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

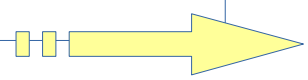
- Dichiarazioni
  - package
    - Gruppo (omogeneo) a cui appartiene la classe
  - import
    - Indica l'uso di classi definite in altri package
    - Eccezione, java.lang non richiede di essere importato
  - class
    - Una sola "public" per file sorgente
- Commenti
  - Multi-line
  - Single-line
 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

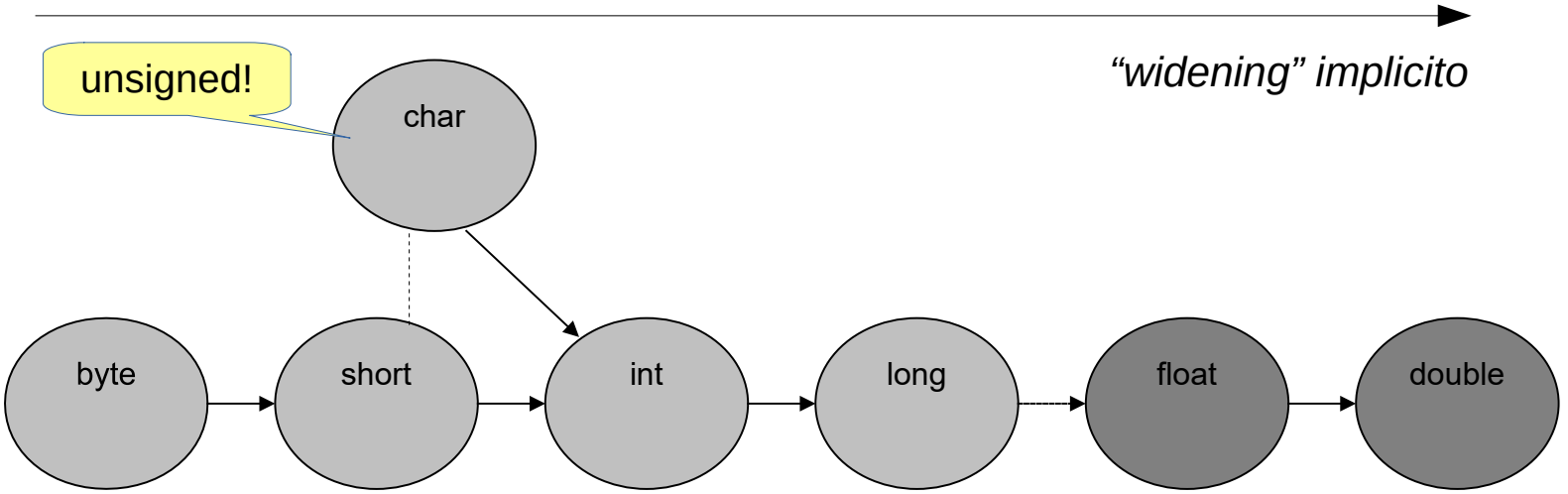
Variabili e tipi di dato

- 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

<table>
<tr><th>bit</th><th colspan="2"></th><th colspan="2">signed integer</th><th>floating point IEEE 754</th></tr>
<tr><td rowspan="2">1(?)</td><td rowspan="2">boolean</td><td>false</td><td colspan="2" rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>true</td></tr>
<tr><td rowspan="2">8</td><td colspan="2" rowspan="2"></td><td rowspan="2">byte</td><td>-128</td><td rowspan="2"></td></tr>
<tr><td>127</td></tr>
<tr><td rowspan="2">16</td><td rowspan="2">char</td><td>'\u0000'</td><td rowspan="2">short</td><td>-32,768</td><td rowspan="2"></td></tr>
<tr><td>'\uFFFF'</td><td>32,767</td></tr>
<tr><td>32</td><td colspan="2" rowspan="2"></td><td rowspan="2">int</td><td>-2^31</td><td rowspan="2">float F</td></tr>
<tr><td>64</td><td>2^31 - 1</td></tr>
<tr><td></td><td colspan="2"></td><td>long L</td><td>-2^63</td><td rowspan="2">double</td></tr>
<tr><td></td><td colspan="2"></td><td></td><td>2^63 - 1</td></tr>
</table>

Cast tra primitivi




```
byte b = 42;
int i = b;
```

```
int i = 42;
byte b = (byte) i;
```

Primitivi vs Reference

Definizione di una variabile

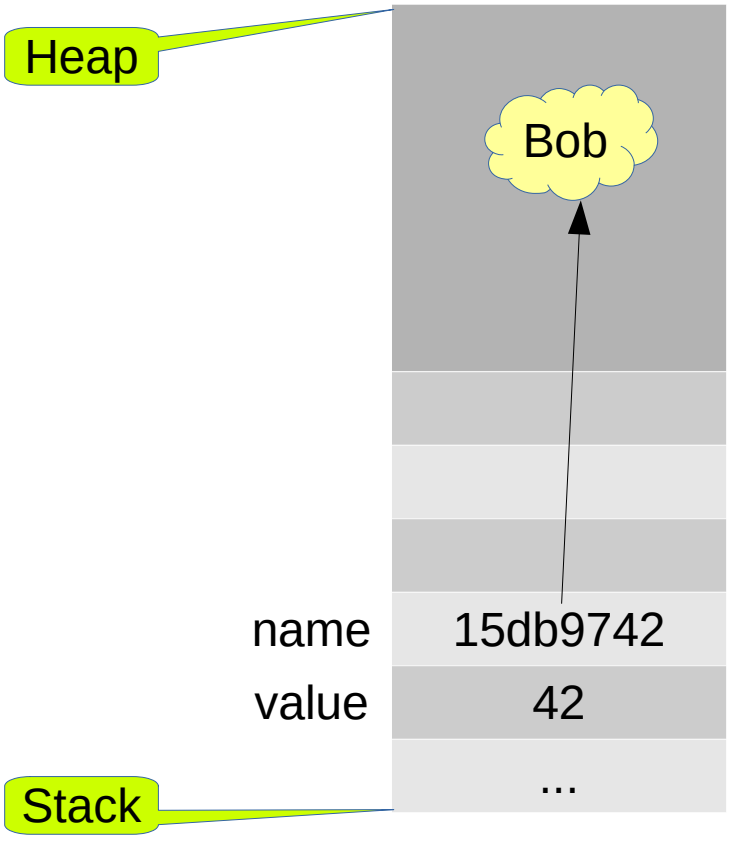
dichiarazione

inizializzazione

```
// primitivo
int value = 42;

// reference
String name = "Bob";
```

Una variabile reference può assumere il valore null



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!)