

UNIVERSITÀ DEGLI STUDI DI VERONA

---

CORSO DI LAUREA IN INFORMATICA

# Programmazione Java

Federico Brutti  
federico.brutti@studenti.univr.it

*Inserire citazione inerente alla materia*

# Contents

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Programmazione orientata agli oggetti . . . . .	3
1.2	Il linguaggio Java . . . . .	3
<b>2</b>	<b>Example 2</b>	<b>4</b>
2.1	Sec1 . . . . .	4
2.2	Sec2 . . . . .	4
2.2.1	SubS1 . . . . .	4

# Chapter 1

## Introduzione

### 1.1 Programmazione orientata agli oggetti

Il corso di programmazione II ha lo scopo di fornire un *modus operandi* differente; passeremo infatti dalla programmazione imperativa ad una **orientata agli oggetti**.

Lo scopo e l'utilità della OOP è quello di gestire appropriatamente un programma con molte linee di codice. Rende il codice estremamente modulare, riutilizzabile e di conseguenza risulta più facile mantenerlo. Per il modo in cui sono trattati gli oggetti, inoltre, il codice è reso particolarmente sicuro.

Familiarizziamo ora con i due concetti base della OOP: **classi** e **oggetti**. Le prime definiscono una struttura dati, alla quale è possibile associare dati, detti **attributi**, e funzioni, chiamate **metodi**. Dalle classi, che in parole povere fungono da tipo di dato, è possibile ottenere gli oggetti, delle loro istanze, con stessi dati e metodi. Infatti, il processo di creazione di un oggetto è detto **istanziamento**.

Il workflow della OOP si basa sulle interazioni fra gli oggetti; queste avvengono grazie alle loro **interfacce**, set di messaggi che l'oggetto può ricevere, mappate ad un metodo nello stesso. Se si riceve un messaggio al di fuori dell'interfaccia, è detto illegale e viene bloccato. Sicuro, no? Ora, l'idea generale per una corretta programmazione a oggetti sono:

1. Identificare i componenti.
2. Definire l'interfaccia dei componenti.
3. Definire le modalità con cui le interfacce consentono l'interazione fra oggetti.
4. Minimizzare le relazioni fra i componenti.

## 1.2 Il linguaggio Java

Unified Modeling Language UML - Linguaggio di modellazione per definire graficamente e documentare un sistema orientato agli oggetti. Ci interessano i diagrammi di classe.

Incapsulamento e information hiding Ereditarietà

# Chapter 2

## Example 2

2.1 Sec1

2.2 Sec2

2.2.1 SubS1