Laboratorio di Programmazione

Lezione 11 – Java 101

Marco Anisetti (teoria)
Dipartimento di Informatica
marco.anisetti@unimi.it

Matteo Luperto (lab. turno A)
Dipartimento di Informatica
matteo.luperto@unimi.it

Nicola Bena (lab. turno B)
Dipartimento di Informatica
nicola.bena@unimi.it

Hello World!

```java
public class Hello {
	public static void main(String[] args) {
		// Stampa una riga e va a capo.
		System.out.println("Hello, World!");
	}
}
```

- Creare una cartella per ogni esercizio
- Creare una classe **`ClassName`**`.java` per ogni classe **`ClassName`**
- Creare una classe contenente il `main`
- (Pre)compilare la classe con il `main`: `javac` **`ClassName`**`.java`
- Eseguire il bytecode `.class` precompilato: `java` **`ClassName`**

Hello World!

```
public class Hello {
	public static void main(String[] args) {
		// Stampa una riga e va a capo.
		System.out.println("Hello, World!");
	}
}
```

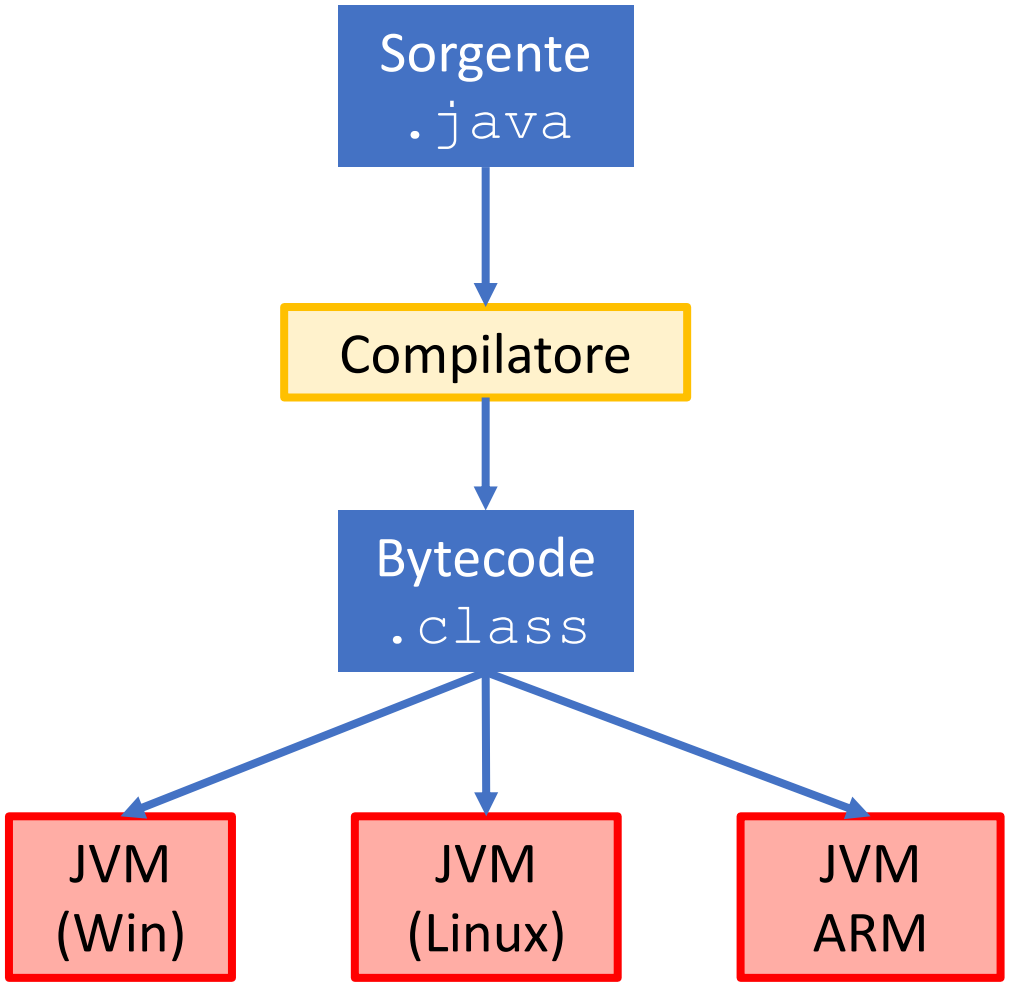
- Creare una cartella per ogni esercizio
- Creare una classe `Hello.java`
- Creare una classe contenente il `main`
- (Pre)compilare la classe con il `main`: `javac Hello.java`
- Eseguire il bytecode `.class` precompilato: `java Hello`

Hello World!

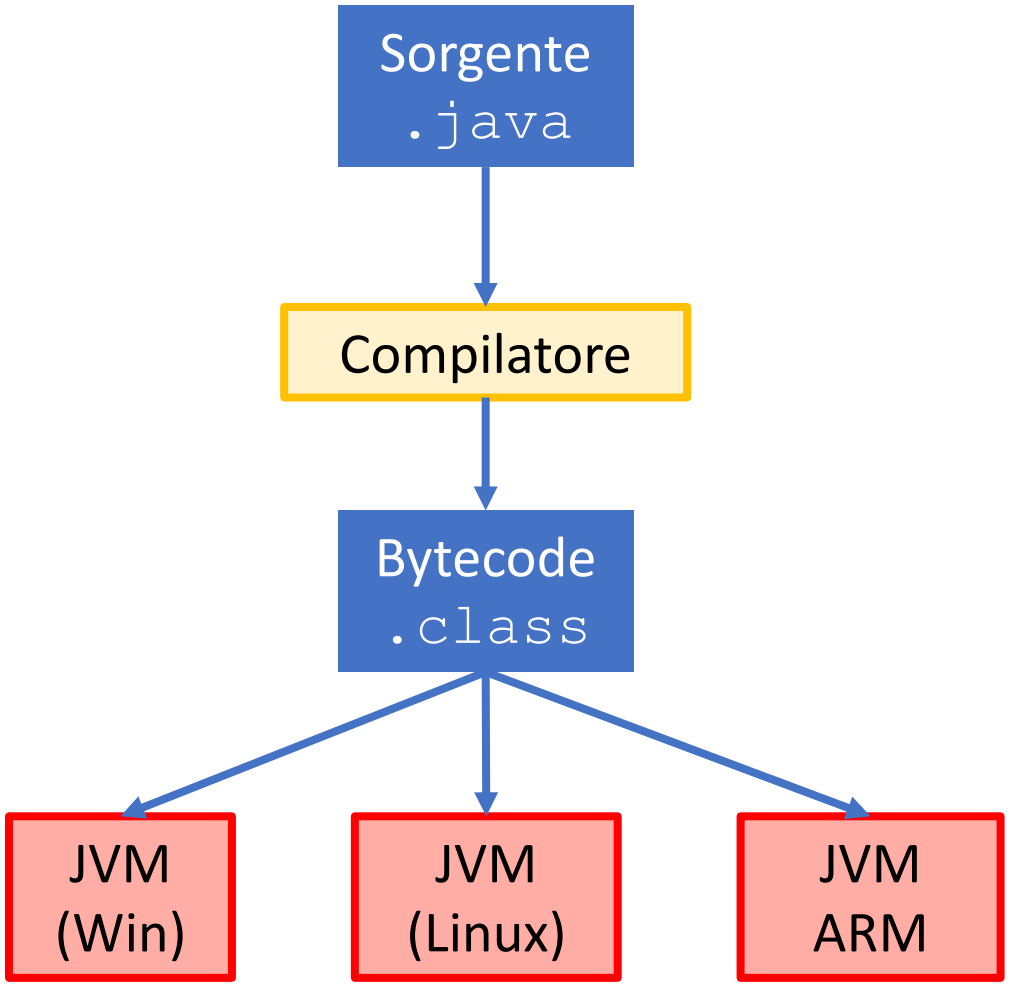
```
public class Hello {
	public static void main(String[] args) {
		// Stampa una riga e va a capo.
		System.out.println("Hello, World!");
	}
}
```

- Creare una cartella per ogni esercizio
- Creare una classe `Hello.java`
- Creare una classe contenente il `main`
- (Pre)compilare la classe con il `main`: `javac Hello.java`
- Eseguire il bytecode `.class` precompilato: `java Hello`

Compilazione



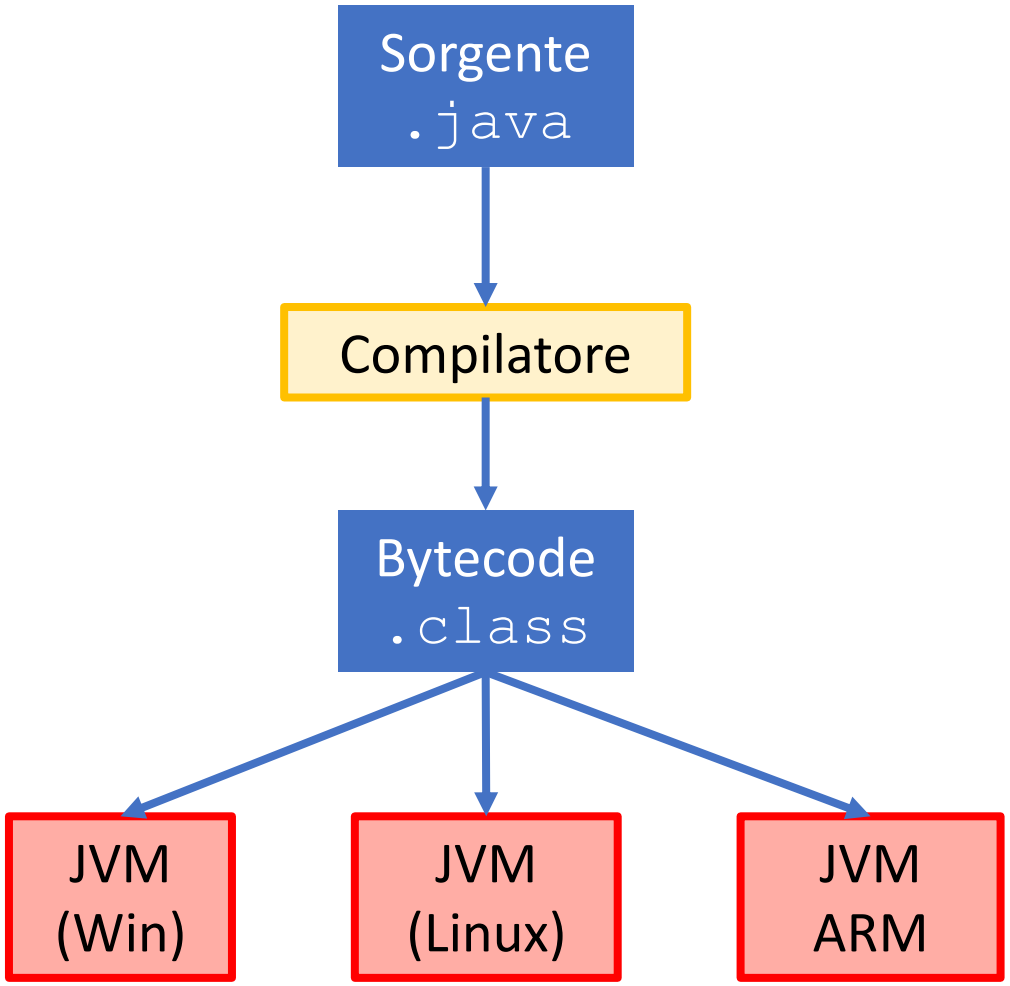
Compilazione

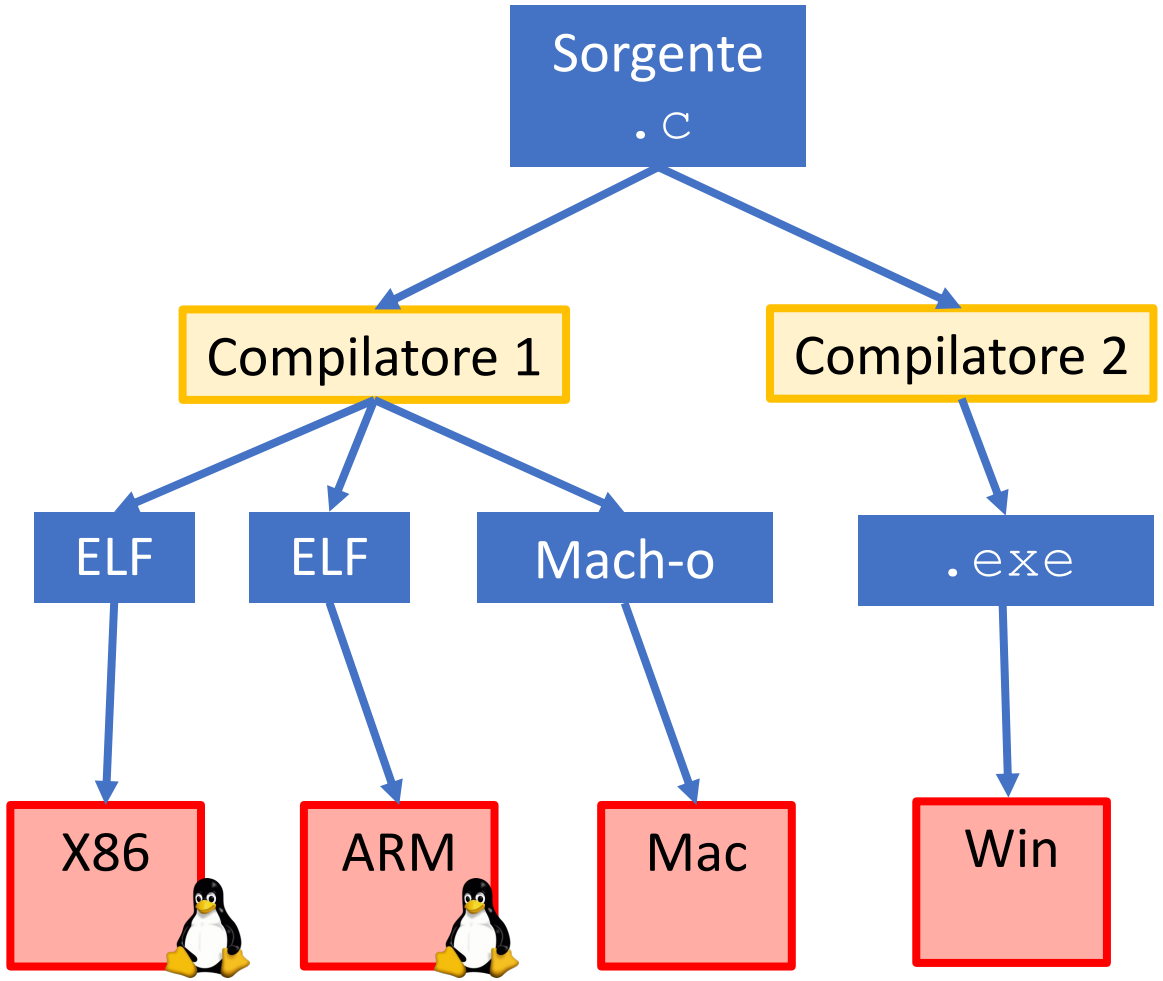



```
javac MyCass.java
```

```
java MyClass
```

Compilazione





Hello World!

```java
public class Hello {
	public static void main(String[] args) {
		// Stampa una riga e va a capo.
		System.out.println("Hello, World!");
	}
}
```

Hello World!

```java
public class Hello {
        public static void main(String[] args) {
                // Stampa una riga e va a capo.
                System.out.println("Hello, World!");
        }
}
```

Dichirazione della classe (`Hello`)

Dichiarazione della funzione `main`

- Non restituisce nulla (`void`)
- Riceve un array di `String`

Hello World!

```java
public class Hello {
        public static void main(String[] args) {
                // Stampa una riga e va a capo.
                System.out.println("Hello, World!");
        }
}
```

Dichirazione della classe (`Hello`)

Dichiarazione della funzione `main`

- Non restituisce nulla (`void`)
- Riceve un array di `String`

```c
#include <stdio.h>

int main(int argc, char*argv[]){
        printf("Hello, World!");
        return 0;
}
```

Java 101

- Siate ordinati: una cartella per ogni Progetto
- Non serve (in questo corso) usare *package* e costrutti simili
- Potete programmare con un editor di testo (o un IDE più avanzato)
- Tutti i file `*.java` contenuti nella stessa cartella sono visibili al momento della compilazione
  - Il nome della classe deve corrispondere al nome del file
  - `Hello.java` → `class Hello`
  - ~~`hello.java` → `class Hello`~~
- Il nome della classe viene scritto in `CamelCase`
- Il nome di variabili e funzioni/metodi viene scritto in `camelCase`

```java
import java.util.Scanner; // Import the Scanner class
// Per compilare: javac Hello.java -> crea i file.class
// Per eseguire: java Hello
public class Hello {

	public static void main(String[] args) {
		// Stampa una riga e va a capo.
		System.out.println("Hello, World!");
		// Stampa una riga e non va a capo).
		System.out.print("Hello, World!");
		// Stampa una riga con formattazione.
		System.out.printf("Hello, %s!\n", "World");

		// Creiamo un oggetto scanner
		Scanner scan = new Scanner(System.in);
		// Leggi un intero
		System.out.println("Ora inserisci un intero.");
		int i = scan.nextInt();
		System.out.printf("Hai inserito, %d.\n", i);
		// Chiudiamo scanner quando non serve più
		scan.close();

	}
}
```

Java 101

```java
public class Main {
  public static int calcolaSommatoria(int a) {
        int result = 0;
        for (int i = 1; i <= a; i++){
              result += i;
        }
        return result;
  }
  public static void main(String[] args){
        int a=10, sommatoria;
        sommatoria = calcolaSommatoria(a);
        System.out.printf("La sommatoria di %d è %d\n", a, sommatoria);
  }
}
```

Java 101

```
public class Main {
    public static int calcolaSommatoria(int a) {
        int result = 0;
        for (int i = 1; i <= a; i++){
            result += i;
        }
        return result;
    }
    public static void main(String[] args){
        int a=10, sommatoria;
        sommatoria = calcolaSommatoria(a);
        System.out.printf("La sommatoria di %d è %d\n", a, sommatoria);
    }
}
```

Tutto il codice dev'essere racchiuso all'interno di una *classe* (capiremo perché)

Java 101

Le funzioni devono essere dichiarate `public static` (poi capiremo perché)

```
public class Main {
  public static int calcolaSommatoria(int a) {
        int result = 0;
        for (int i = 1; i <= a; i++){
              result += i;
        }
        return result;
  }
  public static void main(String[] args){
        int a=10, sommatoria;
        sommatoria = calcolaSommatoria(a);
        System.out.printf("La sommatoria di %d è %d\n", a, sommatoria);
  }
}
```

Java è un linguaggio C-like: segue una sintassi simile a C

- Definizione funzioni
- Dichiarazione variabili
- Selezione e cicli (`if`, `while`, `for`, `switch`)
- Chiamate a funzioni

Java 101: Variabili

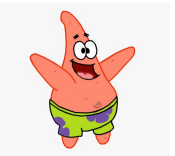
- Tipi semplici: `int`, `long`, `float`, `double`, `boolean`, `char`, …
- Tipi strutturati: next week
- Non si può accedere a variabili locali prima della loro inizializzazione

```
int i;
i = i + 1;
```

Anche in C non posso farlo perché è undefined behavior, ma il compilatore me lo permette senza segnalarlo

```
❯ javac Example01.java
Example01.java:4: error: variable i might not have been initialized
        i = i + 1;
            ^
1 error
```

Un codice con questo errore non può nemmeno essere compilato!



Java 101: Selezione e Cicli

```java
boolean result;
if (a >= 0) {
    result = true;
} else  {
    result = false;
}
```

```java
Scanner scan = new Scanner(System.in);
System.out.print("Inserisci un numero > 0: ");
int i = scan.nextInt();
while (i <= 0) {
  System.out.print("Inserisci un numero > 0: ");
  i = scan.nextInt();
}
System.out.printf("Hai inserito: %d\n", i);
scan.close();
```

```java
Scanner scan = new Scanner(System.in);
int i;
do {
  System.out.print("Inserisci un numero > 0: ");
  i = scan.nextInt();
}
while (i <= 0);
scan.close();
```

Java 101: Selezione e Cicli

```java
int i = 0;
while (i < N){
  // fai qualcosa
  i++;
}
```

```java
for (int i=0; i < N; i++){
  // fai qualcosa
}
```

Java 101: Selezione e Cicli

```
public class Example {
  /* costanti: */
  public static final char ADDIZIONE = 0;
  public static final char SOTTRAZIONE = 1;
  /* dichiarazione funzioni ... */
  public static void main(String[] args){
    int scelta; //
    switch (scelta) {
      case ADDIZIONE:
        result = addizione(n1, n2);
        break
      /* altri case ... */
      default:
        System.out.println("Scelta non valida!");
    }
  }
}
```

Java 101: Selezione e Cicli

```
public class Example {
  /* costanti: */
  public static final char ADDIZIONE = 0;
  public static final char SOTTRAZIONE = 1;
  /* dichiarazione funzioni ... */
  public static void main(String[] args){
    int scelta; //
    switch (scelta) {
      case ADDIZIONE:
        result = addizione(n1, n2);
        break
      /* altri case ... */
      default:
        System.out.println("Scelta non valida!");
    }
  }
}
```

Costanti: variabili dichiarate con prefisso (`public`) `static final` fuori dal `main`

Poi capiremo che non sono davvero variabili