

Miembros estáticos

Variables de instancia, variables de clase

Ya mencionamos la diferencia entre variables de instancia, de clase y locales. Consideremos de nuevo la clase *Circulo*, con dos miembros dato, el *radio* específico para cada círculo y el número *PI* que tiene el mismo valor para todos los círculos. La primera es una variable de instancia y la segunda es una variable de clase.

Para indicar que el miembro *PI* es una variable de clase se le antepone el modificador **static**. El modificador **final** indica que es una constante que no se puede modificar, una vez que la variable *PI* ha sido inicializada.

Definimos también una función miembro denominada *calcularArea* que devuelva el área del círculo

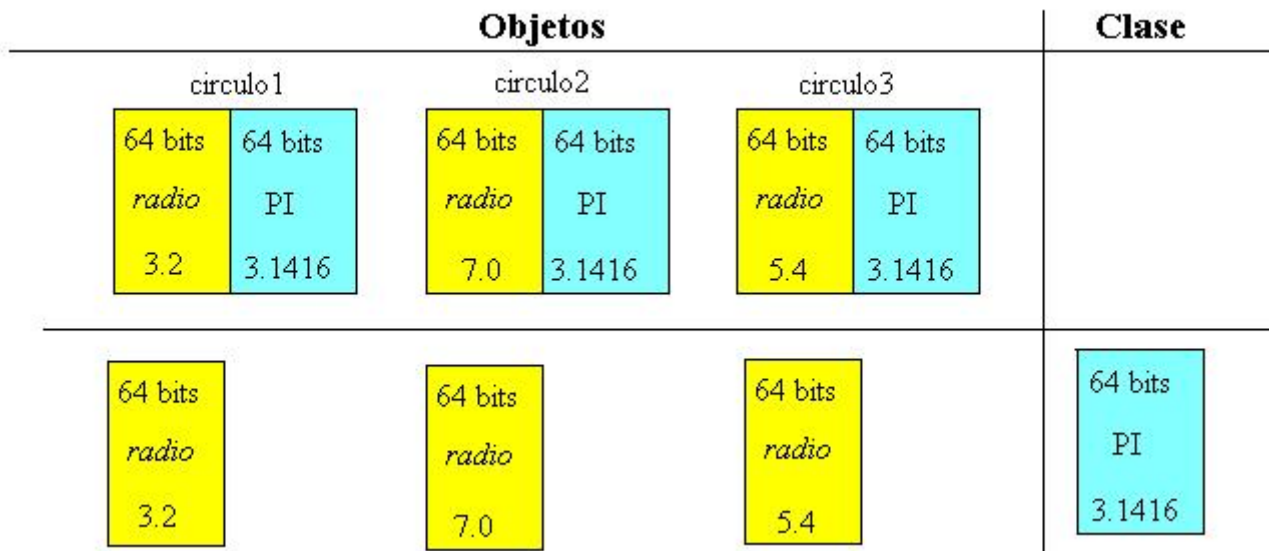
```
public class Circulo{
    static final double PI=3.1416;
    double radio;
    public Circulo(double radio){
        this.radio=radio;
    }
    double calcularArea(){
        return (PI*radio*radio);
    }
}
```

Para calcular el área de un círculo, creamos un objeto *circulo* de la clase *Circulo* dando un valor al radio. Desde este objeto llamamos a la función miembro *calcularArea*.

```
Circulo circulo=new Circulo(2.3);
System.out.println("área: "+circulo.calcularArea());
```

```
public class CirculoApp {
    public static void main(String[] args) {
        Circulo circulo=new Circulo(2.3);
        System.out.println("área: "+circulo.calcularArea());
    }
}
```

Veamos ahora las ventajas que supone declarar la constante *PI* como miembro estático.



Si PI y *radio* fuesen variables de instancia

```
public class Circulo{
    double PI=3.1416;
    double radio;
    //....
}
```

Creamos tres objetos de la clase *Circulo*, de radios 3.2, 7.0, y 5.4

```
Circulo circulo1=new Circulo(3.2);
Circulo circulo2=new Circulo(7.0);
Circulo circulo3=new Circulo(5.4);
```

Al crearse cada objeto se reservaría espacio para el dato *radio* (64 bits), y para el dato PI (otros 64 bits). Véase la sección tipos de datos primitivos. Como vemos en la parte superior de la figura, se desperdicia la memoria del ordenador, guardando tres veces el mismo dato PI.

```
public class Circulo{
    static double PI=3.1416;
    double radio;
    //....
}
```

Declarando PI estático (**static**), la variable PI queda ligada a la clase *Circulo*, y se reserva espacio en memoria una sólo vez, tal como se indica en la parte inferior de la figura. Si además la variable PI no cambia, es una constante, le podemos anteponer la palabra **final**.

```
public class Circulo{
    static final double PI=3.1416;
    double radio;
    //....
}
```

Miembros estáticos

Las variables de clase o miembros estáticos son aquellos a los que se antepone el modificador **static**. Vamos a comprobar que un miembro dato estático guarda el mismo valor en todos los objetos de dicha clase.

Sea una clase denominada *Alumno* con dos miembros dato, la nota de selectividad, y un miembro estático denominado nota de corte. La nota es un atributo que tiene un valor distinto para cada uno de los alumnos u objetos de la clase *Alumno*, mientras que la nota de corte es un atributo que tiene el mismo valor para a un conjunto de alumnos. Se define también en dicha clase una función miembro que determine si está (**true**) o no (**false**) admitido.

```
public class Alumno {
    double nota;
    static double notaCorte=6.0;
    public Alumno(double nota) {
        this.nota=nota;
    }
    boolean estaAdmitido(){
        return (nota>=notaCorte);
    }
}
```

Creamos ahora un array de cuatro alumnos y asignamos a cada uno de ellos una nota.

```
Alumno[] alumnos={new Alumno(5.5), new Alumno(6.3),
    new Alumno(7.2), new Alumno(5.0)};
```

Contamos el número de alumnos que están admitidos

```
int numAdmitidos=0;
for(int i=0; i<alumnos.length; i++){
    if (alumnos[i].estaAdmitido()){
        numAdmitidos++;
    }
}
System.out.println("admitidos "+numAdmitidos);
```

Accedemos al miembro dato *notaCorte* desde un objeto de la clase *Alumno*, para cambiarla a 7.0

```
alumnos[1].notaCorte=7.0;
```

Comprobamos que todos los objetos de la clase *Alumno* tienen dicho miembro dato estático *notaCorte* cambiado a 7.0

```
for(int i=0; i<alumnos.length; i++){
    System.out.println("nota de corte "+alumnos[i].notaCorte);
}
```

El miembro dato *notaCorte* tiene el modificador **static** y por tanto está ligado a la clase más que a cada uno de los objetos de dicha clase. Se puede acceder a dicho miembro con la siguiente sintaxis

```
Nombre_de_la_clase.miembro_estático
```

Si ponemos

```
Alumno.notaCorte=6.5;
for(int i=0; i<alumnos.length; i++){
    System.out.println("nota de corte "+alumnos[i].notaCorte);
}
```

Veremos que todos los objetos de la clase *Alumno* habrán cambiado el valor del miembro dato estático *notaCorte* a 6.5.

Un miembro dato estático de una clase se puede acceder desde un objeto de la clase, o mejor, desde la clase misma.

```
public class AlumnoApp {

    public static void main(String[] args) {
        Alumno[] alumnos={new Alumno(5.5), new Alumno(6.3), new
Alumno(7.2), new Alumno(5.0)};
//alumnos admitidos
        int numAdmitidos=0;
        for(int i=0; i<alumnos.length; i++){
            if (alumnos[i].estaAdmitido()){
                numAdmitidos++;
            }
        }
        System.out.println("admitidos "+numAdmitidos);
//cambiar la nota de corte
        Alumno.notaCorte=6.5;
        for(int i=0; i<alumnos.length; i++){
            System.out.println("nota de corte "+alumnos[i].notaCorte);
        }
    }
}
```