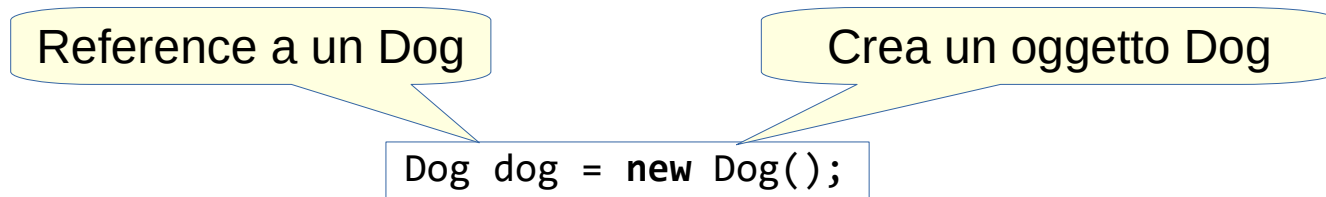


Java SE: OOP

- Classi e oggetti – dati e funzionalità
- Costruttori e static factory method
- Wrapper di primitivi
- Variabili *member* e locali
 - Inizializzazione
- Progetto di riferimento
 - <https://github.com/egalli64/jse> (*modulo 5*)

Classi e oggetti

- Classe:
 - Definita in un package, normalmente in un file che ha il suo stesso nome (.java)
 - Descrive un **nuovo tipo di dato**, che ha sue proprietà e metodi
 - l'accesso ai membri di una classe è indicato con l'operatore di dereferenziazione, il punto "."
 - Tipicamente sono nomi usati per descrivere il problema che si vuole risolvere
- Oggetto
 - Istanza di una classe, che è il suo modello di riferimento



Constructor (ctor)

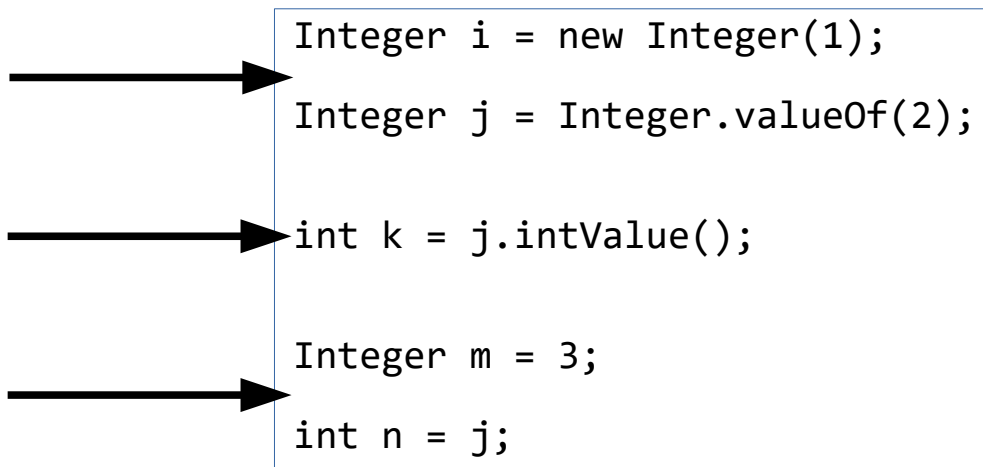
- Metodo speciale, usato per inizializzare lo stato del nuovo oggetto
 - Invocato in seguito alla creazione di un oggetto via operatore **new**
 - Ha lo stesso nome della classe, non ha un return type (nemmeno void)
- Ogni classe può avere svariati costruttori
 - Devono essere distinguibili in base al numero/tipo dei parametri (*overload*)
- Per convenzione, i parametri hanno lo stesso nome delle proprietà che inizializzano
 - Uso della keyword **this** per risolvere l'ambiguità
- Se una classe non ha ctor, Java ne crea uno di default
 - Senza parametri, non fa niente
- In Java non esiste un distruttore (dctor)
 - L'eliminazione di un oggetto dallo heap è responsabilità del Garbage Collector (gc)

Static Factory Method

- Approccio alternativo e più flessibile al costruttore
 - Può avere un nome significativo, ad esempio
 - `My.from(Other)` → crea un `My` from un `Other`
 - `My.getInstance()` → ritorna la singola istanza di una classe
 - ...
 - Può creare un oggetto del tipo richiesto, o derivato, o altro
 - Può incapsulare i passi preparatori alla creazione
 - Permette un maggior controllo sulla creazione

Wrapper di primitivi

- Controparte reference dei tipi primitivi
 - Boolean, Character, Byte, Short, Integer, Long, Float, Double
- Boxing esplicito
 - Costruttore (deprecato da Java 9)
 - Static factory method
- Unboxing esplicito
 - Metodi definiti nel wrapper
- Auto-boxing
- Auto-unboxing



```
Integer i = new Integer(1);  
Integer j = Integer.valueOf(2);  
  
int k = j.intValue();  
  
Integer m = 3;  
int n = j;
```

Alcuni metodi statici dei wrapper

- Boolean
 - **valueOf**(boolean) // reference
 - valueOf(String)
 - **parseBoolean**(String) // primitive
- Integer
 - parseInt(String)
 - toHexString(int)
- Double
 - isNaN(double)
- Character
 - isDigit(char)
 - isLetter(char)
 - isLetterOrDigit(char)
 - isLowerCase(char)
 - isUpperCase(char)
 - toUpperCase(char)
 - toLowerCase(char)

Lo “scope” delle variabili

- Vita limitata al blocco che le contiene
- Member (field, property)
 - di istanza (default)
 - stato dell'oggetto
 - di classe (static)
- Locali (automatiche)
 - Esistenza limitata a un metodo o a un blocco interno
 - Caso particolare, la variabile di ciclo nel loop for, definita subito prima del blocco relativo
- Una variabile locale non può nascondere un'altra locale. Potrebbe però nascondere una proprietà (ma non si fa!)

```
public class Scope {  
    private static int staticMember = 5;  
    private long member = 5;  
  
    public void f() {  
        long local = 7;  
        if(staticMember == 2) {  
            float local = 0.0F;  
            short inner = 12;  
            staticMember = 1 + inner;  
            member = 3 + local;  
        }  
    }  
  
    public static void main(String[] args) {  
        double local = 5;  
        System.out.println(local);  
        staticMember = 12;  
    }  
}
```

Inizializzazione delle variabili

- Finché non viene inizializzata una variabile non può essere usata – errore di compilazione →

```
int j;  
System.out.println(j);
```

- Esplicita per assegnamento (preferita)

- primitivi: diretto
- reference: via operatore new

```
int i = 42;  
int[] data = new int[10];
```

- Implicita by default (solo member)

- primitivi
 - numerici: 0
 - boolean: false
- reference: null

```
private int i;           // 0  
private boolean flag;   // false  
private String t;       // null
```