

Clases anidadas

Programación I

Grado en Ingeniería Informática

MDR, JCRdP y JDGD

Clases anidadas

- ▶ Java permite definir clases dentro de otras clases

```
class Clase {
```

```
    ...
```

```
    [modificadores] class ClaseAnidada {
```

```
        ...
```

```
    }
```

```
}
```

- ▶ Existen dos tipos de clases anidadas: estáticas y no estáticas
 - Clases anidadas estáticas (con modificador **static**)
 - Clases internas (no estáticas, sin modificador **static**)

Clases anidadas

- ▶ **Estáticas:** se deben declarar como miembros de la clase que las engloba
- ▶ Las **clases internas** (no estáticas) se pueden declarar miembros o no. **Cada objeto de la clase interna está unido a un objeto de la clase que la engloba.** Tienen acceso a los elementos (atributos y métodos) de la clase que la engloba.
- ▶ Las clases miembro (estáticas o no) se pueden declarar: públicas, privadas, protegidas y privadas de paquete.
 - Las clases de nivel superior (normales) se pueden declarar solo públicas o privadas de paquete.

¿Para qué usar clases anidada?

- ▶ Para agrupar clases que se usan sólo en un ámbito concreto
- ▶ Para aumentar el encapsulamiento permitiendo ocultar clases completas
- ▶ Permite escribir código más legible y con menos coste de mantenimiento, al tener la definición de las clases cerca de donde se usan

Clases estáticas anidadas

- ▶ Normalmente, se emplean como clases de uso interno de otras o a las que se les da permiso de manipular detalles internos de otra
- ▶ A los efectos son como las clases de nivel superior con dos diferencias:
 - Cumplen las restricciones de accesibilidad según sean públicas, privadas, protegidas o privadas de paquete
 - Se referencian externamente como:
NombreClase.NombreClaseAnidada
- ▶ Ejemplo:

```
Clase.ClaseAnidada obj = new Clase.ClaseAnidada();
```

Ejemplo de clase estática

```
public class Lista{
    private static class Nodo{
        public Object dato;
        public Nodo siguiente;
    }
    private Nodo primero;
    public void añade(Object ref){
        Nodo nuevo = new Nodo();
        nuevo.dato = ref;
        if(primero == null){
            primero = nuevo;
        }else{
            Nodo aux=primero;
            while( aux.siguiente != null){
                aux = aux.siguiente;
            }
            aux.siguiente = nuevo;
        }
    }
    ...
}
```

Clases internas

- ▶ Cuando se crea un objeto de una clase interna, dentro de un método de la clase que la engloba, a éste se le asocia el objeto actual (**this**) de la clase que la engloba.
- ▶ Cuando se crea fuera de un método de la clase que la engloba, es necesario, en el **new**, especificar el objeto de la clase que la engloba que se le asociará:

```
Clase.ClaseInterna objInterna=  
    objeto.new Clase.ClaseInterna();
```

Ejemplo de clase miembro no estática

```
public class Pila{
    private class Nodo{
        private Object dato;
        private Nodo siguiente;
        public Nodo(Object o){ //Añade al principio de la lista
            dato = o;
            siguiente = primero; //primero del objeto asociado
            primero = this;
        }
        ...
    }
    private Nodo primero;
    public void añade(Object ref){
        new Nodo(ref);
    }
    ...
}
```


Clases locales

- ▶ Son clases internas definidas dentro de un método (no estático) o bloque de código.
- ▶ Sólo pueden crearse objetos de estas clases dentro del ámbito de definición (método/bloque)
- ▶ Si implementan una interface o derivan de otra clase, sus objetos podrán ser usados en otros ámbitos debido al polimorfismo

Clases locales anónimas

- ▶ Es posible crear clases en una expresión new, estas clases a los efectos son locales.
- ▶ Los objetos creados se emplean normalmente para ser pasados a otros métodos.
- ▶ Las clases anónimas tienen sentido cuando implementan una interfaz o extienden una clase.
- ▶ El formato es:

```
new NombreInterface o NombreClase([parámetros])  
{  
    [{ código inicialización}]  
    //Métodos redefinidos o implementan una interfaz  
    ...  
}
```

Ejemplo de clase anónima

Comparator<String> como clase normal

```
class MiOrden implements Comparator<String>{
    @Override
    public int compare(String o1, String o2) {
        if(o1.length()==o2.length()){
            return o1.compareTo(o2);
        }
        return o1.length()-o2.length();
    }
}
...
SortedSet<String> datos = new TreeSet<String>(new MiOrden());
...
```

Ejemplo de clase anónima

Comparator<String> como clase anónima

```
SortedSet<String> datos = new TreeSet<String>(  
    new Comparator<String>() {  
        @Override  
        public int compare(String o1, String o2) {  
            if(o1.length()==o2.length()){  
                return o1.compareTo(o2);  
            }  
            return o1.length()-o2.length();  
        }  
    }  
);  
...
```

Tipos de clases anidadas en Java

Clases anidadas
Las definidas dentro de otra

Clases miembro
Definidas al mismo nivel
que métodos y atributos

Estática

Con modificador
`static`
Independiente de
la que la engloba

Sin modificador
`static`
dependiente de
un objeto de la
que la engloba

No estáticas

Locales

Definida
dentro de
un método

Anónimas
Sin nombre,
implementa
o deriva