

Corso Java

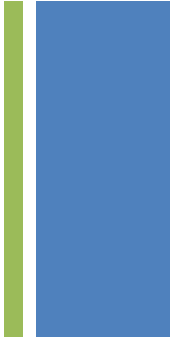
Da Hello World ad accendere una lampada con Android

Giornata 2°

31 Ottobre 2015

[www.campuslacamilla.it](http://www.campuslacamilla.it)

# + Programma di oggi



- **Giornata 2 – Sabato 31 ottobre 2015 – dalle 9 alle 18**
- Utilizzo delle funzioni per effettuare operazioni matematiche/geometriche
- Applicazioni con grafica in Java
- Principio del polimorfismo e MVC
- Creazione della calcolatrice per effettuare operazioni con interfaccia grafica

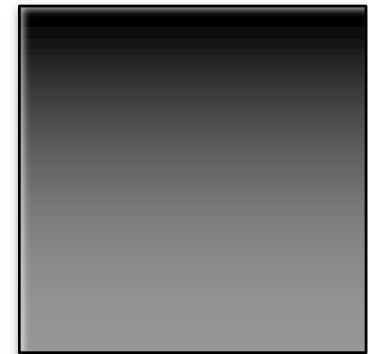
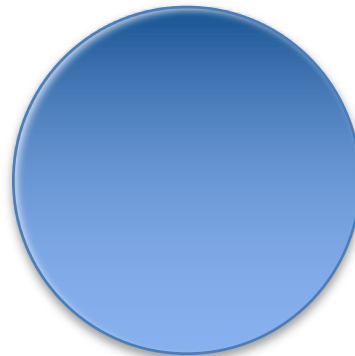


# + Programma Geometria



In questo programma calcoleremo aree e perimetri delle principali figure geometriche, come ad esempio:

Rettangolo, Cerchio e Quadrato





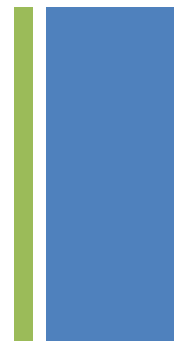
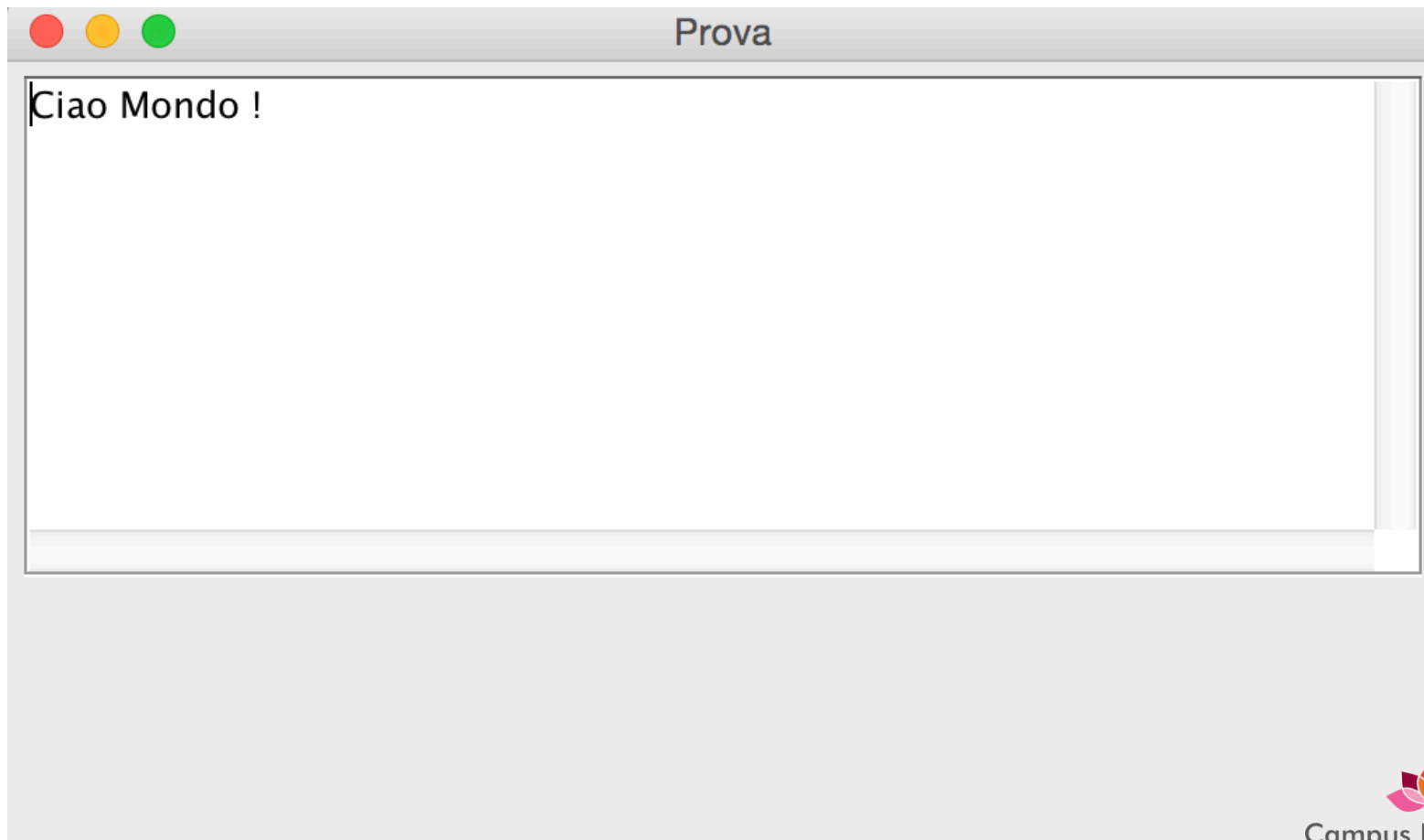
# Applicazioni con Grafica



- Fino ad ora abbiamo usato applicazioni che non hanno un'interfaccia grafica
- Gli utenti finali non gradiscono di dover utilizzare applicazioni stile *terminale*
- Nei prossimi esempi, vedremo cosa mette a disposizione Java per realizzare GUI (Graphic User Interface)



# + Progetto Grafica Test01



# + Progetto Grafica Test02 (1)



- In questo test, dovremmo realizzare un'applicazione in grado di far inserire all'utente il proprio nome e il computer mostrerà un saluto carino



# + Progetto Grafica Test02 (2)



Salutami

Inserisci il tuo nome

Salutami



# Progetto Grafica Test02 (3)



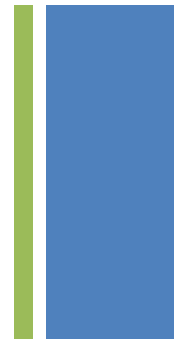
- Sarà necessario realizzare un componente grafico con i seguenti oggetti:

1. **private JLabel label = new JLabel("Inserisci il tuo nome:");**
2. **private JTextField field = new JTextField();**
3. **private JButton button = new JButton("Salutami");**
4. **private JTextArea area = new JTextArea();**





# + Programma Matematica (1)



- In questo programma sarà necessario realizzare dei componenti che permettano di effettuare le principali operazioni, come ad esempio somma, sottrazione, moltiplicazione e divisione
- **Java:**
  - + permette di fare la somma
  - - permette di fare la sottrazione
  - \* permette di fare la moltiplicazione
  - / permette di calcolare il quoziente, % il resto

# + Programma Matematica (2)



- Sarà necessario quindi realizzare 4 classi, che chiameremo Somma, Sottrazione, Moltiplicazione e Divisione.
- Il codice che realizzeremo sarà alla base della calcolatrice grafica che verrà realizzata più avanti.



# + Polimorfismo



- È secondo me l'argomento più difficile di tutto il corso...
- Per capirlo bene bisogna programmare, programmare e programmare ancora...
- Tradotto significa *diverse rappresentazioni*
- Si può riassumere come l'ereditarietà applicata ad oggetti che sono simili tra loro, ma ognuno ha una propria caratteristica





# Esempio strumenti musicali



- Ci sono diversi tipi di strumenti musicali: pianoforte, flauto, violino
- Tutti emettono un suono, ma è particolare per ogni strumento
- Se viene registrato il suono attraverso un recorder MP3, è necessario avere un dispositivo particolare per essere riprodotto ?
- Il polimorfismo è proprio questo: lo stesso file mp3 può contenere diversi strumenti musicali, ma per il vostro iPod non interessa, viene riprodotto comunque
- In questo modo, posso registrare qualsiasi strumento musicale, senza dover modificare il mio recorder MP3



# A cosa serve il polimorfismo ?



- Per capirlo ci vuole tanta pratica
- Ma si può dire che serva per rendere il nostro codice più robusto a possibili cambiamenti.



# + Design pattern



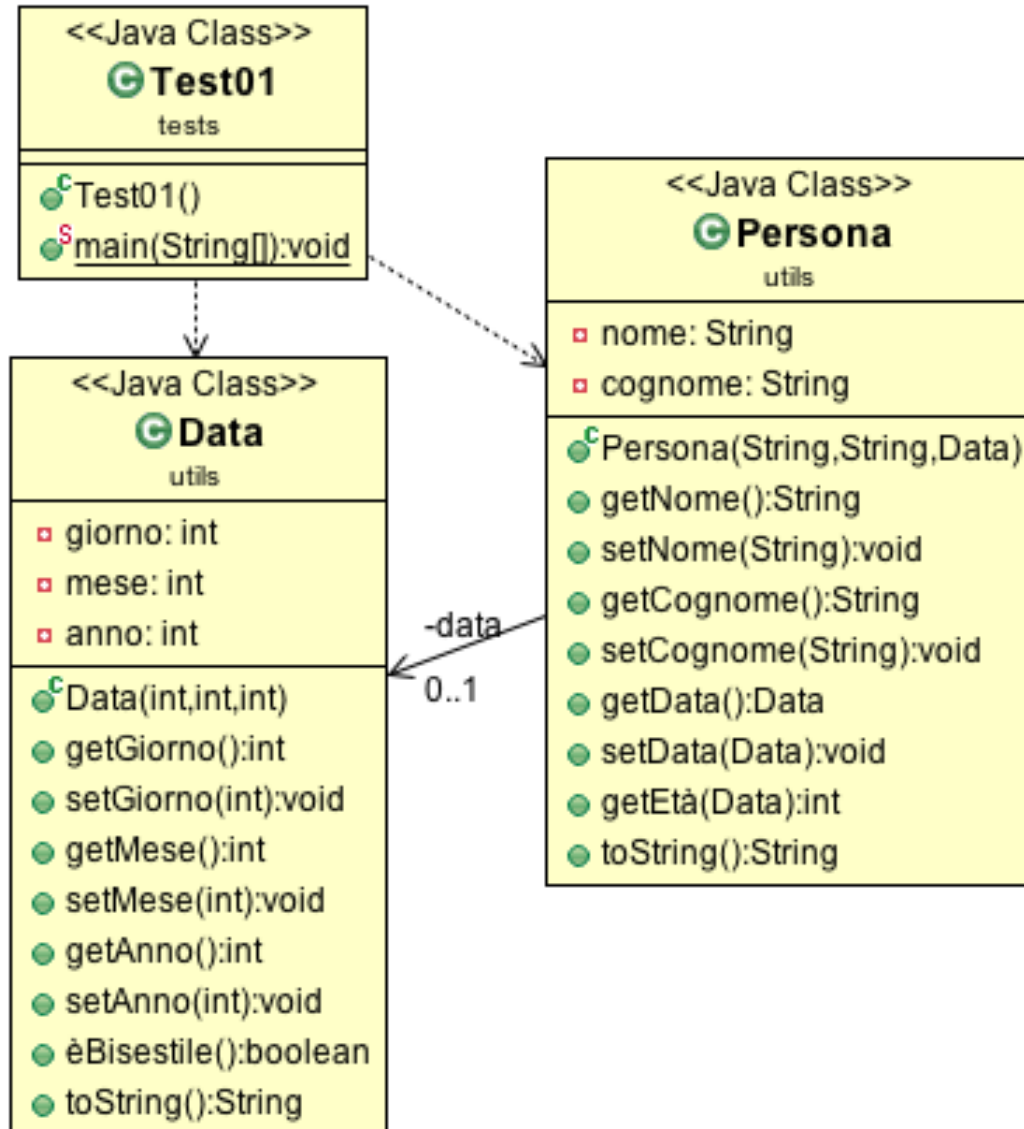
- Che cosa sono ?
- Sono dei “percorsi”/mappe, che permettono di collegare diversi componenti, all’interno della medesima applicazione.
- In pratica permettono di strutturare meglio il proprio codice
- Ne esistono davvero tanti
- Durante il corso ne vedremo qualcuno

# + UML



- Come si fa a vedere come sono collegati i nostri componenti all'interno dell'applicazione ?
- Si possono fare i diagrammi UML
- Esiste un plugin che permette di automatizzare la procedura di realizzazione

# + Esempio UML





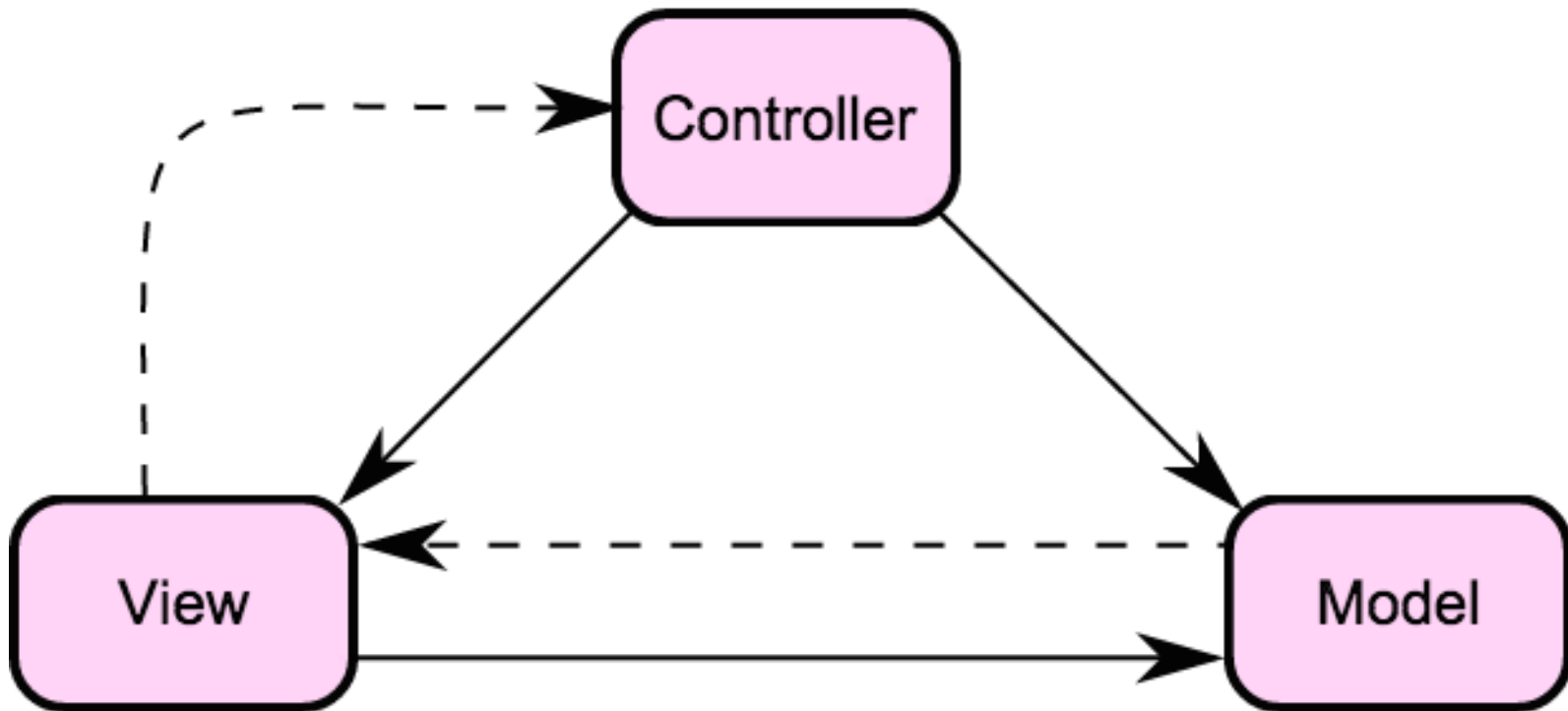
# + MVC



- È il design pattern più famoso, importante nella programmazione ad oggetti
- MVC sta per:
  1. Model = il modello del sistema
  2. View = il componente che mostra lo stato del sistema
  3. Control = il componente che permette di interagire con il sistema



# + UML del DP MVC



# + Programma MVC (1)



- Questo programma deve permettere all'utente di inserire il proprio nome e una volta che premerà il tasto “Salutami”, un messaggio di saluto con il proprio nome comparirà
- Il Modello è il componente che permette di fare il saluto, dato il nome
- Il controller è il componente che permette di ricevere il nome della persona e determina l'azione “Salutami”
- La View mostra il risultato dell'operazione, cioè il saluto



# + Programma MVC (2)

MVP

Inserisci il tuo nome:

Paolo

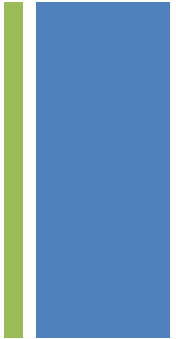
Saluta

Ciao Paolo

Il controller

La view del sistema

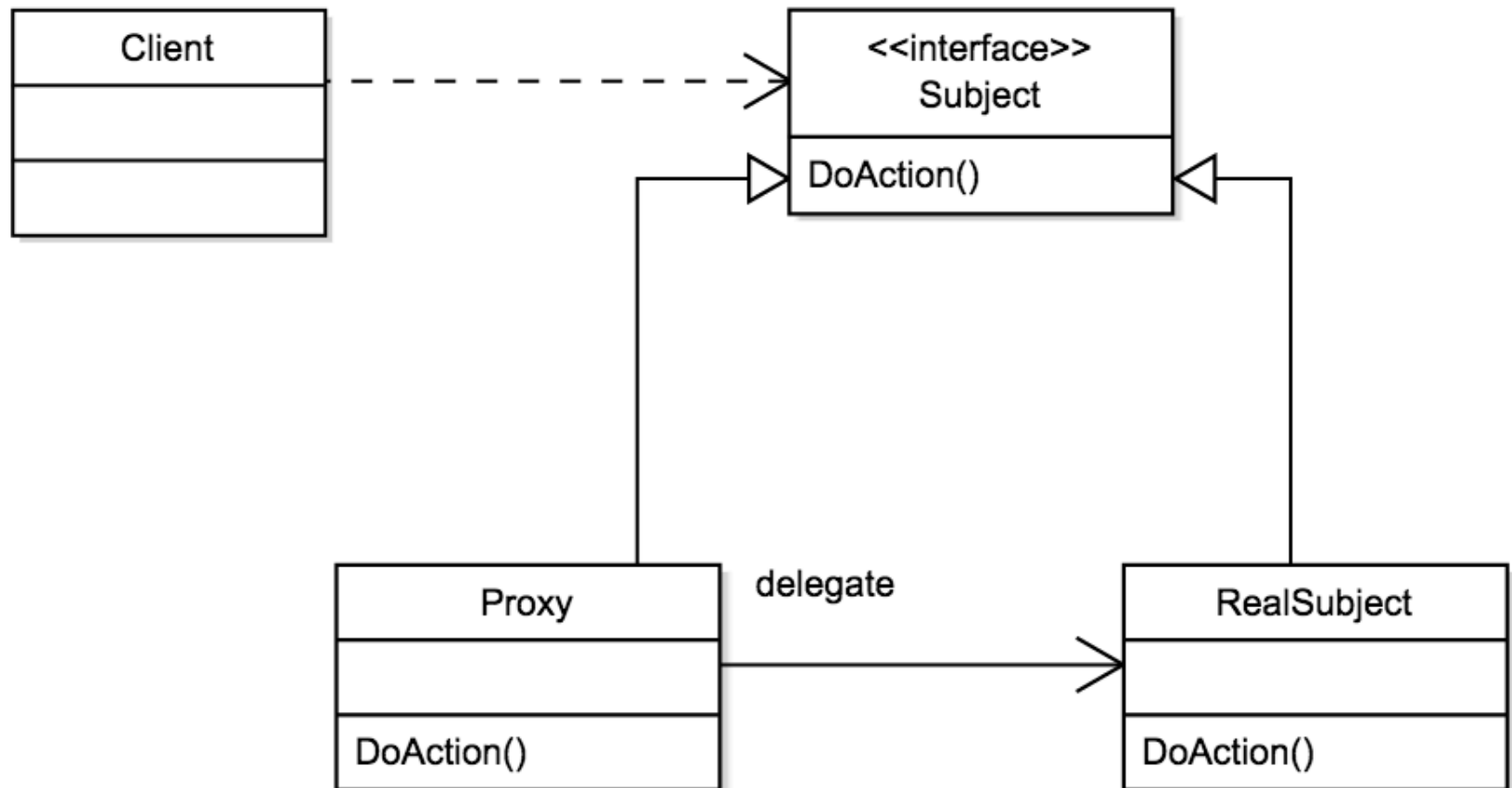
# + Design pattern Proxy



- Introduciamo un altro design pattern molto usato nella programmazione ad oggetti: il Proxy
- In Inglese Proxy significa delegato e il suo utilizzo rispetta tale termine
- In pratica il Proxy funge da delegato per un client, cioè un altro un componente nel nostro programma
- Più semplicemente è un componente a cui si chiede qualcosa e lui fornisce il risultato per conto di un altro



# + UML generico del Proxy





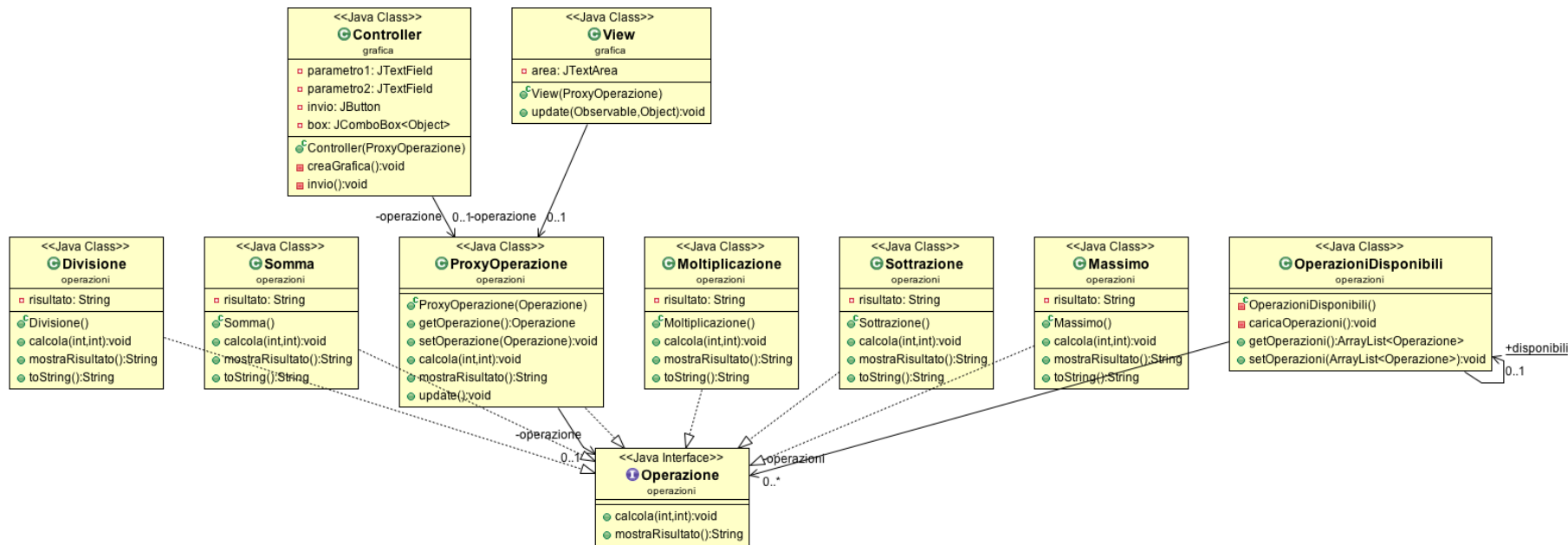
# Calcolatrice: MVC + Proxy



- Questo programma deve svolgere il ruolo della calcolatrice, cioè permetta all'utente di effettuare le operazioni di somma, sottrazione, moltiplicazione e divisione tra due numeri e mostrare il risultato, il tutto attraverso un'interfaccia grafica.
- Questo programma utilizza la combinazione del MVC e del Proxy.
- Ritengo che sia un gioiellino della programmazione, perché include praticamente di tutto !
- In realtà ci sono altri design pattern, che discuteremo insieme.

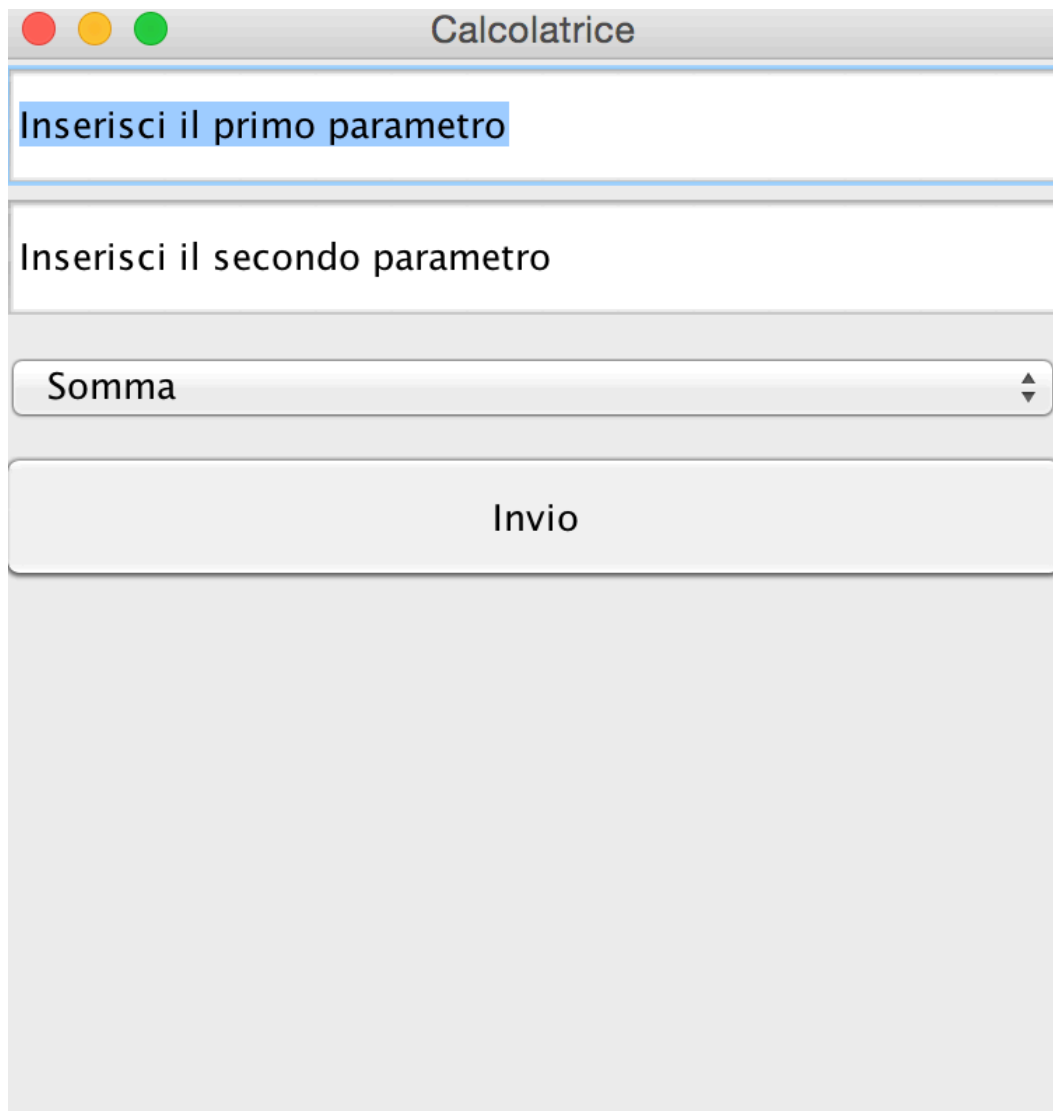


# + UML della Calcolatrice





# + Calcolatrice grafica



Calcolatrice

Inserisci il primo parametro

Inserisci il secondo parametro

Somma

Invio

