14 - Packages

Programmazione e analisi di dati
Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
http://pages.di.unipi.it/milazzo
milazzo@di.unipi.it

Corso di Laurea Magistrale in Informatica Umanistica
A.A. 2019/2020

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

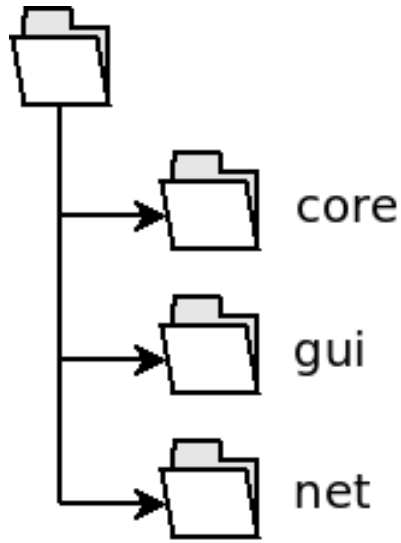
```
package core;

public class ContoCorrente {
    .....
}
```

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

Il compilatore Java da 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)

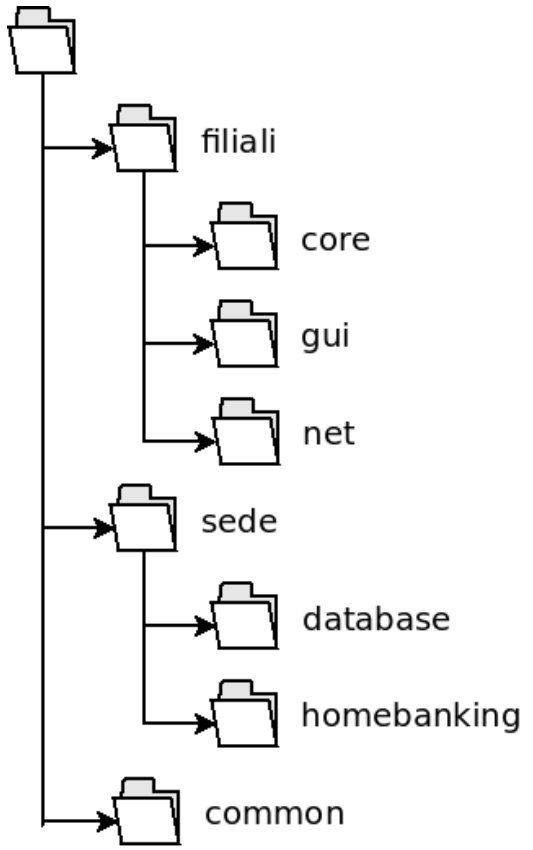
I packages possono essere ulteriormente raggruppati, formando una struttura gerarchica.

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 (5)

Quando si vuole utilizzare una classe che appartiene ad un package diverso da quello corrente bisogna usare la direttiva `import`

Esempio:

```
package filiali.core;

public class ContoCorrente {
    ....
}
```

```
package filiali.gui;

import filiali.core.ContoCorrente;

public class FinestraPrincipale {

    // Se non avessi importato la classe ContoCorrente
    //non potrei usarla in questo metodo
    public void visualizzaSaldo(ContoCorrente x) {

        ......

    }

}
```

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:

`private`	Utilizzabile solo all'interno della stessa classe
senza modificatore	Utilizzabile solo nel package che contiene la classe
`protected`	Utilizzabile nel package che contiene la classe, e in tutte le classi che ereditano da essa
`public`	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;

import filiali.core.ContoCorrente;

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
}
```

```
import filiali.core.ContoCorrente;

public class GestioneFiliale {

    public static void main(String[] args) {

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

    }

}
```