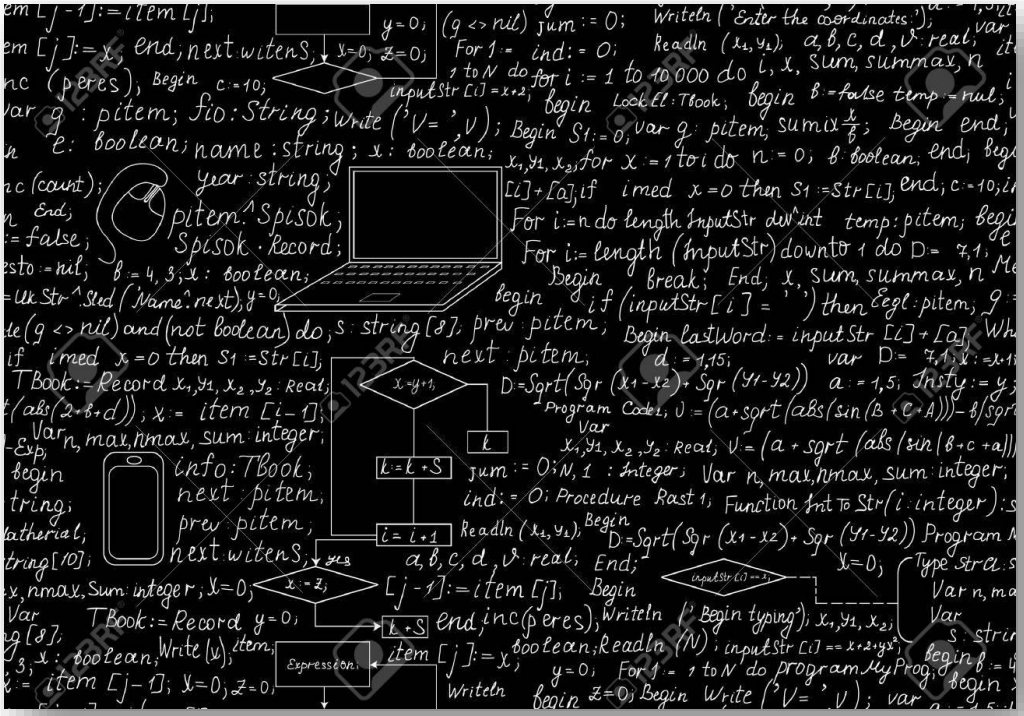


programming pattern

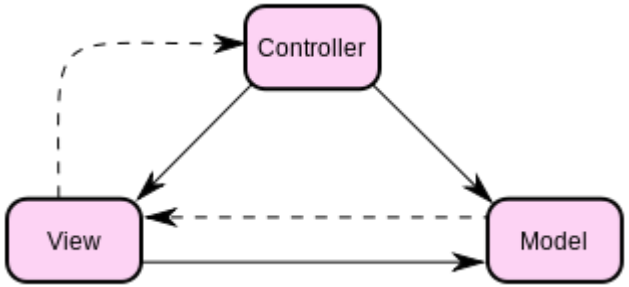
MVC - observer

pattern architetturale (design pattern)

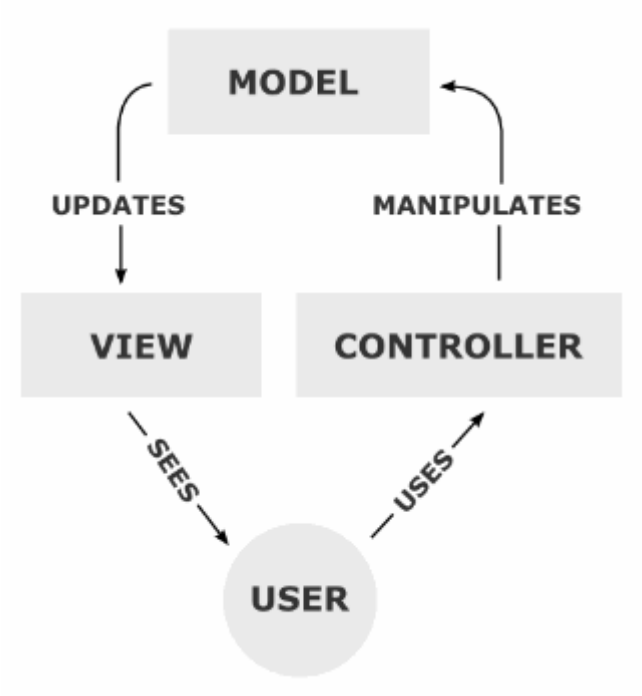
A. Ferrari

- nell'ambito dell'ingegneria del software, un ***design pattern*** è un concetto che può essere definito *"una soluzione progettuale generale a un problema ricorrente"*
- si tratta di un modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software

- **M**odel-**V**iew-**C**ontroller è un ***pattern*** architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business
- originariamente impiegato dal linguaggio Smalltalk, il pattern è stato sposato da numerose tecnologie moderne, come framework basati su PHP, su Ruby, su Python, su Java (Swing …), su Objective C o su .NET



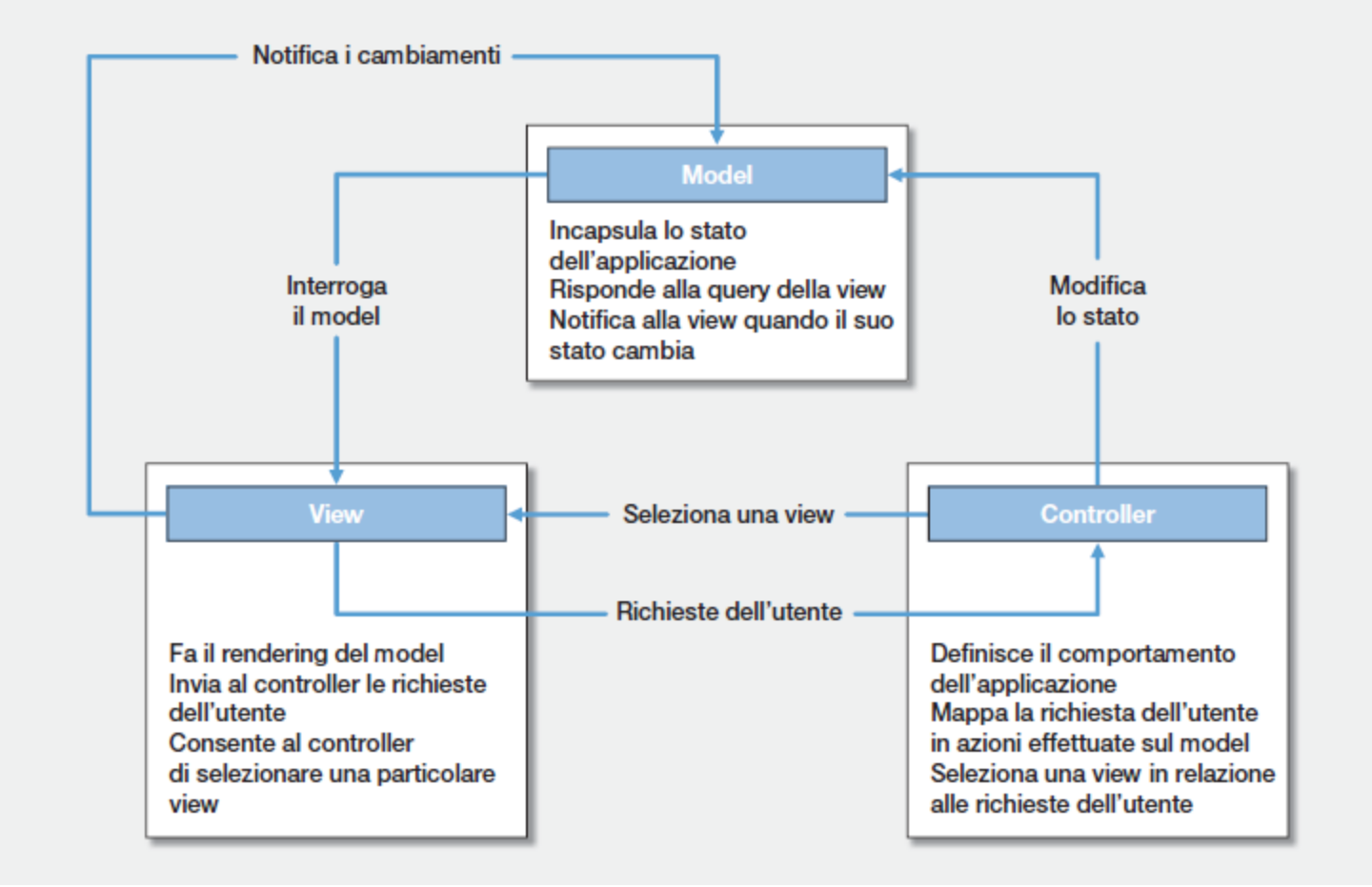
- il pattern è basato sulla ***separazione dei compiti*** fra i componenti software che interpretano tre ruoli principali:
  - il ***model*** fornisce i metodi per ***accedere*** ai ***dati*** utili all'applicazione
  - il ***view*** ***visualizza*** i dati contenuti nel model e si occupa dell'***interazione*** con utenti e agenti
  - il ***controller*** riceve i ***comandi*** dell'utente (in genere attraverso il view) e li ***attua*** modificando lo stato degli altri due componenti



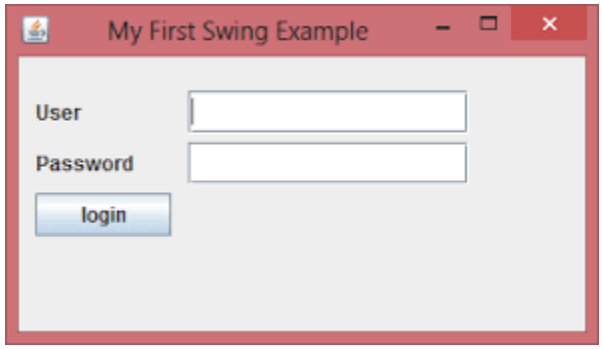
- MVC implica la ***separazione*** fra
  - la ***logica applicativa*** a carico del controller e del model
  - l'***interfaccia utente*** a carico del view

vantaggi

- MVC consente di ***suddividere*** la *complessità* di realizzazione di un'applicazione dotata di GUI allo scopo di rendere più semplici
  - lo sviluppo
  - la manutenzione
  - la riusabilità del codice
- ***indipendenza*** tra i business data (model) la logica di presentazione (view) e quella di controllo (controller)
- ***viste diverse*** per il medesimo model


Notifica i cambiamenti
Model
Incapsula lo stato dell'applicazione
Risponde alla query della view
Notifica alla view quando il suo stato cambia
Interroga il model
Modifica lo stato
View
Seleziona una view
Controller
Richieste dell'utente
Fa il rendering del model
Invia al controller le richieste dell'utente
Consente al controller di selezionare una particolare view
Definisce il comportamento dell'applicazione
Mappa la richiesta dell'utente in azioni effettuate sul model
Seleziona una view in relazione alle richieste dell'utente

- nei componenti della libreria AWT il pattern MVC è realizzato in un modo semplificato:
  - è previsto un unico elemento per l'implementazione degli aspetti view e controller



programmazione guidata dagli eventi

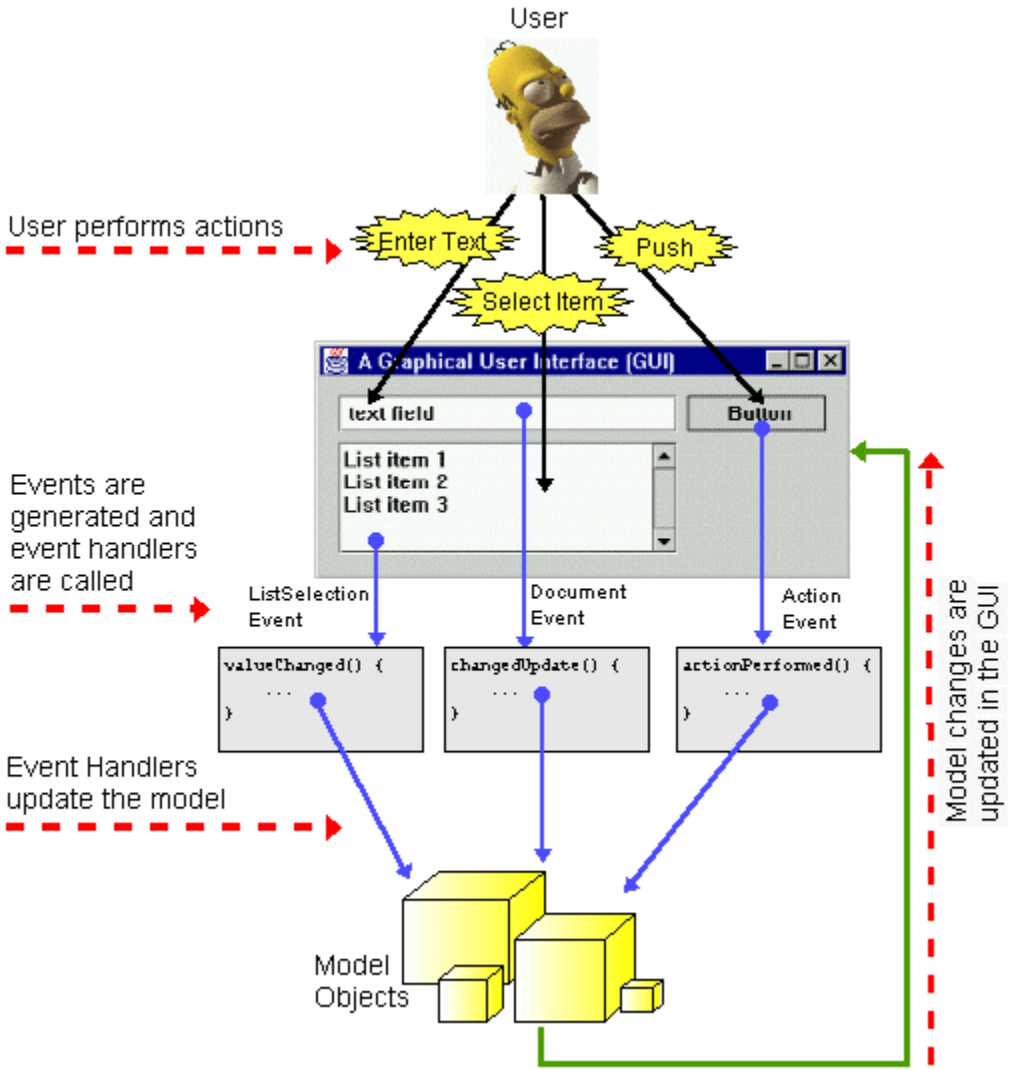
- programmazione ***event driven***:
  - l'esecuzione del codice è guidata dagli eventi generati dall'interazione dell'utente con i componenti grafici

pattern Observer

- il pattern ***Observer*** (Osservatore) definisce una ***dipendenza*** uno a molti fra un soggetto ***osservato*** e vari oggetti «***osservatori***», in modo che se il soggetto osservato modifica il suo stato, a tutti gli oggetti osservatori che si sono esplicitamente registrati viene notificato il cambiamento avvenuto
- per poter notificare i cambiamenti di stato del soggetto agli osservatori, viene richiesto agli osservatori di ***sottoscriversi*** presso il soggetto
- il soggetto mantiene una ***lista*** degli ***osservatori*** registrati, per notificare a ciascuno di essi i propri cambiamenti di stato invocando un metodo specifico

- una classe astratta o una interfaccia definisce il ***prototipo del metodo*** di notifica
  - la sottoscrizione di un'istanza di un oggetto osservatore presso il soggetto rende disponibile un metodo specifico da invocare a ogni aggiornamento dello stato
- nella libreria Java AWT il pattern Observer viene utilizzato per realizzare gli ascoltatori (***Listener***) degli eventi asincroni generati dai componenti grafici

Elementi del pattern	Java AWT
Soggetto osservato	Componente *Button*
Osservatore astratto	Interfaccia *ActionListener*
Osservatore concreto	Oggetto che implementa l'interfaccia *ActionListener*


User
User performs actions
Enter Text
Push
Select Item
A Graphical User Interface (GUI)
text field
Button
List item 1
List item 2
List item 3
Events are generated and event handlers are called
ListSelection Event
Document Event
Action Event
valueChanged() {
...
}
changedUpdate() {
...
}
actionPerformed() {
...
}
Event Handlers update the model
Model Objects
Model changes are updated in the GUI