Sapienza - Università di Roma
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Informatica
Esercitazioni di Progettazione del Software - A.A. 2008/2009
Prova al calcolatore del 19 febbraio 2010

Requisiti

Si vuole realizzare un'applicazione per il gioco del *Tris*. In una partita di *Tris* due giocatori si fronteggiano su una scacchiera contenente esattamente 9 celle, ciascuna associata ad una posizione nella scacchiera. La cella in alto a sinistra ha posizione $x = 1, y = 1$, la coordinata x è crescente da sinistra verso destra e la coordinata y è crescente dall'alto verso il basso. La disposizione delle celle in una scacchiera, con le relative posizioni, è mostrata in Figura 1(b). Una cella può essere vuota, contenere il simbolo "X" o il simbolo "O".

Inizialmente, tutte le celle sono vuote (contengono un simbolo speciale diverso da "X" e "O"). Una partita è suddivisa in turni. Il primo giocatore seleziona una cella vuota che sarà riempita con il simbolo "X"; quindi, l'avversario risponde, selezionando una tra le celle rimaste vuote, che sarà riempita con il simbolo "O". Questo processo viene ripetuto finché:

- la scacchiera contiene una riga, una colonna o una diagonale le cui celle sono riempite con lo stesso simbolo. Se tale simbolo è "X" allora il giocatore che ha mosso per primo vince, altrimenti vince l'altro;
- la scacchiera non contiene più celle vuote, e la partita termina in parità.

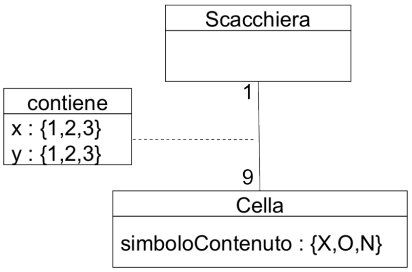
L'applicazione deve:

- creare una nuova scacchiera e le relative celle (inizialmente vuote);
- iterare i seguenti passi, finché la partita non è terminata:
  - dare la possibilità al giocatore che deve muovere di selezionare la cella (vuota) che vuole riempire;
  - aggiornare lo stato della cella selezionata, riempiendola con il simbolo opportuno (sulla base del turno corrente);
  - verificare se la partita è terminata;
- al termine della partita, annunciare il vincitore o la situazione di parità, visualizzando un messaggio sullo schermo.

In Figura 1(c) è riportato il diagramma delle attività corrispondente al comportamento richiesto.

La prova consiste nel completare o modificare il codice fornito insieme al testo, in modo da soddisfare i requisiti sopra riportati. Seguendo le indicazioni riportate nei commenti al codice, si chiede di intervenire sulle seguenti classi:

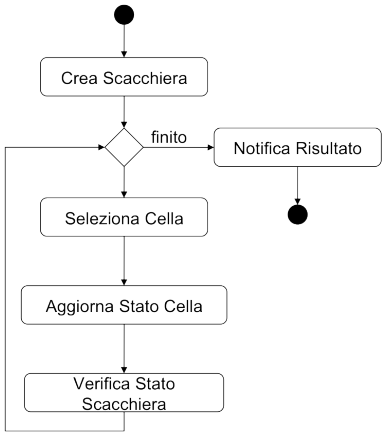
- `Applicazione` (package `gui`);
- `AttivitaPrincipale` (package `attivitacomposte`);



(a) Diagramma UML delle classi

1,1	2,1	3,1
1,2	2,2	3,2
1,3	2,3	3,3

(b) Scacchiera per il gioco dl *Tris*



(c) Diagramma UML delle attività

- `AggiornaStatoCella` (package `attivitaatomiche`);
- `CreaScacchiera` (package `attivitaatomiche`);
- `Scacchiera` (package `tris`);

Tempo a disposizione: **3 ore.**

Gli elaborati non accettati dal compilatore saranno considerati insufficienti.

Per facilitare la comprensione del codice e lo svolgimento della prova, nel seguito sono riportati i documenti di specifica risultanti dalle fasi di analisi e di progetto.

Analisi

Operazioni Utente

Non ci sono operazioni utente da realizzare.

Segnatura delle Attività di I/O

```
InizioSpecificaOperazioniIO AttivitaPrincipale

  operazione: SelezionaCella
    -- Legge le coordinate della cella da riempire, fornite in input dall'utente
    in:  simbolo : SimboloCella -- il simbolo corrispondente al giocatore che deve effettuare la mossa
    out: coordinate : RecordCoordinate


  operazione: NotificaRisultato
    -- Legge il simbolo corrispondente al vincitore (e lo stampa in output)
    in: vincitore : SimboloCella
    out: --
FineSpecifica
```

NOTE:
`RecordCoordinate` è un record con due campi, `x` e `y`, usati per memorizzare la posizione della cella selezionata.
`SimboloCella` è un il tipo associato ai simboli che possono essere inseriti in una cella.

Definizione delle Variabili dell'Attività Principale

```
InizioSpecificaVariabiliAttività AttivitaPrincipale
  laScacchiera: Scacchiera -- scacchiera su cui si gioca la partita gestita dall'attività
  prossimo : SimboloCella -- simbolo (X o O) del giocatore che deve effettuare la prossima mossa
  finale : Boolean -- diventa true quando  la scacchiera è in una configurazione finale
  vincitore : SimboloCella -- simbolo (X o O o VUOTO) del giocatore che ha vinto
                              (VUOTO se la partita non è finita o è parità)
FineSpecifica
```

Definizione delle Attività Atomiche

```
InizioSpecificaAttività CreaScacchiera
    -- Crea la scacchiera su cui giocare la partita, con le relative celle e posizioni
    -- e la restituisce tramite il metodo getScacchiera()
FineSpecifica

InizioSpecificaAttività AggiornaStatoCella
    -- Inserisce nella Scacchiera laScacchiera il SimboloCella simbolo nella cella la cui
    -- posizione è specificata dal RecordCoordinate c (restituito dall'attività di I/O SelezionaCella)
FineSpecifica

InizioSpecificaAttività VerificaStatoScacchiera
    -- 1. Verifica se la Scacchiera laScacchiera è in una configurazione finale
    --   e restituisce  il risultato tramite il metodo isConfigurazioneFinale()
    -- 2. se la configurazione è finale restituisce, tramite il metodo getVincitore(),
    --   il simbolo corrispondente al vincitore: X o O, oppure VUOTO se la partita è finita in parità
FineSpecifica
```

NOTE:
Altre eventuali attività presenti nell'implementazione possono essere trascurate.

Progetto

Responsabilità sulle Associazioni

R: Requisiti; O: Specifica delle Operazioni; M: Vincoli di Molteplicità

Associazione	Classe	Ha Responsabilità
contiene	Scacchiera	SÌ (O,M)
	Cella	SÌ (M)

Strutture di Dati

Rappresentiamo le collezioni omogenee di oggetti mediante le classi `Set` ed `HashSet` del Collection Framework di Java.

Tabelle di Gestione delle Proprietà delle Classi UML

Riassumiamo le scelte differenti da quelle di default mediante la tabella delle proprietà immutabili e la tabella delle assunzioni sulla nascita.

Classe UML	Proprietà Immutabile
-	-

	Proprietà	
Classe UML	**Nota alla nascita**	**Non nota alla nascita**
-	-	-

Altre Considerazioni

Non dobbiamo assumere una particolare sequenza di nascita degli oggetti

Non esistono valori di default per qualche proprietà che siano validi per tutti gli oggetti.