Clase 1-20/8

miércoles, 20 de agosto de 2025 14:40

Paquete es colección de componentes con un nombre., están buenos para agrupar y organizar.
JAVA.lang se importa de una, cualquier otro package lo tengo que importar.
Los nombres de paquetes tienen que ser únicos, me permiten tener el mismo nombre de clase dentro de dos paquetes distintos.
Puedo importar el paquete entero o solo la que voy a usar.

```
package pruebas;
import java.util.List;
import java.awt.List;
public class ConflictoDeNombres {
  //TODO
}
```

¿Se puede hacer?

```
package pruebas;
import java.util.*;
import java.awt.*;
public class ConflictoDeNombres {
    List l;
}
```

¿A qué List hace referencia?

```
package pruebas;
import java.util.*;
import java.awt.*;
public class ConflictoDeNombres {
    //TODO
}
```

NO

```
package pruebas;
import java.util.List;
public class ConflictoDeNombres {
   List l;
   java.awt.List  l2;
}
```

NO

```
package pruebas;
import java.util.*;
public class ConflictoDeNombres {
   List<String> l=new ArrayList<String>();
   java.awt.List  l2=new java.awt.List(4);
}
```

Podemos importar metodos estáticos que simplifican la escritura.
Ej:

```
package pruebas;
import static java.lang.System.out;
public class ImportOUT {
  public static void main(String[] args) {
    out.print("hola");
  }
}
```

Importar estáticamentre importa nombres, no un miembro con ese nombre.

La importación estática **importa nombres**, NO un miembro específico con dicho nombre. JAVA soporta sobrecarga de métodos y también permite que una clase defina atributos con el mismo nombre que un método -> al importar un miembro estático, podríamos importar más de un miembro (métodos y atributos).

Consideremos el siguiente ejemplo:

```
package pruebas;
import static java.util.Arrays.sort;
public class SobrecargaImportEstatico {
public static void main(String args[]) {
   String varones[]={"Juan", "Pedro", "Luis", "Ernesto"};
   sort(varones);
}
}
```

Importa el nombre **sort** no uno de los 18 métodos **sort()** definidos en la clase Arrays

El compilador analizando el tipo, cantidad y orden de los argumentos y determina cuál de lo

```
}
}
```

El compilador analizando el tipo, cantidad y orden de los argumentos y determina cuál de lo métodos **sort()** queremos usar.

Public desde cualquier paquete,
Si no pongo nada es privada dentro del paquete, solo la pueden usar dentro del paquete.
Cuando hago un import del paquete, importa solo nombres públicos

public	protected	package (no tiene palabra clave)	private

Más libre — Más restrictivo

A clases solo public y package, el resto para variables, metodos, etc.
Si a un metodo o una clase no les aclaro visibiliad, es package

```
package labo12;
public class Auto{
  public String marca;
  public Auto() {
    System.out.println("Constructor de Auto");
  }
  void arrancar(){
    System.out.println("arrancar");
  }
}
```

¿Qué observan en este código?

```
import labo12.*;
public class Carrera{
  public Carrera() {
    System.out.println("Constructor de Carrera");
  }
  public static void main(String[] args){
      Auto a=new Auto();
      System.out.println("Marca: "+ a.marca);
      a.arrancar();
  }
}
```

No puedo usar arrancar porque están en paquetes distintos y el mpetodo es privado al paquete.

Privado del paquete (package)

Las variables, métodos y constructores declarados **privados del paquete** son accesibles sólo desde clases pertenecientes al mismo paquete donde se declaran.

```
package labo12;
public class Auto {
public String marca;
Auto() {
   System.out.println("Constructor de Auto");
}
void arrancar(){
    System.out.println("arrancar");
}
}
```

¿Qué observan en el código?

¿Qué relación encuentran entre el especificador de acceso package y la herencia?

```
import labo12.*;
public class Carrera{
public Carrera() {
  System.out.println("Constructor de Carrera");
}
public static void main(String[] args){
  Auto a=new Auto();
  System.out.println("Marca: "+ a.marca);
  a.arrancar();
}
}
```

```
public class Sedan extends Auto {}
```

No se puede. La herencia en java se implementa como que cada objeto que hereda tiene una instancia del objeto padre. Como no puedo hacer new auto no va a andar porque cuando haga new sedan se va a llamar al new auto.

```
package labo12;
public class Auto{
public Auto() {
   System.out.println("Constructor de Auto");
}
void arrancar(){
   System.out.println("arrancar");
}
}
```

¿Qué observan en el código?

```
import labo12.*;
public class Sedan extends Auto{
 public Sedan() {
  System.out.println("Constructor de Sedan");
 }
public static void main(String[] args){
     Sedan x=new Sedan();
      x.arrancar();
 }}
```

Ahí no puede sedan acceder al arrancar porque está en otro paquete.
Con protectred, las subclases creadas por fuera del paquete tienen acceso a los miembros protegidos (package también). Protected se relaciona a full con la herencia. En este caso con un protected en arrancar te lo reconoce.
Desde afuera del paquete si creo un auto (no herencia) igual no podría usar arrancar.

protected

Analizaremos el acceso protected aplicado a variables.

```
package labo12;
public class Auto{
protected String marca;
protected String nroMotor;
public Auto() {
   System.out.println("Constructor de Auto");
}
protected void arrancar() {
   System.out.println("arrancar");
}
}
```

¿Qué observan en el código?

```
import labo12.*;
public class Sedan extends Auto{
private String identificador;
public Sedan() {
  System.out.println("Constructor de Sedan");
}
public void setIdentificador(Auto v){
   this.identificador=this.marca+v.nroMotor;
}
}
```

El this.marca anda , porque accedo por herencia, el v.nromotor no anda porque no heredo.

La idea es hacer todo lo más privado posible

private

Las variables, métodos y constructores declarados **private** solamente están accesibles para la clase que los contiene. Están disponibles para usar dentro de los métodos de dicha clase.

```
public class Postre {
private Postre() {
   System.out.println("Constructor de Auto");
}
}
```

```
public class Helado{
public static void main(String[] args){
 Postre  p=new Postre();
 }
}
```

No puedo hacer ese new pq el constructor es privado. Poddría hacer un metodo estático de clase postre que sea público y devuelva una instancia de postre.
Un método privado solo se poyede usar en la clase.
Puedo extender postre pero NO sobreescribir el metodo. Si heredo el postre no puedo sobreescribir el metodo.
Una subclase no hereda los metodos y clases inaccesibles para ella.

Podría ser confuso: "una subclase NO HEREDA los atributos y métodos de su superclase inaccesibles para ella" -> NO IMPLICA que cuando se crea una instancia de la subclase, no se use memoria para los atributos privados o inaccesibles definidos en la superclase.
Todas las instancias de una subclase incluyen una instancia COMPLETA de la superclase, incluyendo los miembros inaccesibles.
Como los miembros inaccesibles no puden usarse en la subclase, decimos NO SE HEREDAN.