Miembros Static: miembros de clase
Hasta ahora hemos visto que cada vez que creamos un objeto de una clase obtenemos una copia de todos los atributos de la clase.

Miembros static o miembros de clase se llaman así porque le pertenece a la clase. No a un objeto instanciado.

static: los atributos miembros de una clase pueden ser atributos de clase o atributos de instancia; se dice que son atributos de clase si se usa la palabra clave static: en ese caso la variable es única para todas las instancias (objetos) de la clase (ocupa un único lugar en memoria). A veces a las variables de clase se les llama variables estáticas. Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (la variable es diferente para cada objeto). En el caso de una constante no tiene sentido crear un nuevo lugar de memoria por cada objeto de una clase que se cree. Por ello es adecuado el uso de la palabra clave static. Cuando usamos "static final" se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase.
Veamos un Ejemplo:

```
public class Principal {
    public String miembroNoStatic = "una oracion";

    public static void main(String args []) {
         Principal obj1 = new Principal();

         Principal obj2 = new Principal();

         System.out.println(obj1.miembroNoStatic);
         System.out.println(obj2.miembroNoStatic);


    }
}
```

Cuando creamos un obj hace una copia distinta de toda la clase.
Si hacemos que uno de los objetos ahora diga : `"segunda oracion"` vemos que cada uno de los objetos tiene una copia distinta, por eso: si corremos vemos que primero muestra `"una oracion";` y luego `"segunda oracion";` cada uno con sumcopia.

```
public class Principal {
    public String miembroNoStatic = "una oracion";

    public static void main(String args []) {
         Principal obj1 = new Principal();

         Principal obj2 = new Principal();
         obj2.miembroNoStatic = "segunda oracion";

         System.out.println(obj1.miembroNoStatic);
         System.out.println(obj2.miembroNoStatic);
    }
}
```

Pero ahora hacemos el atributo static vemos:

```
public class Principal {
    public static String miembroNoStatic = "una oracion";

    public static void main(String args []) {
         Principal obj1 = new Principal();

         Principal obj2 = new Principal();

         obj2.miembroNoStatic = "segunda oracion";

         System.out.println(obj1.miembroNoStatic);

         System.out.println(obj2.miembroNoStatic);

    }
}
```

Entonces todos los objetos ahora comparte el atributo static, entonces si cambio el atributo este cambiará para todos los objetos. Además ya no necesitan que sean instanciados para ser usados.

```
public class Principal {
    public static String miembroNoStatic = "una oracion";

    public static void main(String args []) {

         System.out.println(Principal.miembroNoStatic);

    }
}
```

Lo mismo sucede con los métodos:

```
public class Principal {
    public static String miembroNoStatic = "una oracion";

    public static void main(String args []) {

         System.out.println(Principal.sumar(2,3));

    }

    public static int sumar(int uno, int dos) {
         int suma = uno + dos;
         return suma;

    }
}
```

Una clase, método o campo o miembro declarado como estático puede ser accedido o invocado sin la necesidad de tener que instanciar un objeto de la clase.

Los campos o miembros de una clase declarados como estáticos son inicializados en el momento en que se carga la clase en memoria, respetando el orden de declaración. Los campos estáticos no pueden ser accedidos desde un contexto no estático; este comportamiento resultará en un error en tiempo de ejecución.

Debido a que los métodos estáticos son enlazados en tiempo de compilación mediante static binding usando la información del tipo, no es posible realizar sobreescritura (*override*) de métodos.

Los métodos estáticos no pueden ser declarados como *abstract*, por el mismo motivo que la imposibilidad de la sobrecarga de métodos.

EJERCICIO

Define una clase Java denominada Circulo que tenga como **atributo de clase (estático) y constante** numeroPi, siendo esta constante de tipo double y valor 3.1416. Además la clase tendrá el atributo radio (tipo double) que representa el radio del círculo, y los métodos para obtener y establecer los atributos. También debe disponer de un método para calcular el área del círculo (método tipo funcion areaCirculo que devuelve el área) y la longitud del círculo (método tipo función que devuelve la longitud). Busca información sobre las fórmulas necesarias para crear estos métodos en internet si no las recuerdas. En una clase con el método main, declara el código que cree un objeto círculo, le pida al usuario el radio y le devuelva el área y la longitud del círculo.

¿Es posible crear un método en la clase Circulo para establecer el valor de numeroPi? ¿Por qué?