

Ejercicio U5_B1_E1

```
class Punto {
    static int contador=0;
    int x , y ;

    Punto ( int x, int y ) {
        this.x = x ;
        this.y = y;
        //aquí NO es obligatorio el nombre de clase para acceder a contador de Punto
        contador++;
        //Punto.contador++;//también ok
    }
}

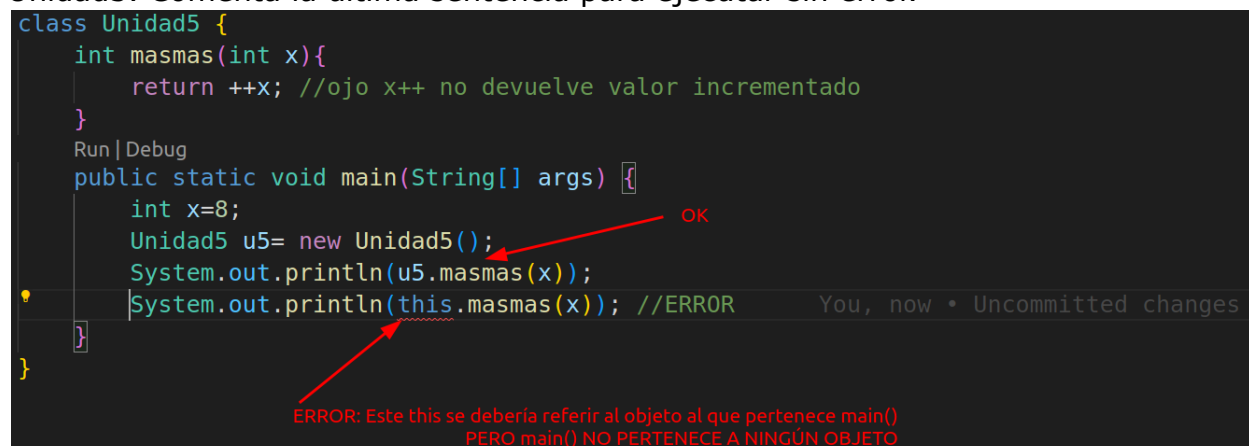
class Unidad5 {
    public static void main(String[] args) {
        new Punto(0,0);new Punto(3,0);new Punto(2,0);new Punto(0,4);
        //aquí es obligatorio el nombre de clase para acceder a contador de Punto
        System.out.println("total de puntos: "+Punto.contador);
    }
}
```

Ejercicio U5_B1_E2

Una solución es declarar masmas() static.

```
class Unidad5 {
    static int masmas(int x){
        return ++x; //ojo x++ no devuelve valor incrementado
    }
    public static void main(String[] args) {
        int x=8;
        System.out.println(masmas(x));
    }
}
```

Otra solución es dejar a masmas() como método instancia y crear un Objeto de tipo Unidad5. Comenta la última sentencia para ejecutar sin error.



```
class Unidad5 {
    int masmas(int x){
        return ++x; //ojo x++ no devuelve valor incrementado
    }
    Run | Debug
    public static void main(String[] args) {
        int x=8;
        Unidad5 u5= new Unidad5();
        System.out.println(u5.masmas(x));
        System.out.println(this.masmas(x)); //ERROR
    }
}
```

OK

ERROR: Este this se deberia referir al objeto al que pertenece main()
PERO main() NO PERTENECE A NINGÚN OBJETO

You, now • Uncommitted changes

Ejercicio U5_B1_E3

```
class Racional{
    int numerador;
    int denominador;
    Racional(int numerador, int denominador){
        this.numerador=numerador;
        this.denominador=denominador;
    }
    static Racional multiplicar(Racional r1, Racional r2){
        return new Racional (r1.numerador*r2.numerador,r1.denominador*r2.denominador);
    }
}
class Unidad5{
    public static void main(String[] args) {
        Racional r1=new Racional(3,4);
        Racional r2=new Racional(1,2);
        //ahora el tercer racional lo crea el método multiplicar
        Racional r3;
        r3=Racional.multiplicar(r1, r2);
        System.out.println("MULTIPLICACIÓN DE NÚMEROS RACIONALES");
        System.out.println("r1 vale: "+r1.numerador+"/"+r1.denominador);
        System.out.println("r2 vale: "+r2.numerador+"/"+r2.denominador);
        System.out.println("r3 vale: "+r3.numerador+"/"+r3.denominador);
    }
}
```

Ejercicio U5_B1_E4

```
class Unidad5{
    static int pot(int base, int exponente){
        int resultado=1;
        for(;exponente>0;exponente--){
            resultado=resultado*base;
        }
        return resultado;
    }
    public static void main(String[] args) {
        System.out.println(Unidad5.pot(2,1));
        System.out.println(pot(5,3));
        System.out.println(new Unidad5().pot(9,0));
    }
}
```

Ejercicio U5_B1_E5

```
class Unidad5{
    public static void main(String[] args) {
        System.out.println(Math.sqrt(25));
        System.out.println(Math.pow(2,8));
    }
}
```

Ejercicio U5_B1_E6

```
class MiClase{
    private int a=6;
    private static int b=7;
```

```

public int getA() {
    return a;//this.a
}

public void setA(int a) {
    this.a = a;
}

public static int getB() {
    return b;//Miclase.b i no es this.b !
}

public static void setB(int b) {
    MiClase.b = b;
}

```

}

Para los atributos static sus correspondientes s get/set también lo son. Si un atributo b es static y hay mil objetos para que guardar en cada objeto un método que devuelve siempre el valor b que es static, es más eficiente hacer un getB() común para todos los objetos de la clase.