

```

public class Agencia {
    private Lista<Contribuyente> contrBuyentes;

    public Agencia() {
        this.contrBuyentes = new ArrayList<>();
    }

    public Contribuyente altaContribuyente(String nombre, String dni, String email, String localidad) {
        Contribuyente c = new Contribuyente(nombre, dni, email, localidad);
        contrBuyentes.add(c);
        return c;
    }

    public Inmueble altaInmueble(String numPartida, double ValorDeLote, double ValorEdificacion, Contribuyente propietario) {
        Inmueble i = new Inmueble(numPartida, ValorDeLote, ValorEdificacion);
        propietario.registrarBien(i);
        return i;
    }

    public Automotor altaAutomotor(String patente, String Marca, LocalDate fechaFabricacion, double Valor, Contribuyente Propietario) {
        Automotor a = new Automotor(patente, Marca, fechaFabricacion, Valor);
        propietario.registrarBien(a);
        return a;
    }
}

```

```
public Embarcacion altaEmbarcacion(string patente, string nombre, Localidad  
fechaFabricacion, double valor, Contribuyente propietario) {  
    Embarcacion e = new Embarcacion(patente, nombre, fechaFabricacion, valor);  
    Propietario.registerBien(e);  
    return e;  
}
```

```
public double calcularImpuesto(Contribuyente c) {  
    return c.calcularImpuesto();  
} * .filter(c -> c.getLocalidad().equals(localidad)).
```

```
public List<Contribuyente> nMasPagan(string localidad, int N) {  
    return this.contribuyentes.stream().sorted((Contribuyente c1, Contribuyente c2) -> Double.compare(c1.calcularImpuesto(), c2.calcularImpuesto())).  
        limit(N).toList();  
}
```

```
public class Contribuyente {  
    private String nombre; private List<Bien> bienes;  
    private String DNI;  
    private String email;  
    private String localidad;  
    public Contribuyente(String n, String dni, String e, String l) {  
        this.bienes = new ArrayList();  
        this.nombre = n; this.DNI = dni; this.email = e; this.localidad = l;  
    }
```

```
public String getLocalidad() { return this.localidad; }  
public double calcularImpuestos() {  
    return this.bienes.stream().mapToDouble(b -> b.calcularImpuestos()).sum();  
}
```

```
public void registrarBien(Bien b){
```

```
    this.bienes.add(b);
```

```
}
```

```
}
```

```
public interface Bien{
```

```
    public abstract double calcularImpuestos();
```

```
}
```

```
public abstract class Vehiculo implements Bien{
```

```
    private String patente; //protected LocalDate fechaFabricacion;
```

```
    protected double valor; //private
```

```
    protected public Vehiculo(String patente, LocalDate fechaFabricacion, double valor){
```

```
        this.patente = patente;
```

```
        this.fechaFabricacion = fechaFabricacion;
```

```
        this.valor = valor;
```

```
}
```

```
    public int calcularAntiguedad(){
```

```
        return (int) ChronoUnit.YEARS.between(this.fechaFabricacion,  
        LocalDate.now());
```

```
}
```

```
}
```

```
public class Inmueble implements Bien{
```

```
    private String numeroDePartida;
```

```
    private double valorDelLote;
```

```
private double ValorEdificacion;
public Immueble (string numeroDePartida, double ValorDelLote,
double ValorEdificacion) {
    this.numeroDePartida = numeroDePartida;
    this.ValorDelLote = ValorDelLote;
    this.ValorEdificacion = ValorEdificacion;
}
```

{

```
public double calcularImpuestos () {
    return (this.ValorLote + this.ValorEdificacion) * 0,01;
}
```

}

{

```
public class Embarcacion extends Vehiculo {
```

```
private string nombre;
```

```
public Embarcacion (string patente, string nombre, string
Localitate FechaFabricacion, double Valor) {
    super (patente, FechaFabricacion, Valor);
    this.nombre = nombre;
}
```

}

```
public double calcularImpuestos () {
```

```
if (this.calcularAntiguedad > 10) {
```

```
    return 0;
}
```

```
else {
```

```
    if (this.Valor < 1.000.000) {
```

```
        return this.Valor * 0,1;
    }
}
```

```
return this.Valor * 0,15;
}
```

}

3  
3  
3

```
public class Automotor extends Vehiculo {
```

```
    private String marca;
```

```
    private String modelo;
```

```
    public Automotor (String patente, String marca, String modelo) {
```

```
        LocalDate fecha, . double valor);
```

```
        super (patente, fecha, valor);
```

```
        this.marca = marca;
```

```
        this.modelo = modelo;
```

3

```
    public double calcularImpuestos () {
```

```
        if (this.calcularAntiguedad () > 10) { return 0; }
```

```
        else { return this.valor * 0.05; }
```

3

3

```
// Test:
```

```
public class ContrabuyenteTest {
```

contrabuyente

```
private Contrabuyente contrabuyente;
```

```
    @Test  
    void contrabuyente_creaContrabuyenteConEmbarcaciones20Millon() {
```

```
        Contrabuyente contrabuyente = new Contrabuyente (20000000);
```

```
        assertEquals (20000000, contrabuyente.getEmbarcaciones () );
```

```
        private Contrabuyente (int embarcaciones) {
```

```
            this.embarcaciones = embarcaciones - 1million;
```

10

@BeforeEach

Void setUp() {

c. ~~Contribuyente~~ = new Contribuyente("n", "DNI", "e", "l") ;

}

@Test

Void testCalcularImpuestosSinBienes() {

assertTrue(0, c. calcularImpuestos());

}

@Test

Void testCalcularImpuestosConEmbarcaciones10Anos() {

Embarcacion e = new Embarcacion("p", "n", LocalDate.of(2002, 1, 1), 1);

c. registrarBien(e);

assertEquals(0, e. calcularImpuestos());

3 @Test

Void testCalcularImpuestosConEmbarcaciones-10Anos-1millon() {

Embarcacion e = new Embarcacion("p", "n", LocalDate.of(2013, 1, 1), 999999.0);

c. registrarBien(e);

assertEquals(999999.0, e. calcularImpuestos());

@Test

Void testCalcularImpuestoConEmbarcaciones-10Anos+1millon() {

Embarcacion e = new Embarcacion("p", "n", LocalDate.of(2013, 1, 1), 1000000.0);

c. registrarBien(e);

assertEquals(1500000.0, e. calcularImpuestos());

3

\*7 - numPólizasPartida: String  
 - ValorDeLote: Real  
 - ValorEdificación: Real

## Agencia

+<<creates>> Agencia()

+ alta Contribuyente( nombre: String, string dni, string email, string localidad): Contribuyente

+ alta Inmueble( numPartida: String, ~~ValorDeLote: Real~~, ValorDeEdificación: Real, propietario: Contribuyente): Inmueble

+ alta Automotor( patente: String, marca: String, FechaFabricación: Date, valor: Real, propietario: Contribuyente): Automotor

+ alta Embarcación( patente String, nombre, FechaFabricación: Date, valor: Real, propietario: Contribuyente)

+ calcularImpuestos(c Contribuyente): Real

+ nMasPayers(localidad: String, N: Integer).

## Contribuyente [\*]

↓ 0..\* - Contribuyentes

## Contribuyente

- nombre: String - DNI: String - email: String - localidad: String

+<<create>> Contribuyente(nombre: String, DNI: String)

email: String, localidad: String)

+ getLocalidad(): String

+ calcularImpuestos(): Real

+ registrarBien(Bien: Bien)

## <<interface>>

### Bien

+<<abstract>> calcularImpuesto()

→ Real

0  
1  
\*  
0

### Inmueble

+<<create>> Inmueble( numeroDePartida: String, valorDeLote: Real, valorEdificación: Real)

+ calcularImpuesto(): Real

### Vehículo

- patente: String

- ~~marca~~ Valor: Real

- FechaDeFabricación: Date

+<<create>> Vehículo(patente: String, fechaFabricación: Date, valor: Real)

+<<create>> calcularAntigüedad(): Integer

### Embarcación

- nombre: String - Embarcación

+<<create>> Embarcación(patente: String, nombre: String, fechaFabricación: Date, valor: Real),

+ calcularImpuesto(): Real

### Automotor

- marca: String

- modelo: String

+<<create>> Automotor(patente: String, marca: String, modelo: String, fecha: Date, valor: Real)

+ calcularImpuesto()