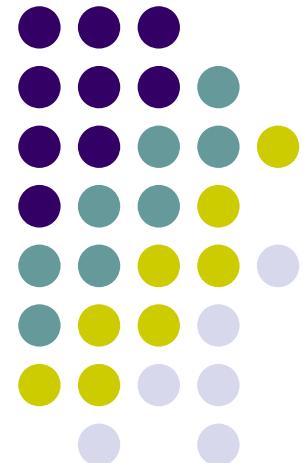
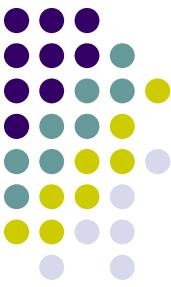


# Corso di Programmazione

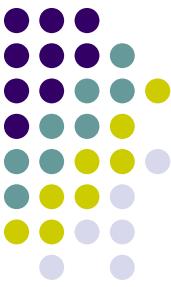
*Packages*





# Moduli in Java

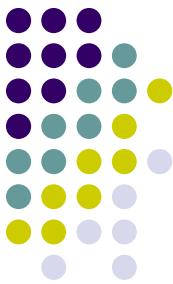
- Java **non** è un linguaggio modulare, cioè non offre meccanismi esplicativi per la creazione di moduli come visto nella lezione relativa alla programmazione modulare, sebbene qualcosa sia stata introdotta nelle ultime versioni
- Come per altri linguaggi (C, C++) è possibile però adottare una disciplina di programmazione che consente la organizzazione del software secondo alcuni principi tipici della modularizzazione
- In particolare le *classi* consentono di realizzare una certa astrazione sui dati e favoriscono l'adozione di uno stile modulare (information hiding, coesione)
- Inoltre Java introduce il concetto di package...



# Il concetto di package

- Un insieme di classi tra loro correlate (che cioè contribuiscono alla realizzazione di una unità concettuale) vengono generalmente raggruppate all'interno di un package
- Un package quindi è costituito da (molte) classi ciascuna delle quali può essere implementata su un file separato
- Il package favorisce ***l'organizzazione del software in componenti, il loro riuso e la realizzazione di librerie***

# Dichiarazione di package



- Una dichiarazione di package ha la forma:  
**package <nomepackage>;**
- Se presente, deve essere all'inizio del file, può essere preceduta solo da linee vuote e commenti
- Esiste una corrispondenza biunivoca fra il nome del package e posizione nel file system delle classi del package
  - Tutte le classi di un package devono trovarsi in una directory (cartella) che ha lo stesso nome del package

# Uso dei package per la gestione di sistemi di grandi dimensioni



- Un sistema software per la gestione di una biblioteca potrebbe essere stato sviluppato implementando le seguenti classi.

GestorePrestiti

InterfacciaGraficaBibliotecario

Libro

UtilityStampaLocale

UtilityAccessoInternet

CopiaDiLibro

UtilityReteLocale

Scaffale

GestoreScaffali

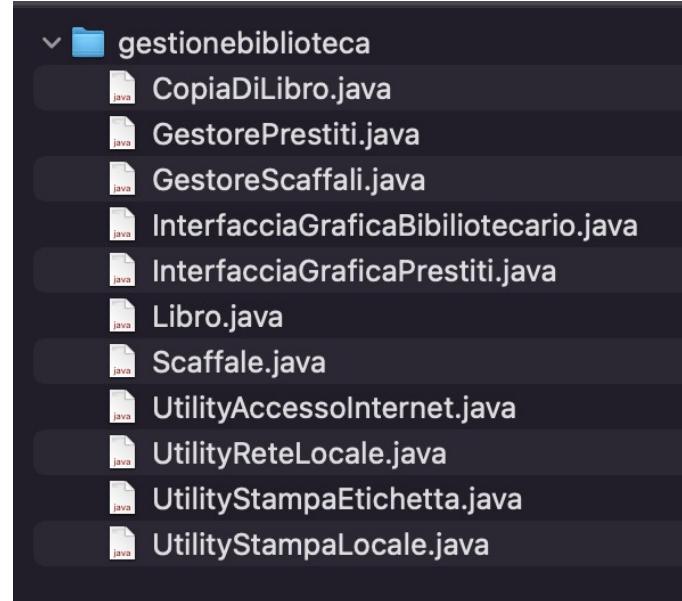
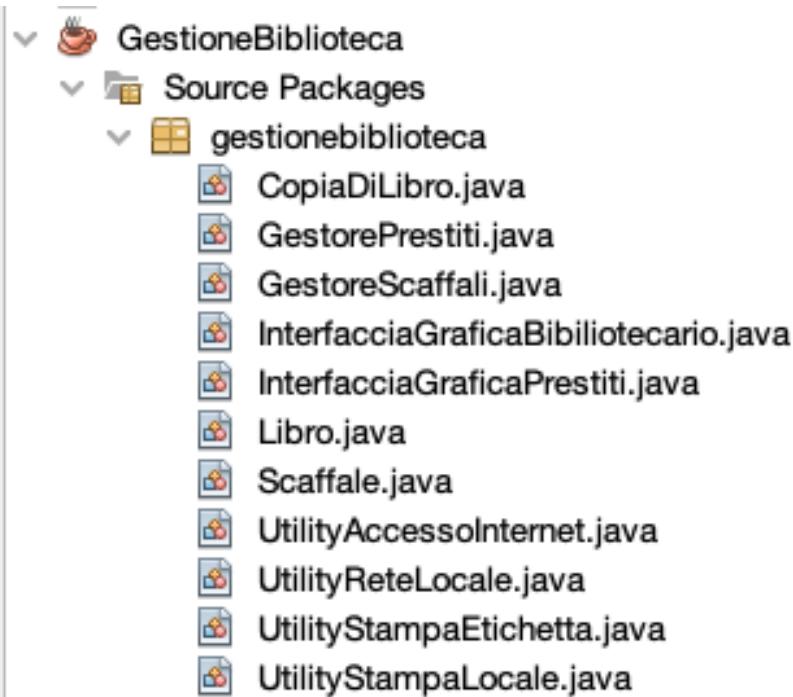
InterfacciaGraficaPrestiti

UtilityStampaEtichetta

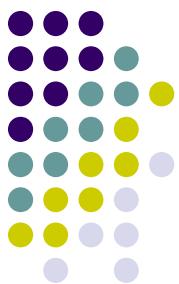
# Uso dei package per la gestione di sistemi di grandi dimensioni



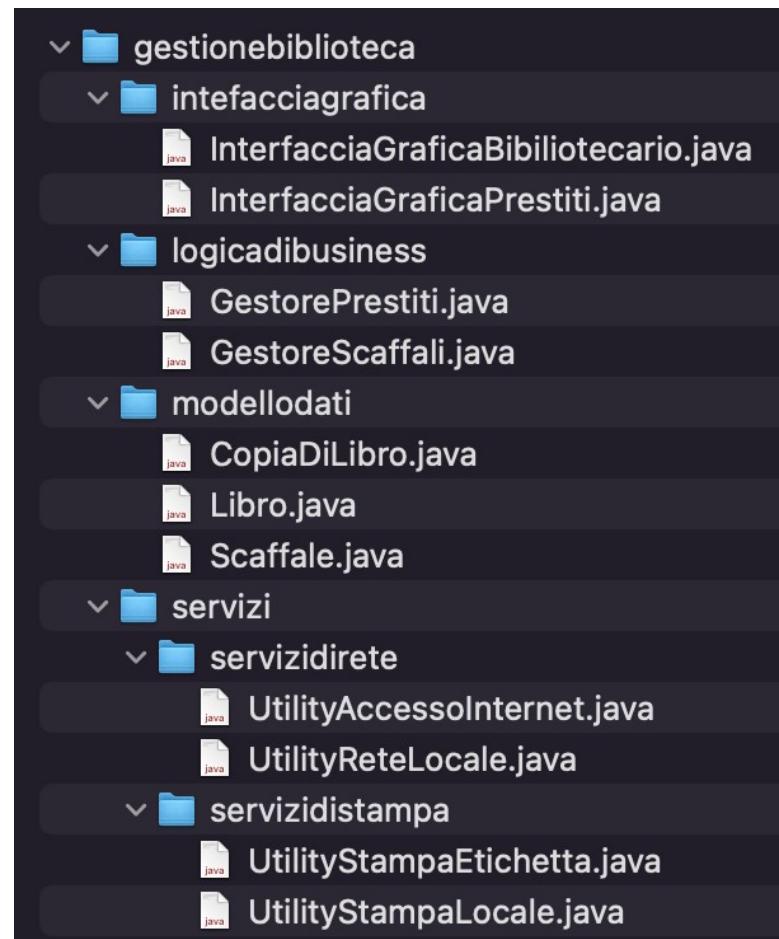
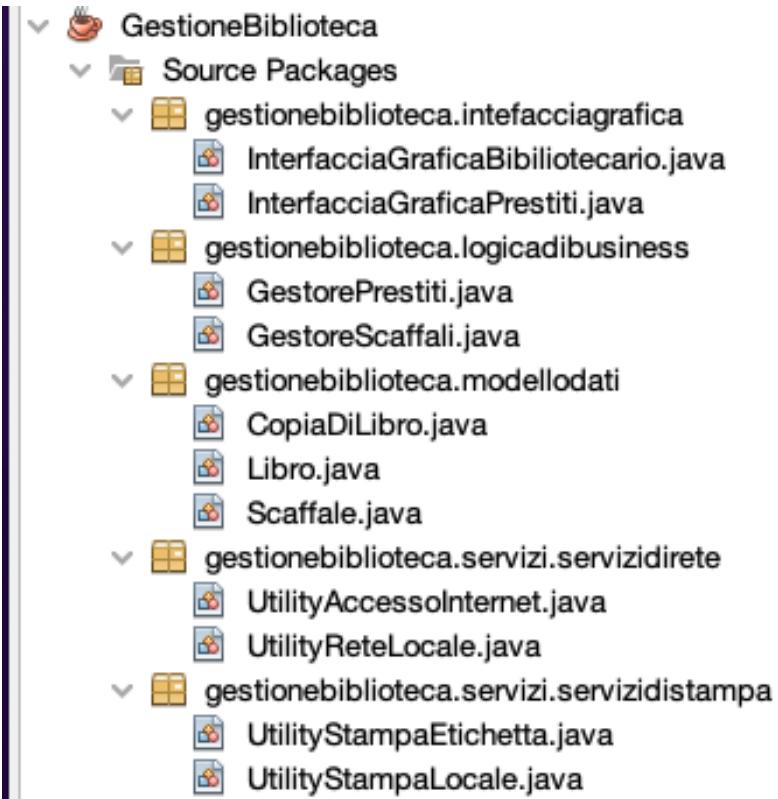
- Ecco come si presenta il progetto in Netbeans e su memoria di massa senza l'uso di package.



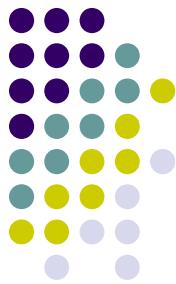
# Uso dei package per la gestione di sistemi di grandi dimensioni



- Ma se utilizzassimo i package per "organizzare" il progetto?



# Nome assoluto (qualificato) di una classe



- Il nome completo di una classe che fa parte di un package comprende il nome del package, ed è detto **nome assoluto o nome qualificato della classe**.
  - Se la classe C appartiene al package P il suo nome assoluto è **P.C**
  - Ad esempio persona.Persona o impiegato.Impiegato
  - In questo modo è possibile anche distinguere classi diverse con lo stesso nome che appartengono a diversi package perché i loro nomi assoluti sono diversi.



# Compilazione ed esecuzione

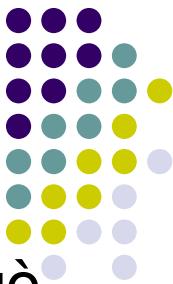
- Per compilare una classe C che fa parte di un package P si può:
  - porsi nella cartella superiore a P
  - lì invocare il **compilatore** con il **percorso** completo della classe: **javac P/C.java**
  - **Come visto in esecuzione:**
    - javac ./eseguibile/Unita17Package.java
- Per eseguire una classe E che fa parte di un package P occorre:
  - porsi nella cartella superiore a P
  - lì invocare l'**interprete** con il **nome assoluto della classe**: **java P.E**
  - **Come visto in esecuzione**
    - java eseguibile.Unita17Package



# Package di default

- Ogni classe in genere dichiara di appartenere ad un package
- In caso contrario è automaticamente assegnata al ***package di default***
  - Per convenzione, questo package fa riferimento alla cartella (directory) corrente
  - Si possono compilare ed eseguire i file nella cartella in cui si trovano, senza premettere percorsi o nomi assoluti

# Sistema dei nomi dei package



- I package possono essere innestati, cioè un package può contenere altri package. Ovviamemente questa organizzazione si deve rispecchiare nella organizzazione delle cartelle
  - Se il package P contiene il package P1 nella cartella P dovrà essere presente la cartella P1.
- Il sistema dei **nomi dei package** perciò è strutturato e sono possibili nomi di package come: P.P1 o java.awt.print
- Di conseguenza anche i nomi assoluti delle classi possono essere strutturati, ad esempio il nome assoluto della classe E che si trova nel package P1 sarà: P.P1.E



# Importazione dei nomi

- In Java ogni volta che si usa una classe è necessario denotarla con il suo nome assoluto!
- Se il nome assoluto è molto lungo questo sistema può risultare *molto* pesante...
- Per tale motivo si introduce il concetto di importazione di nome.

# Dichiarazione *import*



- Per evitare di specificare ogni volta il nome assoluto di una classe, si può importarlo:

**import** *nomeassoluto*;

- Ad esempio se la classe E si trova nel package P1 che a sua volta si trova in P:

**import** *P.P1.E*;

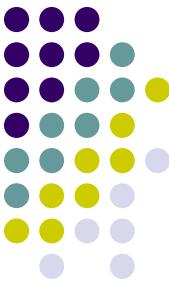
- In questo modo nel codice ci si può riferire alla classe semplicemente con E.
- E' possibile importare tutte le classi di un package in un colpo solo usando \*, ad esempio:

**import** *java.lang.\**

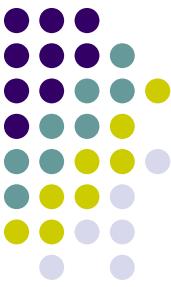
**import** *P.\**

**import** *P.P1.\**

# Quando può essere omessa l'importazione dei nomi?



- Alcuni packages sono importati implicitamente, ad esempio il package `java.lang` è implicitamente importato in ogni progetto Java e quindi si possono utilizzare classi come `System` e `String` senza dovere importare esplicitamente i loro nomi assoluti
- Inoltre ovviamente non è necessaria la dichiarazione di import se tutte le volte che viene usata la classe viene specificato il suo nome assoluto (ad esempio `java.util.Scanner`)



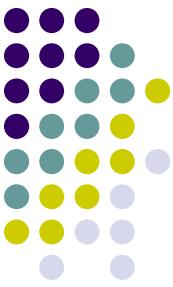
# Package e visibilità

- in Java esiste la **visibilità package**:
  - di default, una classe può essere utilizzata solo da altre classi del suo stesso package.
  - classi definite in *altri* package, non possono accedere a dati e metodi del package. Tuttavia è possibile dire che alcune classi possono essere visibili all'esterno: public.
  - Non è possibile definire una classe visibile in un solo file, la visibilità è a livello di package.
- NOTA: Il file in Java NON definisce un ambiente di visibilità come in C/C++



# Riepilogo modificatori in Java

	<b>default</b>	<b>private</b>	<b>protected</b>	<b>public</b>
same class	yes	yes	yes	yes
same package subclass	yes	no	yes	yes
same package non-subclass	yes	no	yes	yes
different package subclass	no	no	yes	yes
different package non-subclass	no	no	no	yes



# Package di Java

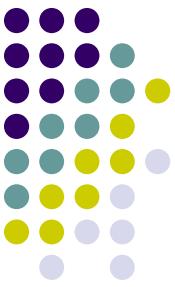
I package che iniziano con **java.** contengono classi che fanno parte delle Java API. Alcuni tra i principali package:

- **lang:** funzioni comuni del linguaggio
- **util:** altre funzioni di servizio
- **io:** input/output
- **text:** formattazione testi
- **awt, javax.swing:** grafica, interfacce grafiche
- **awt.event:** gestione eventi
- **applet:** programmi da eseguire in un browser
- **net:** gestione di rete



# Il package *java.lang*

- Il nucleo centrale dal linguaggio Java è definito nel package `java.lang`
- È sempre importato automaticamente:  
`import java.lang.*` è sottintesa
- Definisce i tipi primitivi e parte della classi di sistema
- Molte altre classi standard sono definite altrove: `java.awt`, `java.util`, `java.io`, ...



# Riferimenti

- Programmare in Java: §2.5.1, §3.2.5,  
§6.2, §6.8,