PROGETTO MOVIDA

Angelo Di Iorio

Università di Bologna

Progetto MOVIDA

- MOVIDA (MOVIes Data and Algorithms) è un'applicazione Java per interagire con una knowledge-base a tema cinema
- MOVIDA permette di:
  - importare una knowledge-base
  - visualizzare informazioni su film e attori
  - cercare informazioni in base a criteri diversi
- Il progetto si svolge in gruppo e ogni gruppo dovrà fornire implementazioni diverse di alcune strutture dati e algoritmi
  - **Obbligatorio registrare il gruppo per sviluppare e consegnare il progetto (istruzioni nelle prossime slide)**

Regole formali

- I gruppi sono composti di 2 persone. **Non sono ammessi gruppi di 3 o più persone.**
- Non sono ammesse consegne individuali a meno di eccezioni da motivare e verificare singolarmente
- Si consegna una volta sola e il voto è valido per l'intero anno accademico (e per gli anni successivi se le regole non cambiano)
- Il progetto vale ¼ del voto complessivo (media pesata)
- Il progetto di quest'anno si può presentare fino a Febbraio 2021

Consegna e discussione

- Il progetto si consegna 5 giorni prima della discussione e va presentato al docente di persona (o su Teams)
- Consegna su IOL, seguiranno istruzioni

- Entrambi i membri del gruppo devono essere presenti alla discussione

- Per la discussione ci saranno appelli su AlmaEsami dedicati al progetto
- **Iscrivere un solo membro del gruppo**

Sessione estiva – discussione progetto

- 3 Giugno, h. 10
- 18 Giugno, h. 14
- 2 Luglio, h. 10
- 16 Luglio, h. 14

- Date da confermare ma variazioni minime di giorni e orario

MOVIDA Core

- Il progetto consiste nella realizzazione di un modulo software pensato per essere integrato in un'applicazione più complessa
- Per gli scopi del corso, non è richiesta l'implementazione dell'interfaccia GUI e da linea di comando ma solo del *core* dell'applicazione
- `MovidaCore` è la classe *entry point* dell'applicazione e sarà testata in modo automatico
- Il risultato dei test sarà il punto di partenza della discussione del progetto

MOVIDA Core

- Un'istanza della classe `MovidaCore` permette di caricare e successivamente recuperare le informazioni relative a film, attori e registi
- La classe `MovidaCore` implementa alcune interfacce, presentate nelle prossime slide e disponibili su IOL, che descrivono tre gruppi di operazioni:
  1. `IMovidaDB`: Caricamento in memoria e lettura dei dati
  2. `IMovidaConfig`: Scelta degli algoritmi e strutture dati da usare
  3. `IMovidaSearch`: Ricerca delle informazioni

Dati in MOVIDA

- Il modello dei dati è (molto) semplificato e prevede due classi:
  - `Movie`: informazioni relative ad un film
    - Titolo
    - Anno
    - Regista
    - Lista di Attori
    - Voti su IMDb
  - `Person`: informazioni relative ad un attore o un regista
    - Nome
- Le classi di partenza sono disponibili su IOL ma possono essere estese, mantendo i metodi `get()` per accedere alle informazioni in lettura

Movie

```
/*** Esempio film
 *
 * Title: Taxi Driver
 * Year: 1976
 * Director: Martin Scorsese
 * Cast: Robert De Niro, Jodie Foster, Cybill Shepherd, Albert Brooks
 * Votes: 684728
 */
public class Movie {

	private String title;
	private Integer year;
	private Integer votes;
	private Person[] cast;
	private Person director;

	// getter omessi
```

Person

```
public class Person {

	private String name;

	public Person(String name) {
		this.name = name;
	}

	public String getName(){
		return this.name;
	}

}
```

Caricamento di dati da file

- MOVIDA definisce un formato testuale per il salvataggio dei dati su file
  - Ogni film è descritto da un record separato dal record successivo con una riga vuota
  - Ogni record è composto da diversi campi (coppie *chiave:valore*), uno per riga
- `MovidaCore` espone i metodi per caricare/salvare i dati in questo formato
  - I dettagli delle operazioni di caricamento e salvataggio sono descritti nell'interfaccia IMovidaDB

Esempio

```
Title: Cape Fear
Year:  1991
Director: Martin Scorsese
Cast: Robert De Niro, Nick Nolte, Jessica Lange, Juliette Lewis
Votes: 163093

Title: Taxi Driver
Year:  1976
Director: Martin Scorsese
Cast:  Robert De Niro, Jodie Foster, Cybill Shepherd, Albert
Brooks
Votes: 684728

Title: Pulp Fiction
Year: 1994
Director: Quentin Tarantino
Cast: John Travolta, Uma Thurman
Votes: 1743616
```

IMovidaDB

```
public interface IMovidaDB {
	public void loadFromFile(File f);
	public void saveToFile(File f);
	public void clear();
	public int countMovies();
	public int countPeople();
	public boolean deleteMovieByTitle(String title);
	public Movie getMovieByTitle(String title);
	public Person getPersonByName(String name);
	public Movie[] getAllMovies();
	public Person[] getAllPeople();
}
```

Scelta algoritmi e strutture dati

- Ogni gruppo deve **implementare**:
  - due algoritmi di ordinamento
  - due realizzazioni della struttura dati dizionario
- Al momento della registrazione ad ogni gruppo sono assegnati gli algoritmi e i dizionari da sviluppare
- Non è sufficiente includere le strutture dati e gli algoritmi di Java Collections o della libreria *asdlab*
  - Si può prendere spunto…

Scelta algoritmi e strutture dati

- `MovidaCore` espone i metodi per selezionare l'algoritmo e il dizionario da usare a run-time, descritti nell'interfaccia `IMovidaConfig`
  - `setSort(…) e setMap(…)`
- Le possibili scelte sono espresse tramite le enumeration `MapImplementation` e `SortingAlgorithm`
- Ogni gruppo dovrà quindi supportare solo i valori corrispondenti ai due algoritmi/dizionari assegnati e gestire il caso in cui il valore in input non è tra questi

IMovidaConfig e opzioni

```
public interface IMovidaConfig {

	public void setSort(SortingAlgorithm a);

	public void setMap(MapImplementation m);
}
```

```
public enum MapImplementation {
	ArrayOrdinato,
	ListaCollegataNonOrdinata,
	ABR,
	AVL,
	Alberi23,
	BTree,
	HashConcatenamento,
	HashIndirizzamentoAperto
}
```

```
public enum SortingAlgorithm {
	InsertionSort,
	SelectionSort,
	BubbleSort,
	MergeSort,
	QuickSort,
	HeapSort
}
```

Ricerche

- Dopo aver caricato le informazioni in un'istanza di `MovidaCore` è possibile interrogare la knowledge-base invocando i metodi dell'interfaccia `IMovidaSearch`
- `MovidaCore` usa l'algoritmo e il dizionario attivo (ultima configurazione)
- Il gruppo è libero di scegliere come integrare le parti sviluppate, quando invocare i metodi, e come strutturare e memorizzare le informazioni all'interno della classe `MovidaCore`

IMovidaSearch

```
public interface IMovidaSearch {

	public Movie[] searchMoviesByTitle(String title);

	public Movie[] searchMoviesInYear(Integer year);

	public Movie[] searchMoviesDirectedBy(String name);

	public Movie[] searchMoviesStarredBy(String name);

	public Movie[] searchMostVotedMovies(Integer N);

	public Movie[] searchMostRecentMovies(Integer N);

	public Person[] searchMostActiveActors(Integer N);

}
```

Semplificazioni

- Il progetto è chiaramente semplificato rispetto ad un'applicazione reale ed è ritagliato su questo corso
- In particolare:
  - Una persona è identificata univocamente dal nome (completo), non sono gestite omonimie
  - Un film è identificato univocamente dal titolo (normalizzato)
  - Non è necessario gestire la persistenza su un database ma è sufficiente caricare/serializzare la knowledge-base in blocco sul filesystem
  - Non è necessario gestire l'update di singoli record o record collegati

Specifiche aperte

- Queste specifiche sono volutamente incomplete
- Aggiungeremo una o più interfacce per descrivere operazioni su reti (grafi) costruite a partire dai dati di MOVIDA
- Inoltre, vi invito a segnalare eventuali imprecisioni o miglioramenti **entro il 13 maggio**
- Se necessario pubblicheremo una versione aggiornata delle interfacce che recepisce i suggerimenti
  - modifiche minime, la struttura resta invariata

Come organizzare i file del progetto

- Il package `movida.commons` contiene le interfacce ed enumerazioni comuni
- Il progetto deve essere contenuto in un package separato che include la classe `MovidaCore`
- Usare come nome del package la concatenazione dei cognomi dei membri del gruppo, tutto in minuscolo, senza spazi, apostrofi, accenti
- Esempi:
- `<Zavattaro, Di Iorio>` → `zavattarodiiorio`
- `<D'Annata, Olè>` → `dannataole`

Come registrare un gruppo

- Compilare il form su: http://diiorio.nws.cs.unibo.it/asd1920/progetto/
- Riceverete una mail con i dati e gli algoritmi/dizionari assegnati

Valutazione del progetto

- La valutazione del progetto tiene conto sia del codice consegnato che della presentazione
- Un punto importante: deve essere frutto del vostro lavoro, dovete essere in grado di spiegarlo e modificarlo il giorno della consegna

Domande?

- Se ho seguito le lezioni l'anno scorso devo fare il progetto dell'anno scorso o quello di quest'anno?
  - Bisogna consegnare il nuovo progetto, con le nuove regole
- Se ho già un voto al progetto dagli anni scorsi devo consegnare il nuovo progetto?
  - No, i voti sono ancora validi per questo anno accademico

Domande?

- Il progetto va consegnato prima dello scritto o nella stessa sessione?
  - No, le due prove sono separate e contribuiscono al voto finale (media pesata)

- Altre domande?

PROGETTO MOVIDA – ESTENSIONE GRAFI

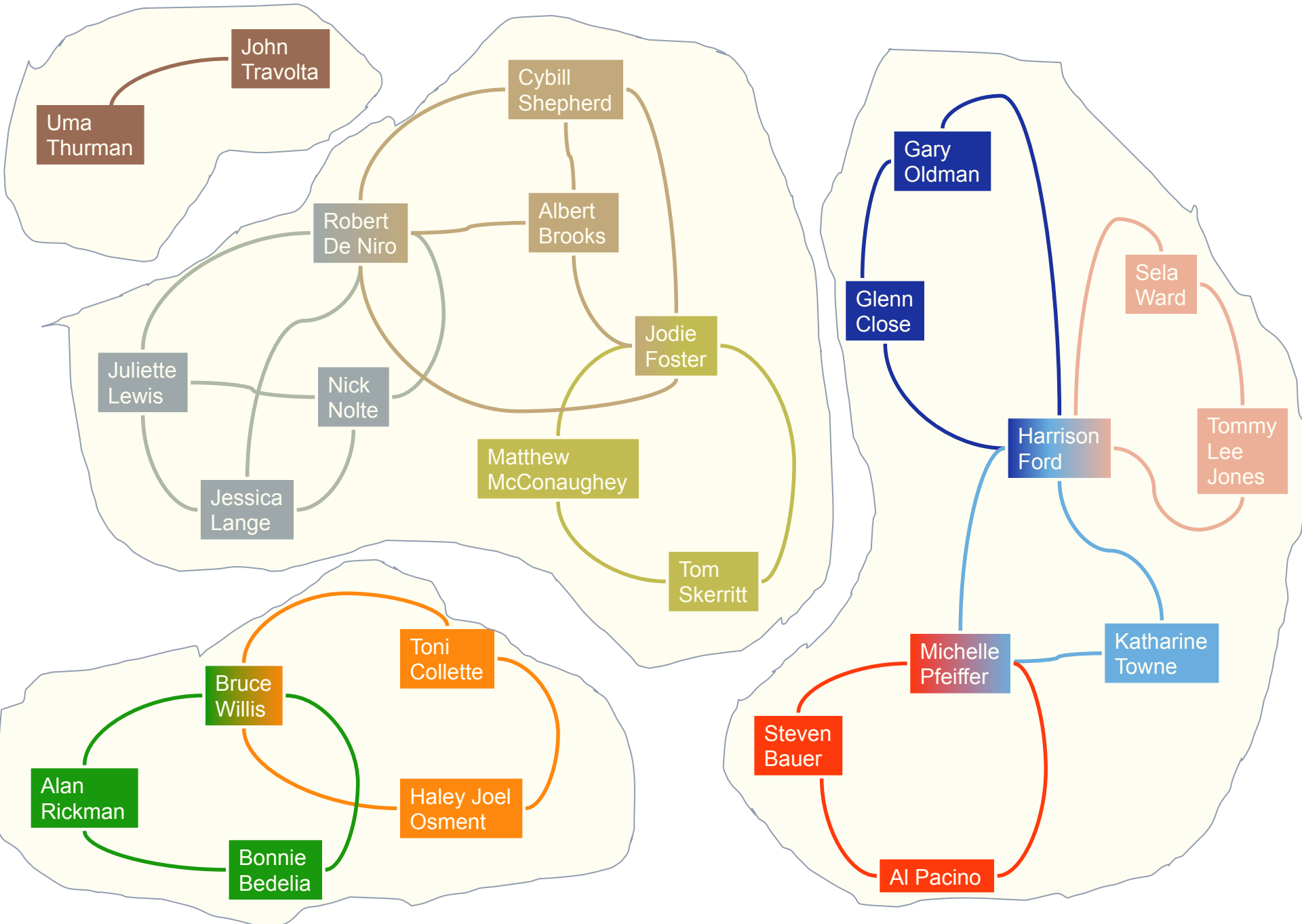
Angelo Di Iorio
Università di Bologna

Collaborazioni in MOVIDA

- MOVIDA permette di osservare le collaborazioni tra attori, intese come partecipazioni agli stessi film
- E' possibile identificare gruppi di attori che hanno collaborato tra loro
- Possiamo assumere, per semplicità, che tutti gli attori di un gruppo abbiano lo stesso produttore cinematografico
- Un produttore ha la necessità di organizzare al meglio le collaborazioni in un gruppo di attori per massimizzarne il successo

Terminologia

- Introduciamo la seguente terminologia:
  - Due attori A e B sono **collaboratori diretti** se hanno partecipato allo stesso film
  - Due attori A e B sono **collaboratori indiretti** se è possibile trovare una lista di attori, con A come primo elemento e B ultimo, per cui ogni elemento è un collaboratore diretto del suo successore (e quindi predecessore)
  - Il **team** di un attore è dato da se stesso e da tutti i suoi collaboratori indiretti
    - I team sono quindi disgiunti
- La slide seguente mostra le collaborazioni, dirette e indirette, e i team costruiti sui dati forniti all'inizio del progetto (file `esempio-formato-dati.txt`)


John Travolta
Uma Thurman
Cybill Shepherd
Robert De Niro
Albert Brooks
Jodie Foster
Juliette Lewis
Nick Nolte
Jessica Lange
Matthew McConaughey
Tom Skerritt
Gary Oldman
Glenn Close
Sela Ward
Harrison Ford
Tommy Lee Jones
Michelle Pfeiffer
Katharine Towne
Steven Bauer
Al Pacino
Toni Collette
Bruce Willis
Alan Rickman
Haley Joel Osment
Bonnie Bedelia

Classe `Collaboration`

- Introduciamo una classe `Collaboration` che rappresenta appunto la collaborazione diretta tra due attori
- La classe memorizza i due attori e l'insieme di tutti i film in cui hanno collaborato
  - Nella realtà si potrebbe estendere con ulteriori informazioni sulla collaborazione, ad esempio premi vinti dagli attori, etc.
  - Per ogni coppia di attori esiste quindi una sola `Collaboration`
- Ogni collaborazione è caratterizzata da un punteggio (*score*)
- Per semplicità il punteggio è la media tra i voti di tutti i film in cui la coppia di attori ha collaborato
  - Anche qui si potrebbe usare uno score diverso ma l'impostazione generale non cambia

Classe `Collaboration`

- La classe `Collaboration` è stata aggiunta al package `movida.commons`
- La classe può essere estesa o modificata ma deve mantenere i metodi `get()` per accedere alle informazioni in lettura su attori e score
- Il metodo `getScore()` non deve essere modificato

```
public Person getActorA();

public Person getActorB();

public double getScore();
```

Collaborazioni caratteristiche di un team

- Vogliamo fornire al produttore un metodo per individuare un "*insieme di collaborazioni caratteristiche di un team*" (ICC) cioè un insieme **minimale** di collaborazioni che coinvolgono **tutti** i membri del team
- Per definizione ICC(T) soddisfa le seguenti proprietà:
  - se si elimina una collaborazione da ICC(T) non è possibile raggiungere, tramite collaborazione diretta o indiretta, tutti i membri del team T
  - non è possibile aggiungere una collaborazione a ICC(T) che collega due attori già presenti in collaborazioni di ICC(T)
- Si noti che se T include `K` attori allora ICC(T) ha `K-1` collaborazioni. Inoltre T può avere diversi insiemi di collaborazioni caratteristiche.
- `MovidaCore` permette di identificare l'insieme di collaborazioni caratteristiche di un team con lo *score complessivo* più alto

IMovidaCollaborations

- `MovidaCore` deve implementare anche l'interfaccia `IMovidaCollaborations` per:
  - Identificare i collaboratori diretti di un attore
  - Ricostruire il team di un attore
  - Identificare l'insieme di collaborazioni caratteristiche del team di un attore, che massimizza lo score complessivo (come descritto nella slide precedente)

```
public Person[] getDirectCollaboratorsOf(Person actor);

public Person[] getTeamOf(Person actor);

public Collaboration[]
maximizeCollaborationsInTheTeamOf(Person actor);
```

Note e vincoli

- Questa parte delle specifiche è comune a tutti i gruppi e non è necessario registrarsi nuovamente sul sistema di prenotazione dei progetti
- Ogni gruppo è libero di scegliere gli algoritmi che preferisce
- Non è richiesto re-implementare le strutture dati utili per i metodi (come nel caso di dizionari e ordinamento) ma è ammesso usare le strutture già disponibili in librerie esterne e Java Collections
- Tuttavia non è ammesso usare direttamente le implementazioni degli algoritmi su grafi della libreria `asdlab`