersonas) {  
            System.out.println(persona);  
        }  
        System.out.println();  
  
        //Ordenar la lista usando la interface Comparable, ordenando por edad  
        Collections.sort(listadoPersonas);  
        for (Persona persona : listadoPersonas) {  
            System.out.println(persona);  
        }  
    }  
}
```

**FUNDACIÓN
ACCENTURE**

accenture