



Metodologie di Programmazione

Lezione 9: Riferimenti a oggetti, heap & stack

Lezione 9:

Sommario



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Riferimenti e oggetti
- Anatomia della memoria: stack & heap
- Campi statici
- Metodi statici
- Ancora sui package

Tipi di dato in Java: valori primitivi vs. oggetti

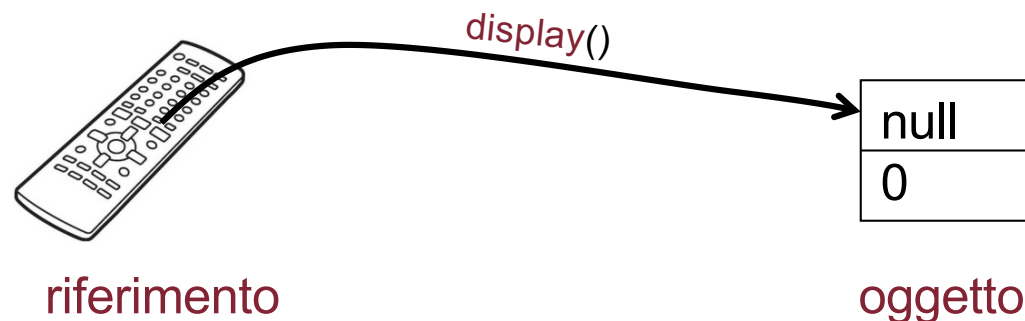


SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- E' importante **tenere a mente** la differenza tra:
 - **Valori** di tipo **primitivo** (int, char, boolean, float, double, ecc.)
 - **Oggetti** (istanze delle classi)
- La loro rappresentazione in memoria è differente:
 - **Valori primitivi**: memoria allocata automaticamente a tempo di compilazione
 - **Oggetti**: memoria allocata durante l'esecuzione del programma (operatore new)

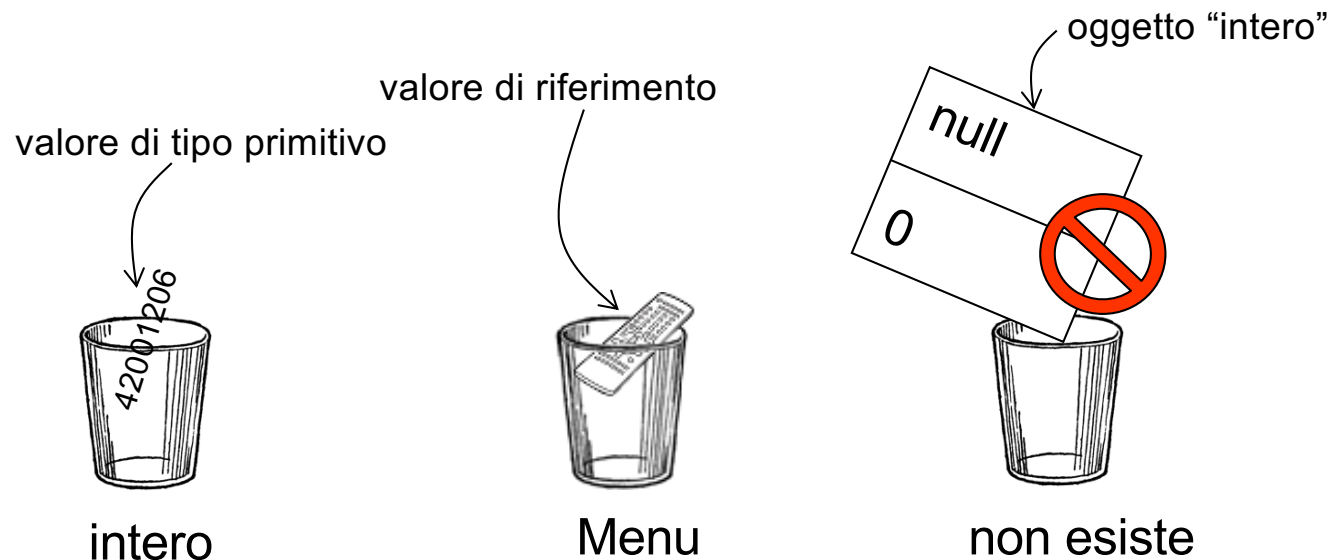
Riferimenti e oggetti

- I **riferimenti** sono ciò che è rimasto in **Java** dei puntatori (di linguaggi di più basso livello)
- Un **riferimento** è un **indirizzo di memoria**
 - Tuttavia non conosciamo il **valore numerico dell'indirizzo**
- **Assimilabili** ai tipi di dati primitivi
- Quindi gli oggetti non sono mai memorizzati direttamente nelle variabili, ma solo mediante il loro **riferimento**



Variabili di tipi primitivi e variabili “riferimento”

- Le **variabili** contengono:
 - valori di tipi di dati primitivi
 - oppure **riferimenti** a oggetti
- Non esistono variabili che **contengono oggetti**
 - Analogamente ai **puntatori in C**



Oggetti: i tre passi della dichiarazione, creazione e assegnazione

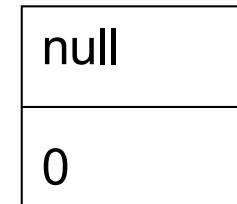
- **Dichiarazione:**

- Menu menu = new Menu();



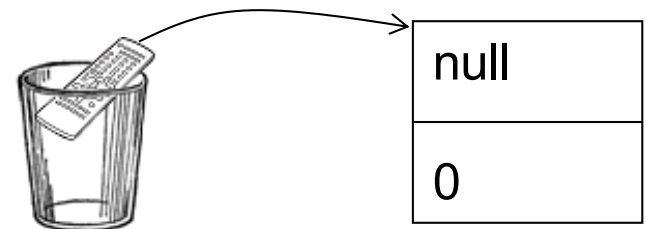
- **Creazione:**

- Menu menu = new Menu();



- **Assegnazione:**

- Menu menu = new Menu();



Anatomia della memoria: stack e heap

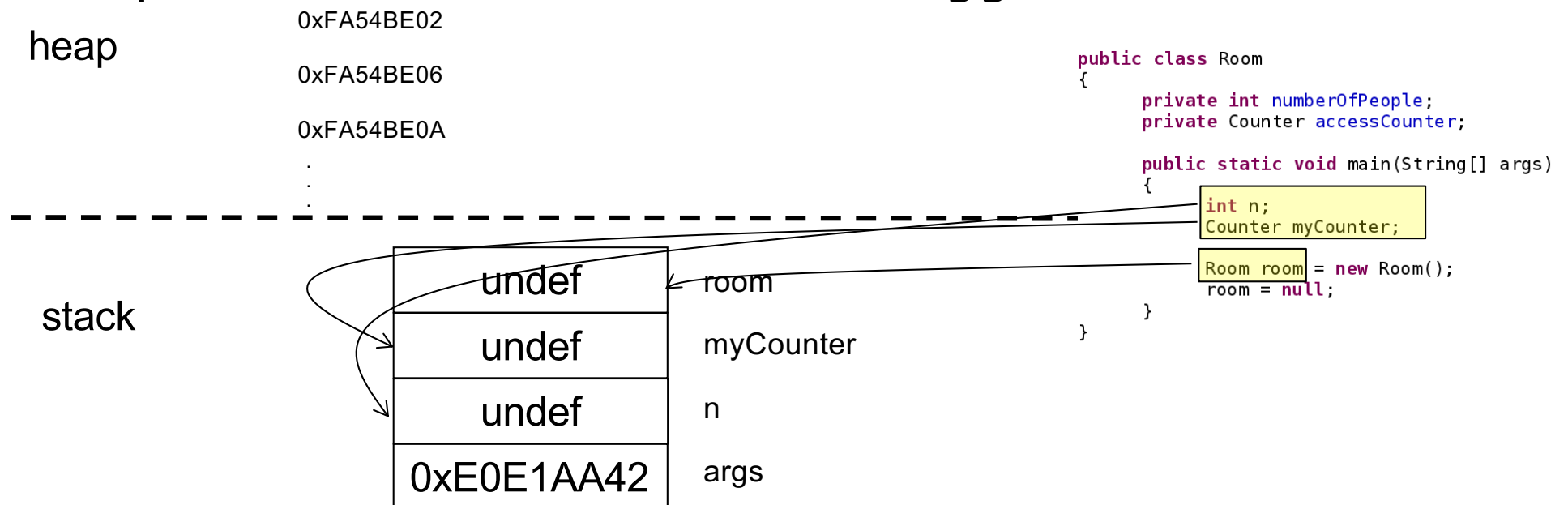


SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Esistono due tipi di memoria: lo **heap** e lo **stack**
- Sullo **stack** vanno le **variabili locali** e i **campi statici**
- Sullo **heap** vanno le aree di memoria allocate per la **creazione dinamica** di oggetti

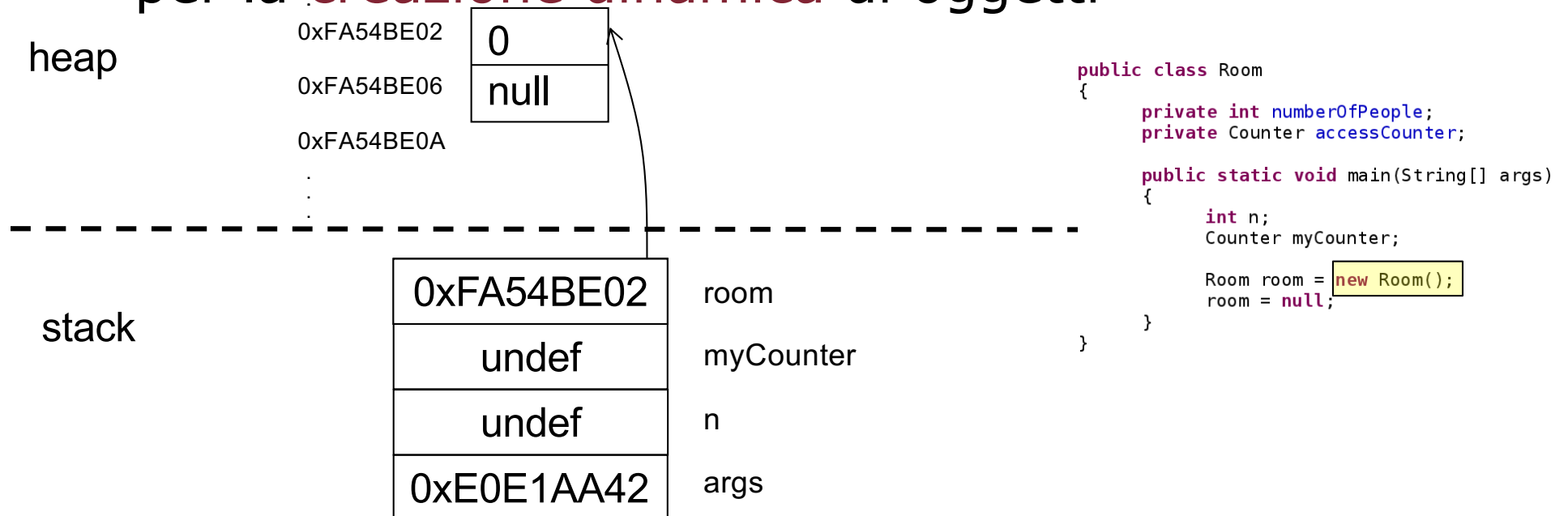
Anatomia della memoria: stack e heap

- Esistono due tipi di memoria: lo **heap** e lo **stack**
- Sullo **stack** vanno le **variabili locali** e i **campi statici**
- Sullo **heap** vanno le aree di memoria allocate per la **creazione dinamica** di oggetti



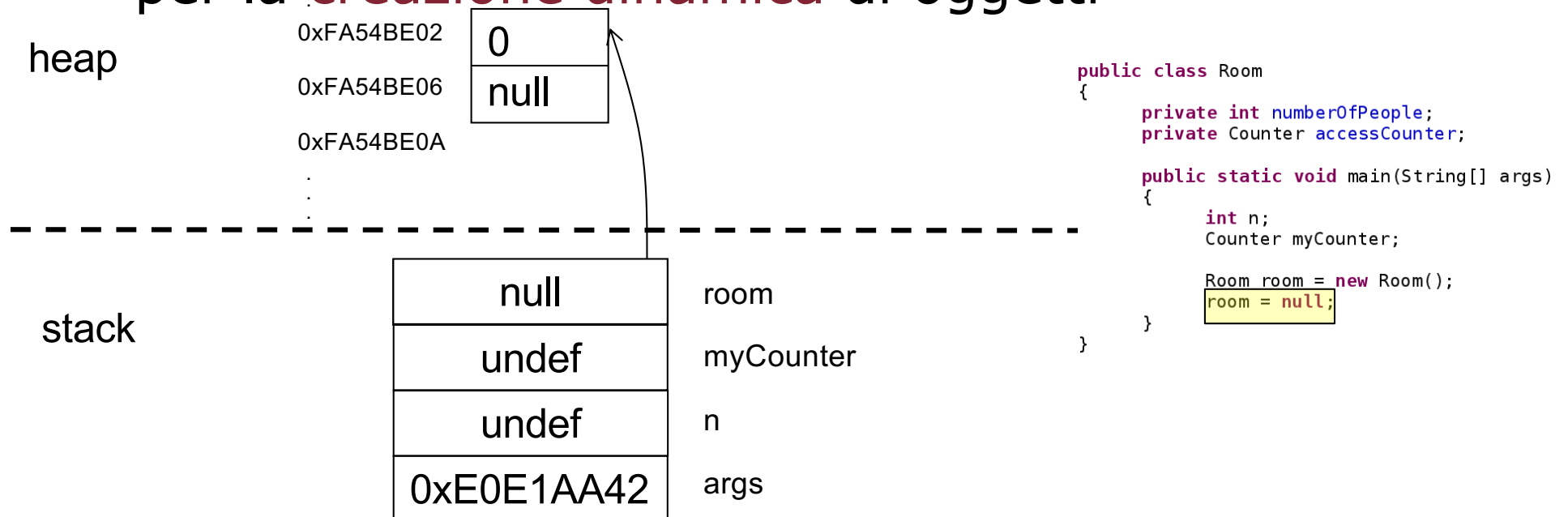
Anatomia della memoria: stack e heap

- Esistono due tipi di memoria: lo **heap** e lo **stack**
- Sullo **stack** vanno le **variabili locali** e i **campi statici**
- Sullo **heap** vanno le aree di memoria allocate per la **creazione dinamica** di oggetti



Anatomia della memoria: stack e heap

- Esistono due tipi di memoria: lo **heap** e lo **stack**
- Sullo **stack** vanno le **variabili locali** e i **campi statici**
- Sullo **heap** vanno le aree di memoria allocate per la **creazione dinamica** di oggetti



Campi di classe: la parola chiave static

- I campi di una classe possono essere dichiarati **static**
- Un campo static è relativo **all'intera classe**, **NON** al **singolo oggetto** istanziato
- Un campo static esiste in **una sola locazione di memoria**, allocata prima di qualsiasi oggetto della classe in una zona speciale di memoria nativa chiamata **MetaSpace** (pre-Java 8: PermGen)
- Viceversa, per ogni campo **non static** esiste **una locazione di memoria per ogni oggetto**, allocata a seguito dell'istruzione **new**

Esempio

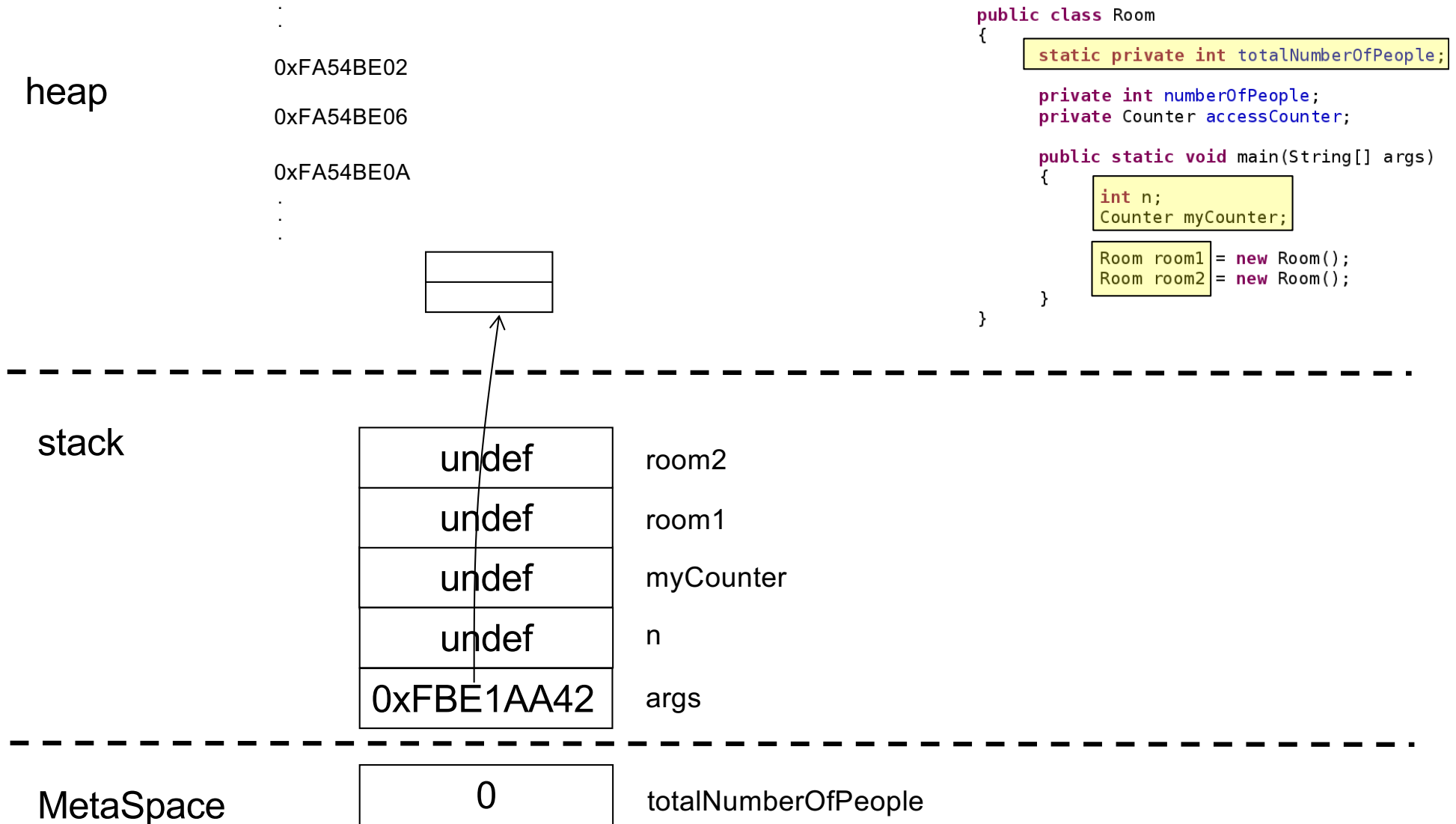
```
public class Room
{
    static private int totalNumberOfPeople;

    private int numberOfPeople;
    private Counter accessCounter;

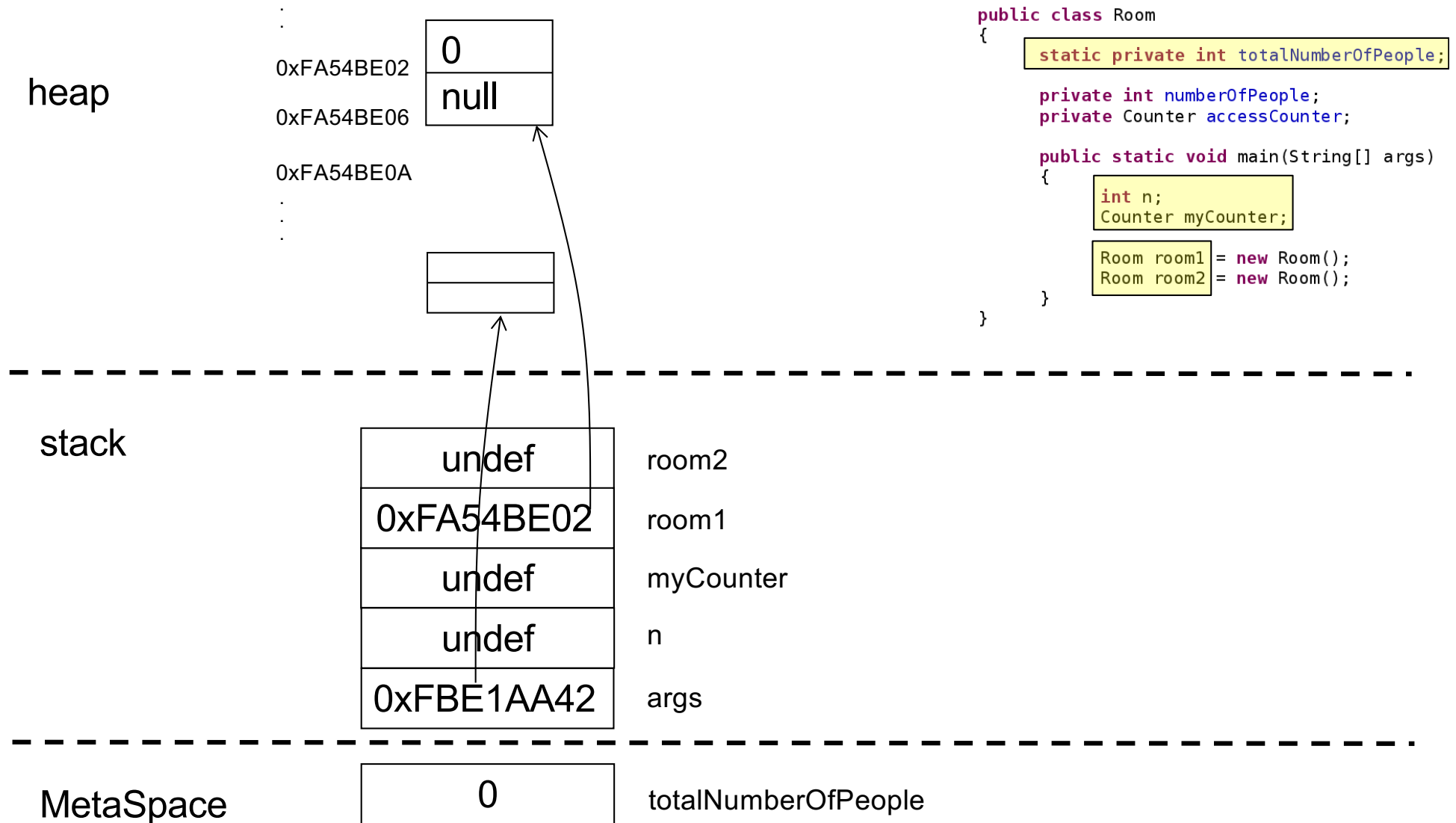
    public static void main(String[] args)
    {
        int n;
        Counter myCounter;

        Room room1 = new Room();
        Room room2 = new Room();
    }
}
```

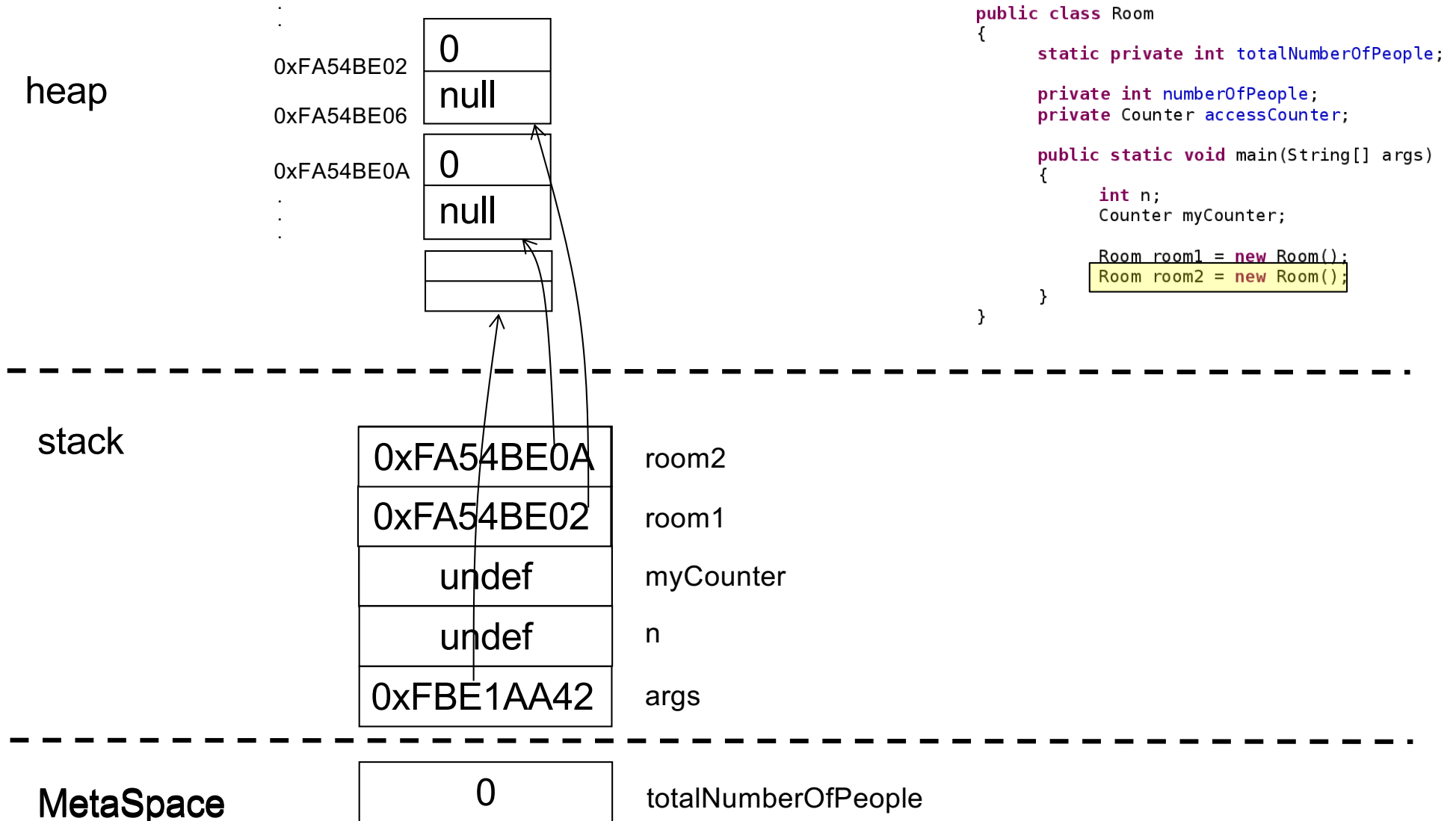
Anatomia della memoria (con campi statici)



Anatomia della memoria (con campi statici)



Anatomia della memoria (con campi statici)



Esercizio:

punti e segmenti

- Progettare una classe **Punto** per la rappresentazione di un **punto nello spazio tridimensionale**
- E una classe **Segmento** per rappresentare un **segmento nello spazio tridimensionale**
- Scrivere una **classe di test** che crei:
 - due oggetti della classe Punto con coordinate (1, 3, 8) e (4, 4, 7)
 - un oggetto della classe Segmento che rappresenti il segmento che unisce i due punti di cui sopra
- Raffigurare l'evoluzione dello stato della memoria, distinguendo tra **stack** e **heap**

Metodi statici

- I metodi **statici** sono **metodi di classe**
- **NON** hanno **accesso ai campi di istanza**
- Ma hanno **accesso ai campi di classe**

accesso da metodo
non statico
a campo statico

```
public class ContaIstanze
{
    static private int numberOfInstances;

    public ContaIstanze()
    {
        numberOfInstances++;
    }

    static public void main(String[] args)
    {
        new ContaIstanze();
        new ContaIstanze();
        new ContaIstanze();

        System.out.println("Numero di istanze create finora: "+numberOfInstances);
    }
}
```

accesso da metodo statico
a campo statico

Lettura dell'input da console

- Si effettua con la classe `java.util.Scanner`
- Costruita passando al costruttore lo stream di input (`System.in` di tipo `java.io.InputStream`)

```
public class ChatBotNonCosiInterattivo
{
    public static void main(String[] args)
    {
        // crea uno Scanner per ottenere l'input da console
        java.util.Scanner input = new java.util.Scanner(System.in);
        System.out.println("Come ti chiami?");

        // legge i caratteri digitati finche' non viene inserito
        // il carattere di nuova riga (l'utente preme invio)
        String nome = input.nextLine();

        System.out.println("Ciao "+nome+"!");
    }
}
```

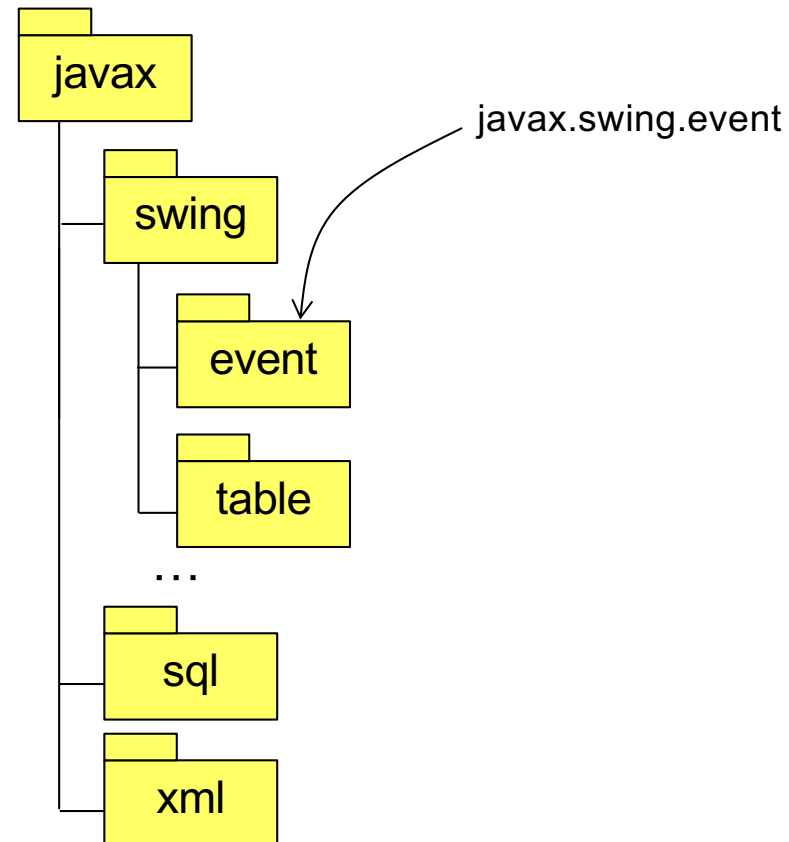
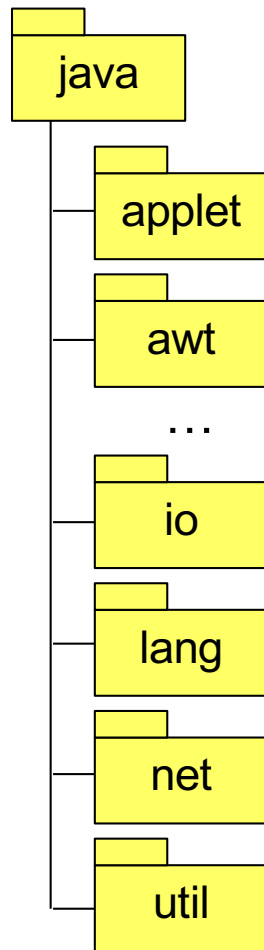
cos'è questo? →

Ancora sui package

- Le classi vengono inserite (categorizzate) in collezioni dette **package**
- Ogni package racchiude classi con **funzionalità correlate**
- Quando si utilizza una classe è necessario **specificarne il package** (come per **Scanner**, che appartiene al package `java.util`)
- Le classi che abbiamo usato finora (es. **System**, **String**) appartengono al package **speciale** `java.lang`
 - Questo package non deve essere specificato

Package standard

- Le **API (Application Programming Interface)** di Java sono organizzate in numerosi package



La dichiarazione import

- Per evitare di specificare il package di una classe ogni volta che viene usata, è sufficiente importare la classe

```
import java.util.Scanner;
```

```
public class ChatBotNonCosiInterattivo
{
    public static void main(String[] args)
    {
        // crea uno Scanner per ottenere l'input da console
        Scanner input = new Scanner(System.in);

        System.out.println("Come ti chiami?");

        // leggere i caratteri digitati finche' non viene inserito
        // il carattere di nuova riga (l'utente preme invio)
        String nome = input.nextLine();

        System.out.println("Ciao "+nome+"!");
    }
}
```

La dichiarazione import

- Per evitare di specificare il package di una classe ogni volta che viene usata, è sufficiente importare la classe
- O l'intero package:

```
import java.util.*;
```

Attenzione: non è ricorsivo!

```
public class ChatBotNonCosiInterattivo
{
    public static void main(String[] args)
    {
        // crea uno Scanner per ottenere l'input da console
        Scanner input = new Scanner(System.in);

        System.out.println("Come ti chiami?");

        // leggere i caratteri digitati finche' non viene inserito
        // il carattere di nuova riga (l'utente preme invio)
        String nome = input.nextLine();

        System.out.println("Ciao "+nome+"!");
    }
}
```

Creazione di nuovi package



SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- I **package** sono rappresentati fisicamente da **cartelle** (**String.class** si trova sotto **java/lang/**)
- Una classe può essere inserita in un determinato **package** semplicemente
- specificandolo all'inizio del file (parola chiave **package**)
- posizionando il file nella corretta sottocartella
- **Eclipse** fa tutto questo per voi!

Esempio: la classe Triangolo

```
package it.navigli.geometry;

/**
 * La figura geometrica triangolo
 * @author navigli
 */
public class Triangolo
{
    /**
     * Base del triangolo
     */
    private double base;

    /**
     * Altezza del triangolo
     */
    private double altezza;

    /**
     * Costruttore del triangolo
     * @param base base del triangolo
     * @param altezza altezza del triangolo
     */
    public Triangolo(double base, double altezza)
    {
        this.base = base;
        this.altezza = altezza;
    }

    /**
     * Restituisce l'area della figura
     * @return l'area
     */
    public double getArea()
    {
        return base*altezza/2.0;
    }
}
```

Triangolo.java si deve
trovare nella cartella
it/navigli/geometry

Commento Javadoc per la classe

Javadoc per un campo

Javadoc per il costruttore

Parametri Javadoc per il costruttore

Javadoc per un metodo

Riferimento all'oggetto costruito

Javadoc per il valore restituito