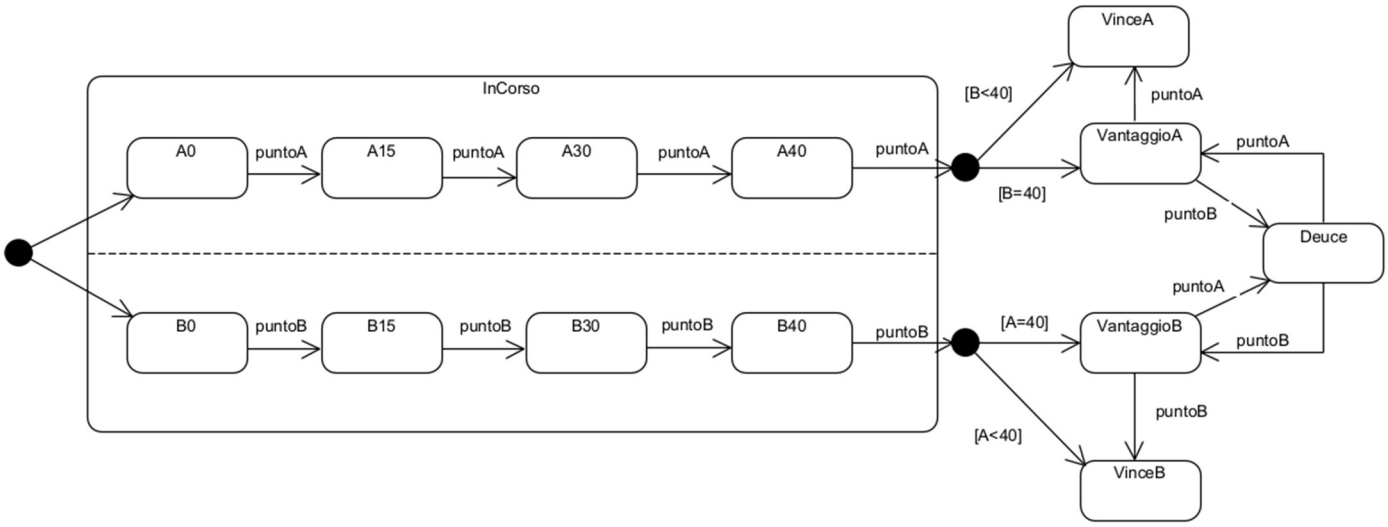
Gli internazionali di Roma

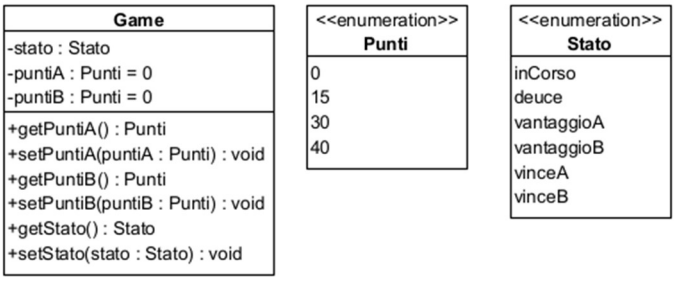
Ricordo le seguenti regole per determinare il vincitore di un game:

- Il punteggio avanza in questa sequenza: 0, 15, 30, 40.
- Il primo giocatore che arriva a 40 punti e poi fa un altro punto vince il game.
- Se il punteggio è 40-40, si entra in una situazione di parità chiamata "Deuce" (parità).
- In una situazione di Deuce, un giocatore deve guadagnare due punti consecutivi per vincere il game.
- In una situazione di Deuce il giocatore che fa un punto va in vantaggio. Se l'altro giocatore fa un punto si torna in situazione di Deuce.

Domanda 1 (6 punti)
Disegnare un diagramma di macchina a stati che descriva gli stati di un game di tennis tra i giocatori A e B. Si consiglia di usare uno stato composito parallelo.
Una possibile soluzione:



Domanda 2 (5 punti)
Date le classi a lato, il metodo `aggiornaGame(Game g, bool puntoA, bool puntoB)` aggiorna il Game dopo ogni punto.



Quante possibili combinazioni di valori di input, senza escludere combinazioni errate, ci possono essere?
Quale tecnica di riduzione del numero di combinazioni vi sembra più adatta? Perché?

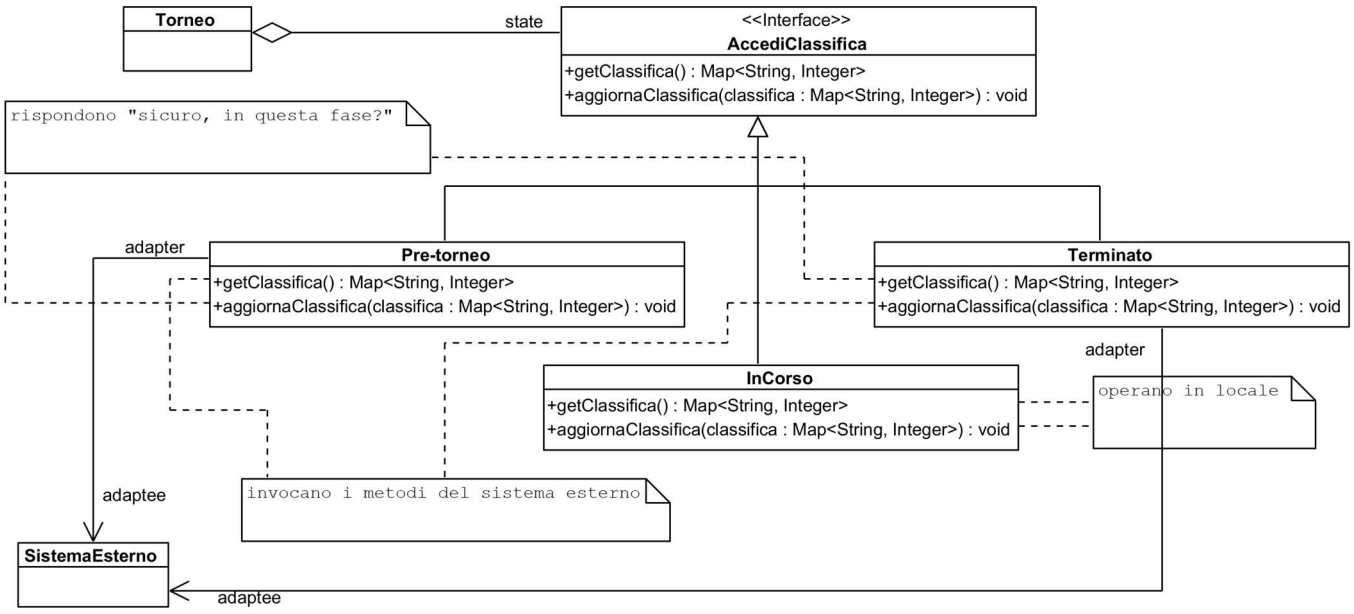
Una possibile soluzione:
6 (Game.stato) * 4 (Game.puntiA) * 4 (Game.puntiB) * 2 (puntoA) * 2 (puntoB)

Un modo per ridurre le combinazione è applicare dei vincoli, in particolare di errore (se Game.stato== vinceA/B, nessuno fa punto e non serve provare tutte le combinazioni di puntiA e puntiB) e di proprietà (vale puntoA xor puntoB; se Game.stato==vantaggioA/B o Deuche i punti di A e B sono 40).

Domanda 3 (8 punti) Per semplicità in questo esercizio si supponga esista solo la classifica ATP.

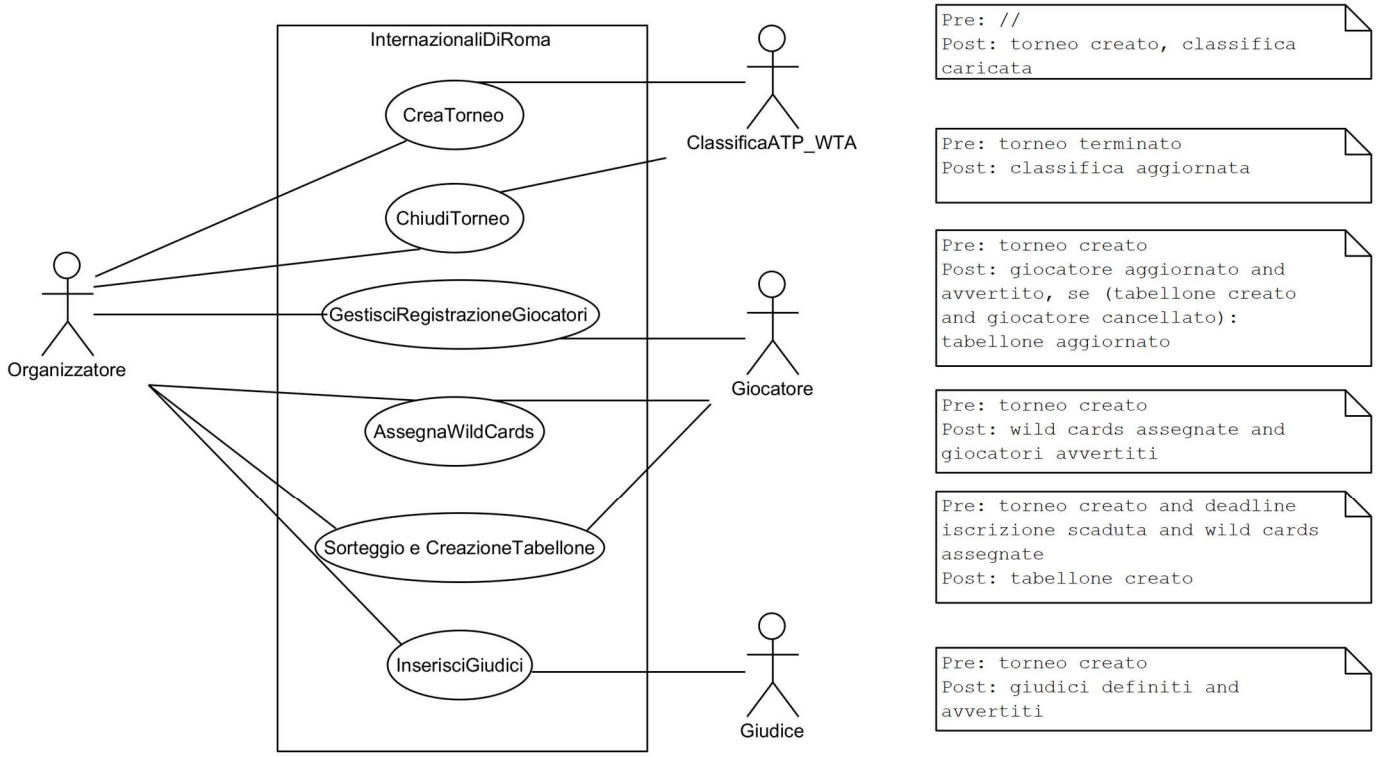
Sia data l'interfaccia `AccediClassifica` con i seguenti metodi:
`Map<String, Integer> getClassifica()`// per ottenere la classifica ATP
`void aggiornaClassifica(Map<String, Integer> classifica)`// per aggiornare la classifica ATP

Un oggetto di tipo `Torneo` può trovarsi in 3 stati: *pre-torneo, inCorso,* e *terminato*. Prima dell'inizio del torneo, il sistema deve richiedere i punti in classifica ATP a un sistema esterno; durante il torneo, aggiorna la classifica APT locale in base ai risultati degli incontri; alla fine del torneo il sistema deve inviare i punti aggiornati al sistema esterno. La classe `Torneo` invoca l'implementazione appropriata di `AccediClassifica` a seconda dello *stato* del torneo. Una di queste implementazioni deve fungere da *adapter* verso il sistema esterno.
Progettare un diagramma UML che, applicando i design pattern opportuni, rappresenti la struttura delle classi e le relazioni tra di esse e dare lo pseudocodice dei metodi, ove si ritenga utile per illustrare la soluzione proposta.
Una possibile soluzione:



Domanda 4 (6 punti) Fornire il diagramma dei casi d'uso che riguardano le fasi di preparazione del torneo (prima dell'inizio del torneo stesso, cioè prima dell'inizio del primo incontro) in cui un organizzatore è l'attore principale. Per ogni caso d'uso indicare pre- e post-condizioni.

Una possibile soluzione:



Domanda 5 (5 punti) Dare un diagramma delle classi che modelli le relazioni tra le seguenti classi: Torneo, Giocatore, Incontro, Giudice, Set, Game. Porre attenzione alle molteplicità. (L'incontro è diretto da un arbitro (o giudice di sedia) coadiuvato da giudici di linea (da 7 a 9) e un giudice di rete. Non serve contiate il numero di incontri nel torneo)

Una possibile soluzione: