

# 15 - Packages

Programmazione e analisi di dati  
Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa  
<http://www.di.unipi.it/~milazzo>  
[milazzo@di.unipi.it](mailto:milazzo@di.unipi.it)

Corso di Laurea Magistrale in Informatica Umanistica  
A.A. 2013/2014

# Packages (1)

I **packages** in Java sono un meccanismo che consente di **raggruppare** le classi.

La **Libreria Standard** di Java (**Java API**) è organizzata in packages:

- `String` è una classe del package `java.lang`
- `Scanner` è una classe del package `java.util`

Un package riunisce classi **logicamente correlate** tra loro, ad esempio:

- Il package `java.lang` riunisce classi fondamentali del linguaggio Java (`String`, `Math`, ....)
- Il package `java.util` riunisce classi di frequente utilizzo (`Scanner`, `Random`, `Timer`, ....)
- I packages `java.awt` e `java.swing` riuniscono classi per costruire interfacce grafiche

## Packages (2)

Spesso è utile definire uno o più packages per le **proprie classi**:

- quando il programma inizia a diventare complesso (molte classi) un raggruppamento in packages può consentire di fare **ordine**
- ad esempio, se stiamo realizzando un programma di gestione di una banca potremmo organizzare le nostre classi nei seguenti packages:
  - ▶ **core** per le classi che costituiscono il “nucleo” del nostro programma (gestione conti correnti, utenti, ecc...)
  - ▶ **gui** per le classi che gestiscono l'interfaccia grafica (Graphical User Interface, GUI)
  - ▶ **net** per le classi che gestiscono il collegamento via rete con la sede centrale

## Packages (3)

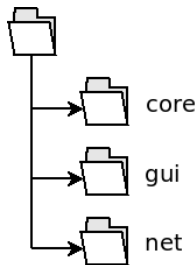
Per specificare in quale package debba essere inclusa una classe bisogna usare la primitiva **package** all'inizio del file sorgente della classe stessa

```
package core;  
  
public class ContoCorrente {  
    .....  
}
```

Inoltre, i file java delle varie classi devono essere salvati in **diverse directory** che corrispondono ai vari packages.

Il compilatore Java dà errore se i file non sono nelle directory giuste...

Se non si specifica nessun package, la classe farà parte del package **default** corrispondente alla directory principale



## Packages (4)

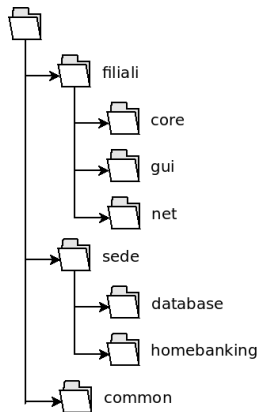
I packages possono essere ulteriormente raggruppati, formando una **struttura gerarchica**.

Il raggruppamento viene fatto tramite un **prefisso** nel nome del package

Nel nostro programma potremmo suddividere i packages tra quelli relativi a filiali, sede centrale e condivisi:

- `filiali.core`
- `filiali.gui`
- `filiali.net`
- `sede.database`
- `sede.homebanking`
- `common`

La strutturazione si riflette nell'organizzazione in directory!



# Packages e modificatori di visibilità (1)

I packages hanno anche un ruolo legato alla **visibilità** dei membri di classi.

Rivediamo la tabella dei modificatori di visibilità:

<code>private</code>	Utilizzabile solo all'interno della stessa classe
senza modificatore	Utilizzabile solo nel <b>package</b> che contiene la classe
<code>protected</code>	Utilizzabile nel <b>package</b> che contiene la classe, e in tutte le classi che <b>ereditano</b> da essa
<code>public</code>	Utilizzabile ovunque

La differenza tra “senza modificatore” e `protected` la vedremo a breve...

## Packages e modificatori di visibilità (2)

Esempio:

```
package filiali.core;

public class ContoCorrente {

    double saldo; // senza modificatore

    .... // altri membri

}
```

```
package filiali.core;

public class ControlloreConti {

    public void controlla(ContoCorrente x) {

        ....
        if (x.saldo==0) .... // OK, stesso package
        ....

    }

}
```

## Packages e modificatori di visibilità (3)

Esempio:

```
package filiali.core;  
  
public class ContoCorrente {  
    double saldo; // senza modificatore  
    .... // altri membri  
}
```

```
package filiali.gui;  
  
public class FinestraPrincipale {  
    public void visualizzaSaldo(ContoCorrente x) {  
        ....  
        if (x.saldo==0) .... // ERRORE, package diverso  
        ....  
    }  
}
```



## Packages e modificatori di visibilità (4)

Esempio:

```
package filiali.core;

public class ContoCorrente {

    double saldo; // senza modificatore

    .... // altri membri
}
```

```
public class GestioneFiliale {

    public static void main(String[] args) {

        ....
        if (x.saldo==0) .... // ERRORE, package diverso (default)
        ....
    }

}
```