Package e livelli di accesso

Un package è un insieme di classi

Di solito, un package contiene classi correlate fra loro

Esempio: nel package `java.awt` ci sono le classi `Point`, `Rectangle` ecc

Usare le classi di un package

La classe `Point` sta nel package `java.awt`

Tre modi per usare questa classe:

1. importare tutto il package:

   ```
   import java.awt.*;

   class Prova {
     Point p;
   }
   ```

2. importare solo la classe:

   ```
   import java.awt.Point;

   class Prova {
     Point p;
   }
   ```

3. usare il nome di package prima della classe:

   ```
   class Prova {
     java.awt.Point p;
   }
   ```

Definire un nuovo package

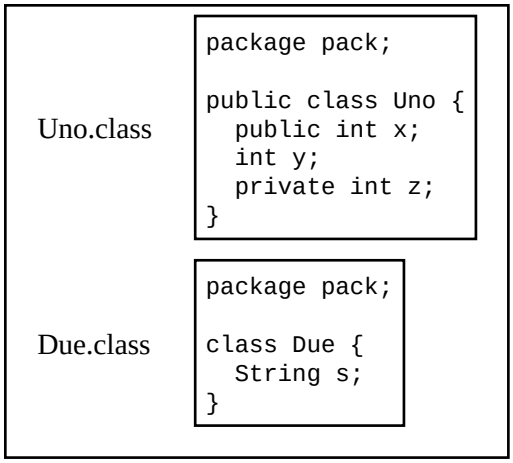
Tutte le classi del package devono stare nella stessa directory

Ogni file inizia con la dichiarazione:

```
package nomepackage;
```

Definizione di package: esempio

directory `pack/`:



Quello che conta sono i file `.class`

Nella figura sono stati messi i corrispondenti file sorgente `.java`

Uso del package

Il nuovo package si può usare come quelli predefiniti

Per esempio:

```
import pack.*;

class Prova {
  public static void main(String args[]) {
    Uno a=new Uno();

    a.x=12;
  }
}
```

Perchè `public`?

Per le classi, esistono due livelli di accesso:

- `public`
- (nessuna specifica)

Quando non si scrive nulla, l'accesso è consentito soltanto alle altri classi dello stesso package

Nelle classi del nostro package:

`public class Uno {...}`
: la classe è accessibile a tutti

`class Due {...}`
: la classe si può usare solo all'interno del package

Accesso alle componenti e ai metodi

Esistono quattro livelli di accesso:

- `public`
- `protected`
- (nessuna specifica)
- `private`

Anche qui, nessuna specifica=accesso solo da altre classi dello stesso package

```
package pack;

public class Uno {
  public int x;
  int y;
  private int z;
}
```

`public int x;`
: la componente `x` è visibile a tutti

`int y;`
: la compomente `y` è visibile solo alle classi che stanno dentro il package (`pack`, in questo caso)

`private int z;`
: si può accedere a `z` solo all'interno di questa classe

Lo stesso vale anche per i metodi

Se non dichiaro il package?

Se in una classe non si specifica un package:

Esiste un package di default, che contiene tutte le classi che non hanno una specifica di package

Quando si omette la specifica di accesso, si permette l'accesso all'interno dello stesso package

Vale anche in nel caso del package senza nome

Livello `protected`

Se una componente o metodo è dichiarata `protected` allora è visibile:

- a tutte le classi dello stesso package
- a tutte le sue sottoclassi, anche in package diversi

A che serve?

Differenza `protected`/nulla

Il livello di accesso di default (nessuna specifica di accesso) impedisce l'accesso a tutte le classi fuori dal package, *anche le sottoclassi*

Con `protected`, le sottoclassi hanno accesso

Diritti di accesso: principi

Principio generale: restringere il più possibile

Riformulato: permettere l'accesso solo alle classi, componenti e metodi di cui le altre classi hanno realmente bisogno

Conseguenza: non permettere l'accesso a metodi e componenti ausiliari

Vantaggio: se si cambia l'implementazione, le altre classi non hanno bisogno di modifiche

---

Incapsulamento

Rendere private tutte le componenti, e pubblici solo i metodi che fanno operazioni permesse alle altre classi

```
class Incap {
  private int x;

  public int getX() {
    return this.x;
  }

  public void setX(int x) {
    this.x=x;
  }
}
```

Vantaggio: posso restringere/eliminare l'accesso in lettura o scrittura modificando i metodi

Posso aggiungere controlli (es., `x` deve essere positivo)