

# Modificadores de Acceso



# Modificadores de Acceso

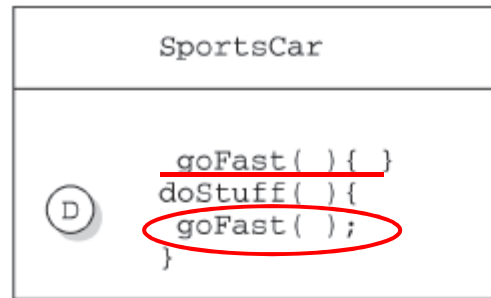
- Public
- Protected
- Private

Variables

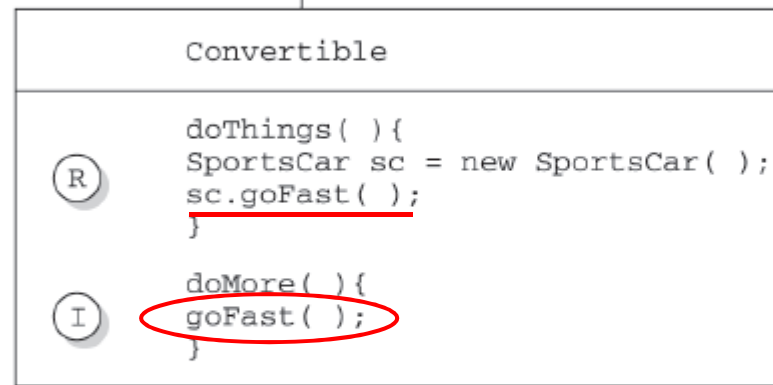
Métodos



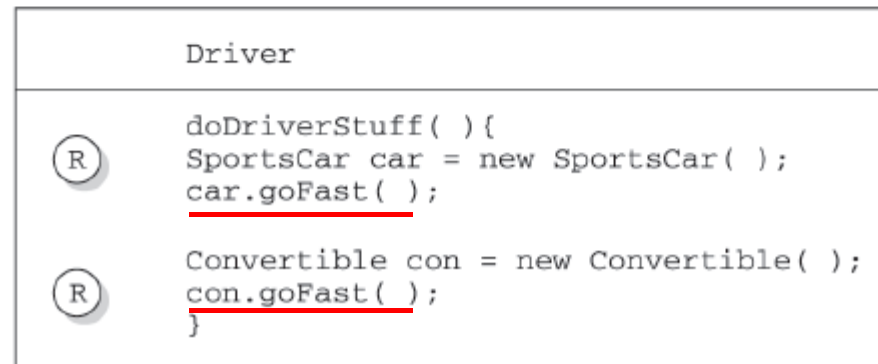
## Herencia



Invocar un Método de la misma clase  
superclass



Invocar un Método usando la  
referencia de una clase  
subclass



Invocar un Método Heredado



# Public

- Corresponde al nivel de acceso menos restrictivo
- Un método o variable que es declarado público, significa que el resto de las clases, sin importar el paquete, pueden tener acceso (siempre que exista visibilidad entre clases)



# Public

## Diferentes Paquetes

```
package book;  
import cert.*; // Import all classes in the cert package  
class Goo {  
    public static void main(String[] args) {  
        Sludge o = new Sludge();  
        o.testIt();  
    }  
}
```

Invoca por ser público

```
package cert;  
public class Sludge {  
    public void testIt() { System.out.println("sludge"); }  
}
```

Público



```
package cert;
public class Roo {
    public String doRooThings() {
        // imagine the fun code that goes here
        return "fun";
    }
}

class Toon {
    public static void main(String[] args) {
        Cloo c = new Cloo();
        System.out.println(c.doRooThings()); //No problem; method
                                           // is public
    }
}
```

```
package notcert; //Not the package Roo is in
import cert.Roo;
class Cloo extends Roo {
    public void testCloo() {
        System.out.println(doRooThings());
    }
}
```



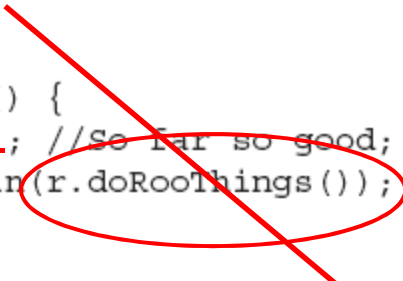
# Private

- El acceso de tipo private corresponde al nivel de acceso más restrictivo. Los miembros privados sólo son accesibles dentro del cuerpo de la clase o estructura en la que se declaran
- Los miembros marcados como privados no pueden ser accedidos en ninguna otra clase que no sea en la que se declaró



```
package cert;
public class Roo {
    private String doRooThings() {
        // imagine the fun code that goes here, but only the Roo
        // class knows
        return "fun";
    }
}
```

```
package notcert;
import cert.Roo;
class UseARoo {
    public void testIt() {
        Roo r = new Roo(); //So far so good; class Roo is public
        System.out.println(r.doRooThings()); //Compiler error!
    }
}
```

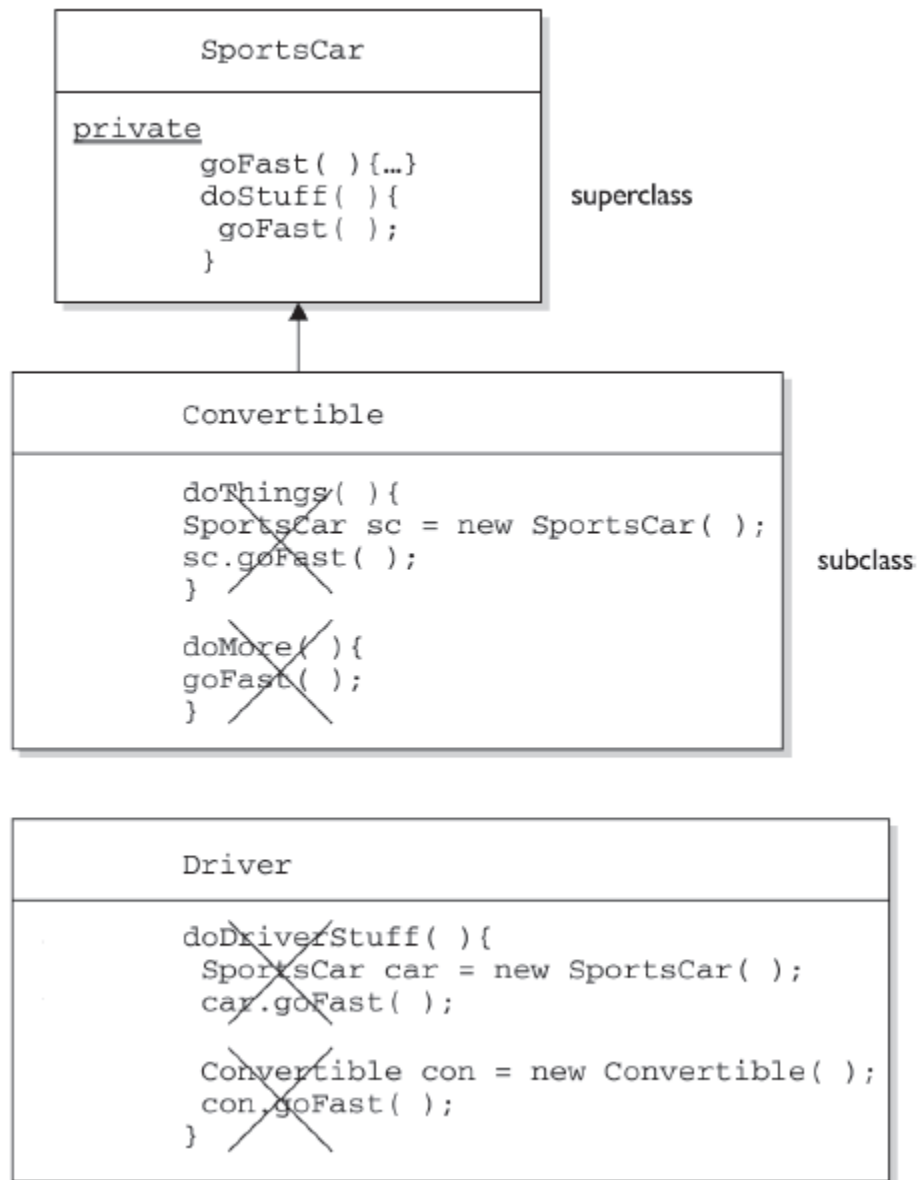




```
package cert;
public class Roo {
    private String doRooThings() {
        // imagine the fun code that goes here, but no other class
        // will know
        return "fun";
    }
}
```

```
package cert;                                //Cloo and Roo are in the same package
class Cloo extends Roo { //Still OK, superclass Roo is public
    public void testCloo() {
        System.out.println(doRooThings()); //Compiler error!
    }
}
```





# Protected

- Un miembro protegido es accesible dentro de su clase y por clases derivadas
- Puede ser accedida a través de herencia, inclusive si la subclase se encuentra en otro paquete



Neighbor  
Referencia de  
Child

```
package certification;  
public class Parent {  
    protected int x = 9; // protected access  
}
```

```
package other; // Different package  
import certification.Parent;  
class Child extends Parent {  
    public void testit() {  
        System.out.println("x is " + x); // No problem; Child  
                                           // inherits x  
    }  
}
```

```
Parent p = new Parent(); // Can we access x using the  
                           // p reference?  
System.out.println("X in parent is " + p.x); // Compiler  
                                              // error!
```



# Resumen

Visibilidad	Public	Protected	Default	Private
Desde la misma clase	✓	✓	✓	✓
Desde otra clase en el mismo paquete	✓	✓	✓	✗
Desde una subclase en el mismo paquete	✓	✓	✓	✗
Desde una subclase fuera del mismo paquete	✓	✓	✗	✗
Desde una no subclase fuera del paquete	✓	✗	✗	✗