

Java SE

- Struttura del codice
- Variabili
 - Tipi di dato: primitivi e reference
 - Stack e heap
 - Array e stringhe
- Progetto di riferimento
 - <https://github.com/egalli64/jse> (*modulo 1*)

Struttura del codice /1

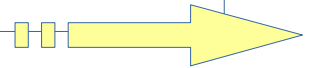
- Dichiarazioni

- **package**
 - Gruppo (omogeneo) a cui appartiene la classe
- **import**
 - Indica l'uso di classi definite in altri package
 - Eccezione, java.lang non richiede di essere importato
- **class**
 - Una sola "public" per file sorgente

- Commenti

- Multi-line
- Single-line
- Javadoc-style

```
/*  
 * A simple Java source file  
 */  
package m01.s02;  
  
import java.lang.Math; // not required  
  
/**  
 * A "hello world" class  
 *  
 * @author Emanuele Galli  
 */  
public class Simple {  
    public static void main(String[] args) {  
        System.out.println(Math.PI);  
    }  
}  
  
class PackageClass {  
    // TODO: Not implemented (yet)  
}
```



Struttura del codice /2

- Parentesi
 - Graffe
 - Blocchi, body di classi e metodi
 - Tonde
 - Qui identificano metodi
 - Per la definizione – `main()` – lista dei parametri
 - Per l'invocazione – `println()` – lista degli argomenti
 - Quadre
 - Identificano array
- Punto e virgola
 - Obbligatorio per indicare il termine di uno statement

```
public class Simple {  
    public static void main(String[] args) {  
        System.out.println(Math.PI);  
    }  
}
```

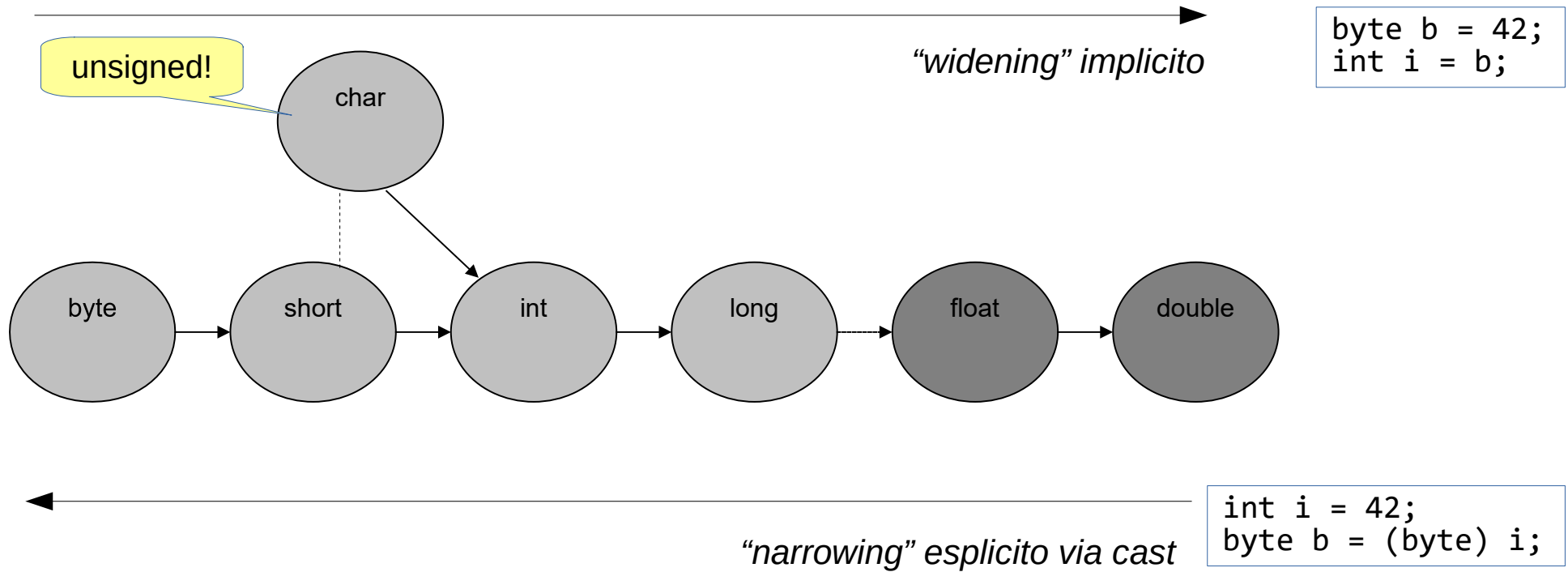
Variabili e tipi di dato

- Variabile
 - Locazione di memoria
 - Nome usato per accedere il dato nella memoria
 - Case sensitive, non tutti i caratteri sono utilizzabili per un identificatore
- Tipo di dato
 - Determina il valore della variabile e le operazioni disponibili su di essa
 - In Java ci sono due famiglie di tipi di dato: **primitivi**, **reference** (class / interface)
 - Da **Java 10**, si può lasciare dedurre dal compilatore il tipo delle variabili locali → **var**
- Costanti
 - Tipo prefissato dalla keyword **final**, naming convention: UPPERCASE_UNDERSCORE

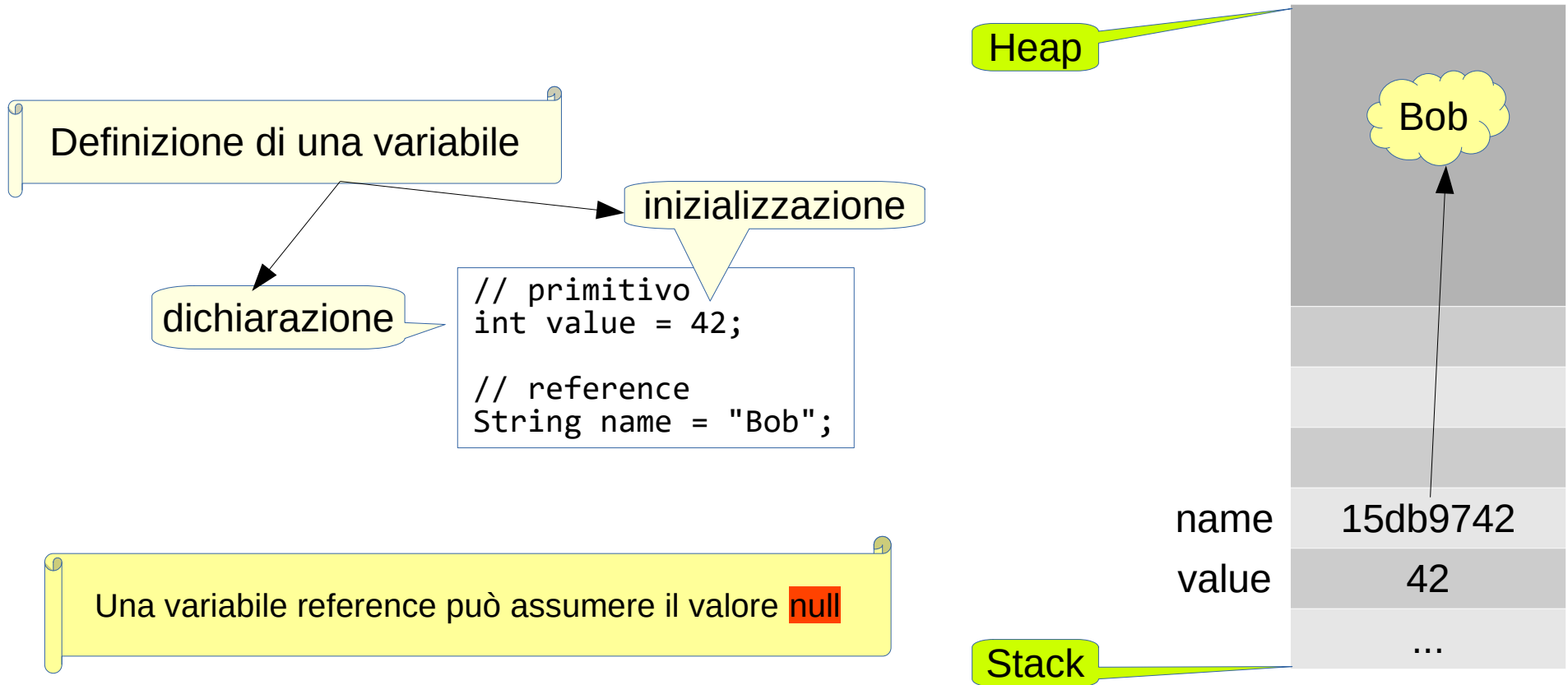
Tipi primitivi

| bit | | | signed integer | floating point IEEE 754 |
|------|---------|----------------------|----------------|-------------------------|
| 1(?) | boolean | false true | | |
| 8 | | | byte | |
| | | | | |
| 16 | char | '\u0000' '\uFFFF' | short | |
| | | | | |
| 32 | | | int | float |
| | | | | |
| 64 | | | long | double |
| | | | | |

Cast tra primitivi



Primitivi vs Reference



Array

- Sequenza indicizzata – base 0 – di valori tutti dello stesso tipo (primitivo o reference), memorizzati nello **heap**.
- La sua dimensione è definita al momento della creazione, e non può più essere cambiata
- È un reference, ma non implementa metodi suoi, ha solo la proprietà (readonly) **length**
- Tentativo di accedere a un elemento esterno → **ArrayIndexOutOfBoundsException**
- Metodi di utilità nella classe **Arrays**: copyOf(), sort(), fill(), equals(), toString(), deepToString(), ...

```
int[] anArray = new int[12];  
anArray[0] = 7;  
  
int value = anArray[5];  
// value = anArray[12]; // exception
```

```
int[] data = { 1, 4, 3 };  
  
// data[data.length] = 21; // exception  
System.out.println(array.length); // 3
```

```
int[][] matrix = new int[4][5];  
  
int value = matrix[2][3];
```

| | | | | |
|--------|--------|--------|--------|--------|
| [0][0] | [0][1] | [0][2] | [0][3] | [0][4] |
| [1][0] | [1][1] | [1][2] | [1][3] | [1][4] |
| [2][0] | [2][1] | [2][2] | [2][3] | [2][4] |
| [3][0] | [3][1] | [3][2] | [3][3] | [3][4] |

String

- Un singolo carattere è rappresentato dal primitivo **char**
- **String** è una classe, un reference, le sue istanze sono oggetti nello heap
 - rappresenta una sequenza immutabile di caratteri
- **StringBuilder**, mutabile, è usata per creare stringhe complesse
 - Come indica il nome, implementa il pattern Builder

```
char c = 'x';  
String s = new String("hello");  
String t = "hello";
```

Forma standard (ma crea due oggetti!)

Forma semplificata (preferita!)