Programmazione Orientata agli Oggetti in Linguaggio Java

Ruoli e Responsabilità: Strati Applicativi

versione 2.3

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Ruoli e Strati Applicativi
- Architettura di Base
- Esempio: Indovina Il Numero
- Esempio: La Morra Cinese
- Vantaggi degli Strati Applicativi
- Linee Guida

Ruoli e Strati Applicativi

- Ruolo di un componente
  - ⇨suoi compiti nell'applicazione
  - ⇨descritti dalla sua interfaccia
- Come scegliere l'interfaccia
  - ⇨altri principi guida della programmazione
  - ⇨massimizzare la coesione
  - ⇨minimizzare l'accoppiamento

Ruoli e Strati Applicativi

- Minimizzare l'accoppiamento
  - ⇨i componenti dovrebbero dipendere meno possibile gli uni dagli altri
  - ⇨la comunicazione dovrebbe essere basata su interfacce chiare e ragionevolmente semplici
  - ⇨i componenti non devono conoscere i dettagli delle implementazioni di altri componenti
  - ⇨cambiamenti nell'implementazione di un componente non devono influenzare gli altri

Ruoli e Strati Applicativi

- Massimizzare la coesione
  - ogni componente dovrebbe avere una serie di compiti ben identificati e dello stesso tipo
  - un componente non dovrebbe occuparsi di fare più cose di natura diverse
- In altri termini
  - i componenti dovrebbero essere specializzati

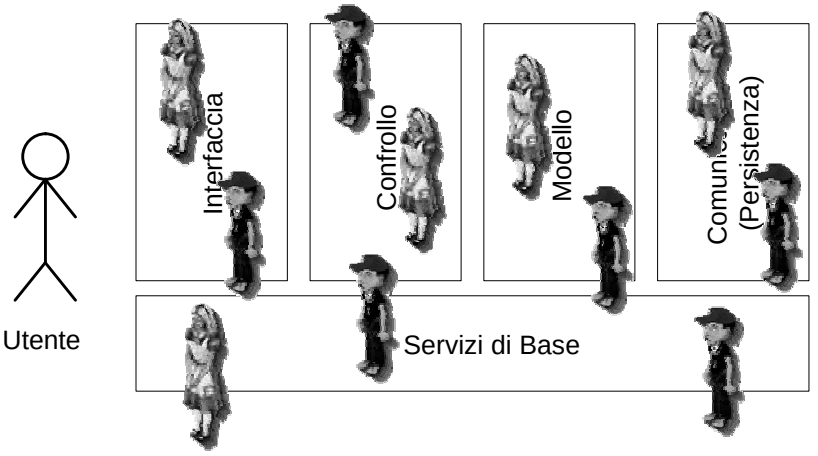
Ruoli e Strati Applicativi

- I ruoli tipici di un’applicazione
  - corrispondono agli strati applicativi
  - ogni strato è una collezione di componenti che sono in grado di portare a termine una funzione specifica
- Metafora
  - il ruolo è il “mestiere” del componente
  - lo strato applicativo rappresenta il luogo di lavoro del componente

Ruoli e Strati Applicativi

- I principali strati applicativi
  - ⇨corrispondono ad un'architettura considerata standard per le applicazioni a oggetti
  - ⇨interfaccia
  - ⇨controllo
  - ⇨modello
  - ⇨comunicazione con l'esterno
  - ⇨servizi di base

Ruoli e Strati Applicativi



Ruoli e Strati Applicativi

- Interfaccia
  - ⇨gestisce l'interazione con l'utente (schermi)
- Controllo
  - ⇨gestisce il flusso di controllo dell'applicazione (prende le decisioni su cosa fare dopo)
- Modello
  - ⇨mantiene lo stato dell'applicazione (dati) e implementa la "logica applicativa"

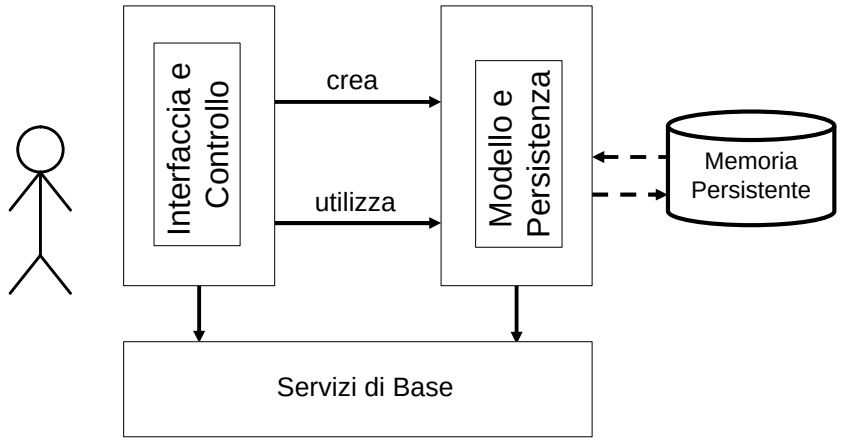
Ruoli e Strati Applicativi

- Comunicazione con l'esterno
  - ⇨realizza tutte le forme di comunicazione del sistema con sistemi esterni
  - ⇨in particolare: gestione della persistenza – es: comunicazione con il file system
- Servizi di base
  - ⇨componenti di utilizzo trasversale; es: librerie matematiche, collezioni, ecc.

Architettura di Base

- Problema
  - ⇨come organizzare questi strati in un'architettura ?
  - ⇨quali sono le regole di comunicazione ?
  - ⇨per cominciare, un'architettura semplice
- Architettura di base
  - ⇨interfaccia e controllo assieme
  - ⇨modello e persistenza assieme
  - ⇨più i componenti di base

Architettura di Base



Architettura di Base

- Interfaccia e Controllo
  - ⇨l'applicazione è fatta di una sequenza di schermi – tipicamente menu con opzioni
  - ⇨in ogni schermo vengono visualizzati dati e vengono acquisiti dati dall'utente
  - ⇨sulla base dei dati forniti dall'utente, viene decisa la prossima operazione
  - ⇨ed il successivo schermo da visualizzare

Architettura di Base

- Modello e Persistenza
  - ⇨il modello è fatto di una serie di componenti che mantengono lo stato dell'applicazione
  - ⇨eseguono la logica applicativa – ovvero le operazioni fondamentali dell'applicazione
  - ⇨eventualmente contengono il codice per la gestione della persistenza (es: gestione del file system)

Architettura di Base

- Analogo
  - ⇨l'applicazione è un negozio di lavanderia
  - ⇨i componenti di interfaccia&controllo stanno al banco e ricevono dai clienti le commesse (funzione di interfaccia)
  - ⇨comandano il funzionamento dei componenti di modello&persistenza (funzione di controllo)

Architettura di Base

- Analogo (continua)
  - ⇨i componenti di modello&persistenza lavorano nel retrobottega
  - ⇨si occupano del lavaggio, delle macchine, tengono la contabilità e lo stato del magazzino (logica applicativa)
  - ⇨i componenti di interfaccia&controllo restituiscono gli indumenti lavati ai clienti e gestiscono i reclami (funzione di controllo)

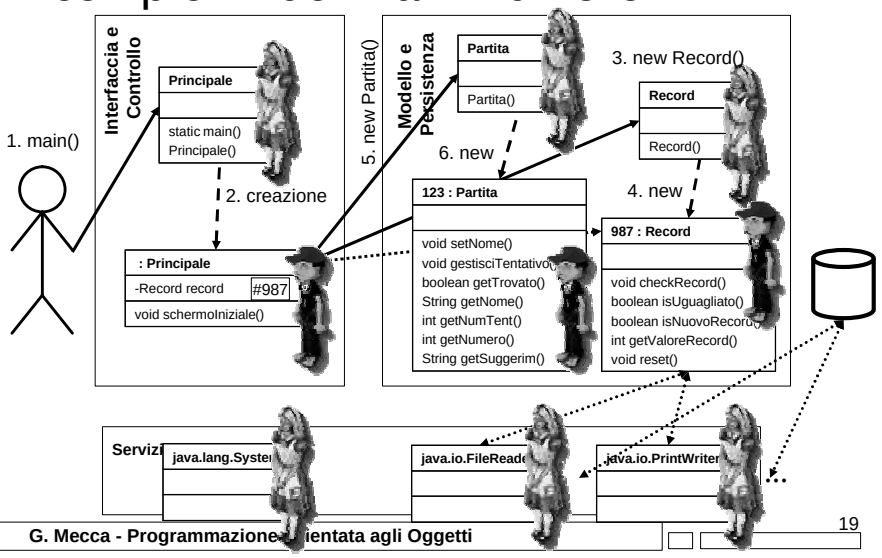
Architettura di Base

- I servizi di base
  - componenti delle librerie di Java
  - varie centinaia di classi di utilizzo più o meno comune
- Analogo
  - i servizi di base sono gli strumenti di lavoro della lavanderia
  - dalla macchina lavatrice al ferro da stiro

Esempio: Indovina il Numero

- Interfaccia e controllo
  - comunicazione con l'utente (console)
  - gestisce il controllo della partita
- Modello e Persistenza
  - stato della partita e regole di funzionamento del gioco
  - gestione del record persistente su file

Esempio: Indovina il Numero



Esempio: Indovina il Numero

ATTENZIONE
al concetto
di bean

- Struttura tipica di una classe del modello
  - ⇨una serie di proprietà private (stato)
  - ⇨metodi get e set per le proprietà
  - ⇨costruttore no-arg
  - ⇨altri metodi pubblici relativi alla logica applicativa
  - ⇨eventuali altri costruttori
- Terminologia Java per queste classi
  - ⇨“bean” o “JavaBean”

```
package it.unibas.indovinailnumero;

public class Partita {

    public Partita() { ... }

    public String getNome() { ... }
    public void setNome(String nome) { ... }

    public boolean getTrovato() { ... }

    public String getSuggerimento() { ... }

    public int getNumeroDiTentativi() { ... }

    public int getNumeroDaIndovinare() { ... }

    public void gestisciTentativo(int tentativo) { ... }
    ...
}
```

Partita è un JavaBean

Esempio: Indovina il Numero

- In un JavaBean
  - ⇨i metodi get e set devono rispettare la convenzione dei nomi
  - ⇨es: String nome: public String getNome(); public void setNome(String nome);
  - ⇨per le proprietà di tipo boolean invece che get è possibile utilizzare anche is
  - ⇨es: boolean trovato: boolean getTrovato() oppure boolean isTrovato()

Esempio: Indovina il Numero

- La classe Record
  - ⇨interfaccia: checkRecord, isUguagliato, isNuovoRecord, getValoreRecord
- L'implementazione
  - ⇨utilizza il file system per salvare il valore del record corrente in un file di testo
  - ⇨il formato del file è estremamente semplice (un'unica riga contenente un numero)

```
package it.unibas.indovinailnumero;

public class Record {

    public Record() {...}

    public boolean isNuovoRecord() { ... }

    public boolean isUguagliato() { ... }

    public int getValoreRecord() { ...}

    public void checkRecord(int tentativi) { ... }

    public void reset() { ... }
    ...
}
```

Record è un JavaBean

Esempio: La Morra Cinese

- it.unibas.morracinese
  - ⇨un'applicazione per giocare alla morra cinese
- Due casi d'uso principali
  - ⇨"Utente gioca partita"
  - ⇨"Utente visualizza punteggi"
  - ⇨si tratta di un'applicazione leggermente più complessa che richiede di utilizzare un numero superiore di classi

Esempio: La Morra Cinese

- Modello e persistenza
  - ⇨ non ci sono requisiti di persistenza
  - ⇨ il modello è composto di varie classi che rispecchiano i concetti della specifica
- Tre concetti principali
  - ⇨ il concetto di "sessione di gioco"
  - ⇨ il concetto di "partita"
  - ⇨ il concetto di "mano"

Esempio: La Morra Cinese

- Le relazioni tra questi concetti
  - ⇨ l'utente partecipa ad una sessione di gioco e gioca varie partite (di cui è necessario tenere il punteggio)
  - ⇨ durante la partita il giocatore gioca varie mani (di cui è necessario tenere il punteggio)
  - ⇨ ciascuna mano è composta degli oggetti giocati dal giocatore e dal computer

Esempio: La Morra Cinese

- Gioco
  - ⇨ gestisce lo stato dell'intera sessione di gioco (punteggio delle partite e nome giocatore)
- Partita
  - ⇨ mantiene lo stato di ogni singola partita (punteggio delle mani)
- Mano
  - ⇨ gestisce le singole mani (giocata del computer e del giocatore)

```
package it.unibas.morracinese;

public class Gioco {

    public String getNomeGiocatore() { ... }

    public void setNomeGiocatore(String nomeGiocatore) { ... }

    public int getPartiteVinteDalComputer() { ... }

    public int getPartiteVinteDalGiocatore() { ... }

    public void gestisciPartita(Partita partita) { ... }
    ...
}
```


```
package it.unibas.morracinese;

public class Partita {
    public final static int LIMITEMANI = 3;

    public final static int PARTITAVINTADALCOMPUTER = 0;
    public final static int PARTITAVINTADALGIOCATORE = 1;
    public final static int PARTITAINCORSO = 2;

    public int getManiVinteDalComputer() { ... }

    public int getManiVinteDalGiocatore() { ... }

    public int getPareggi() { ... }

    public void gestisciMano(Mano mano) { ... }

    public int getStato(){ ... }
    ...
}
```

```
package it.unibas.morracinese;

public class Mano {
    public final static int CARTA = 1;
    public final static int FORBICI = 2;
    public final static int SASSO = 3;

    public final static int MANOVINTADALCOMPUTER = 0;
    public final static int MANOVINTADALGIOCATORE = 1;
    public final static int MANOINPAREGGIO = 2;

    public void gioca(int giocataGiocatore) { ... }

    public boolean isValida(int giocata) { ... }

    public int getGiocataGiocatore() { ... }

    public int getGiocataComputer() { ... }

    public int getEsito() { ...}
    ...
}
```


Esempio: La Morra Cinese

- Attenzione
  - ⇨ ci sono altri concetti potenzialmente interessanti
  - ⇨ ma che non danno luogo a classi perchè rappresentano informazioni troppo semplici
- Esempi
  - ⇨ il concetto di punteggio (si tratta di un numero)
  - ⇨ il concetto di oggetto (“carta”, “forbici” o “sasso”): potrebbe essere una classe; in effetti, però, si tratta di un numero o di una stringa; per semplicità decidiamo di trascurarla

Esempio: La Morra Cinese

- Interfaccia e controllo
  - ⇨ il fatto che ci siano due casi d'uso rende più complesso lo strato di interfaccia&controllo
  - ⇨ vari casi d'uso corrispondono tipicamente ad un menu principale con altrettante opzioni
- Una scelta tipica
  - ⇨ per evitare classi troppo numerose, una classe di interfaccia e controllo per ciascun caso d'uso

Esempio: La Morra Cinese

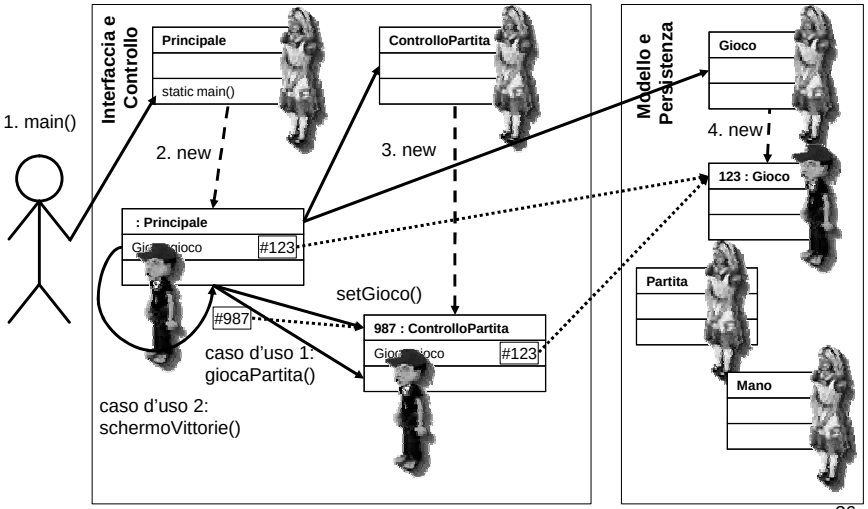
- Principale
  - ⇨ la classe del menu principale
  - ⇨ gestisce anche il caso d'uso "Utente visualizza punteggi"
- ControlloPartita
  - ⇨ gestisce all'occorrenza il caso d'uso "Utente gioca partita"
  - ⇨ deve conoscere lo stato del gioco per il nome dell'utente

Esempio: La Morra Cinese

>> Principale.java
>> ControlloPartita.java

- Flusso di controllo
  - ⇨ Principale visualizza il menu principale
  - ⇨ utilizza un oggetto di tipo ControlloPartita (variabile locale al metodo gioca())
  - ⇨ entrambi conoscono lo stato del gioco
  - ⇨ se l'utente sceglie di giocare una nuova partita, attribuisce la responsabilità al ControlloPartita e poi riprende il controllo

Esempio: La Morra Cinese



Vantaggi degli Strati Applicativi

- Riassumendo
  - ⇨gli strati applicativi sono uno strumento importante
  - ⇨le funzionalità sono concentrate in punti precisi
- Due vantaggi principali
  - ⇨da una parte aiutano a progettare il codice
  - ⇨dall'altra aiutano ad interpretare e a manutenere il codice

Vantaggi degli Strati Applicativi

- In particolare
  - ⇨facilitano la manutenzione perchè proteggono gli strati dalle variazioni degli altri
- Proteggersi dalle variazioni
  - ⇨un altro principio metodologico fondamentale
  - ⇨isolare le parti che cambiano da quelle che non cambiano
- Esempio
  - ⇨cambiare il tipo di interfaccia o di persistenza

Vantaggi degli Strati Applicativi

- Tecnologie per l'interfaccia
  - ⇨console (la più semplice)
  - ⇨interfaccia grafica (GUI) (menu, bottoni,...)
  - ⇨interfaccia Web
- Tecnologie per la persistenza
  - ⇨file system con formato libero
  - ⇨file system e XML
  - ⇨DBMS (sistemi di gestione di basi di dati)

Vantaggi degli Strati Applicativi

- Frequentemente
  - ⇨una stessa logica applicativa viene utilizzata con tipologie di interfaccia diverse
  - ⇨se il codice dell'interfaccia fosse "diffuso" in tutta l'applicazione, cambiare interfaccia vorrebbe dire cambiare tutta l'applicazione
- Viceversa
  - ⇨gli strati applicativi isolano parte dell'applicazione dalle modifiche del resto

Linee Guida

- Attenzione
  - ⇨perchè gli strati applicativi consentano di organizzare bene il codice è necessario rispettare delle regole
- Linee guida per interfaccia&controllo
  - ⇨l'inizio dell'esecuzione è sempre nello strato di interfaccia&controllo (main)
  - ⇨i componenti di interfaccia&controllo visualizzano dati e acquisiscono dati

Linee Guida

- Linee guida (continua)
  - ⇨i componenti di interfaccia&controllo creano i componenti di modello&persistenza
  - ⇨i componenti di interfaccia&controllo chiamano metodi dei componenti di modello&persistenza per accedere allo stato dell'applicazione
  - ⇨i componenti di interfaccia&controllo prendono decisioni sullo schermo successivo

Linee Guida

- Linee guida (continua)
  - ⇨i componenti di interfaccia&controllo NON accedono a sistemi esterni (es: disco)
  - ⇨i componenti di interfaccia&controllo NON eseguono logica applicativa
- Linee guida per modello&persistenza
  - ⇨i componenti di modello&persistenza vengono creati dai componenti di interfaccia&controllo

Linee Guida

- Linee guida (continua)
  - ⇨i componenti di modello&persistenza NON creano componenti di interfaccia&controllo
  - ⇨i componenti di modello&persistenza vengono utilizzati e modificati dai componenti di interfaccia&controllo
  - ⇨i componenti di modello&persistenza NON utilizzano i componenti di interfaccia&controllo

Linee Guida

- Linee guida (continua)
  - ⇨i componenti di modello&persistenza mantengono lo stato dell'applicazione e conoscono le regole di funzionamento dell'applicazione
  - ⇨i componenti di modello&persistenza NON visualizzano dati nè acquisiscono dati dall'utente

Linee Guida

- Attenzione
  - ⇨differenza tra package e strato applicativo
  - ⇨sono due concetti indipendenti
- Il package
  - ⇨è uno strumento per organizzare il codice e i nomi delle classi
  - ⇨implicazioni sintattiche (istruzione package)
  - ⇨metaforicamente, rappresenta il condominio (l'indirizzo) di una classe

Linee Guida

- Lo strato applicativo
  - ⇨è uno strumento per dividere il lavoro tra i componenti
  - ⇨non ha implicazioni sintattiche
  - ⇨metaforicamente: il luogo di lavoro del componente
- Molto spesso
  - ⇨i package corrispondono agli strati applicativi
  - ⇨ma questo non è in alcun modo necessario

Riassumendo

- Ruoli e Strati Applicativi
- Architettura di Base
- Esempio: Indovina Il Numero
- Esempio: La Morra Cinese
- Vantaggi degli Strati Applicativi
- Linee Guida

Termini della Licenza