SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica ed in Ingegneria dei Sistemi Informatici

**Corso di Progettazione del Software**

Esame del **12 Settembre 2014**

*Tempo a disposizione: 3 ore*

**Requisiti.** L'applicazione da progettare riguarda una parte di un videogioco urban fantasy per tablet. Un gioco è costituito da diversi giocatori, ciascuno dei quali appartiene esclusivamente al gioco stesso. I giocatori hanno un nome (una stringa) ed una posizione sul piano di gioco (i dettagli di questo piano non interessano), data in coordinate cartesiane, e sono divisi in 3 categorie: i vampiri, i licantropi, e mutaforma. I licantropi sono organizzati in tribú e di ognuno di loro interessa la tribù di appartenenza (caratterizzata da un nome). I mutaforma assumono la forma di un altro giocatore vampiro o licantropo prendendone tutti i poteri. Durante il gioco possono cambiare forma secondo le proprie necessità, ma in ogni momento del gioco devono avere la forma di uno e un solo giocatore. Negli scontri i vampiri sono più forti dei licantropi isolati, ma se questi sono vicini (cioè sono a distanza cartesiana inferiore a 10 unità) ad altri licantropi della stessa tribù allora diventano più forti del vampiro. Anche i mutaforma possono essere attaccati ma non attaccano mai per primi. I dettagli di come si tengono i conteggi degli scontri vinti e degli scontri persi non interessano.

In questo compito siamo interessati al comportamento dei soli mutaforma. Un mutaforma è inizialmente a *riposo*. Se quando è a *riposo* riceve un *attacco* da un altro giocatore passa allo *scontro*. Se il giocatore che lo attacca è un vampiro allora verifica se ci sono altri licantropi vicini e in tal caso prende la forma di un membro della tribù del licantropo più vicino e *attacca* a sua volta il vampiro. Se il giocatore che lo attacca è un licantropo isolato (tutti i sui compagni di tribù sono a più di 10 unità di distanza da lui) allora si trasforma in un vampiro e lo *attacca* a sua volta. Se invece il licantropo che lo attacca è vicino ad altri membri della sua tribù, allora assume a sua volta la forma di un membro della stessa tribú, chiedendo a chi lo ha attaccato la *pace*. Quando l'attacco è *finito* torna a *riposo*.

Siamo interessati alla seguente attività principale. L'attività prende in input un gioco $G$ e concorrentemente esegue le seguenti due sottoattività: *(i)* gioca, e *(ii)* analisi. La sottoattività di gioco *(i)* avvia il gioco attivando tutti i giocatori di $G$ mandando opportuni eventi (i dettagli non interessano). Poi si mette in attesa del comando di fine-gioco da parte dell'utente che interrompe il gioco. La sottoattività di analisi *(ii)* calcola la percentuale di vampiri, licantropi e mutaforma del gioco. Una volta che tali sottoattività sono state completate, si chiede ai giocatori di valutare il gioco appena giocato e si stampano le percentuali calcolate dalla sottattività di analisi.

**Domanda 1.** Basandosi sui requisiti riportati sopra, effettuare l'analisi producendo lo schema concettuale in UML per l'applicazione, comprensivo del diagramma delle classi (inclusi vincoli non esprimibili in UML), diagramma stati e transizioni per la classe *Mutaforma*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica dell'attività principale e delle sottoattività NON atomiche (indicando in modo esplicito quali attività atomiche sono di I/O e quali sono Task), motivando, qualora ce ne fosse bisogno, le scelte di progetto.

**Domanda 2.** Effettuare il progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte di progetto. È obbligatorio definire solo le responsabilità sulle associazioni del diagramma delle classi.

**Domanda 3.** Effettuare la realizzazione, producendo un programma JAVA e motivando, qualora ce ne fosse bisogno, le scelte di progetto. È obbligatorio realizzare in JAVA solo i seguenti aspetti dello schema concettuale:

- La classe `Giocatore`, la classe `Licantropo` (senza rappresentazione del diagramma stati e trasizioni, visto che non è noto), la `Mutaforma` con classe `MutaformaFired`, e le classi JAVA per rappresentare le *associazioni* di cui `Licantropo` e `Mutaforma` hanno responsabilità.
- L'*attività principale* (NON le sue sottoattività).

Si può assumere di avere una funzione ausiliaria privata che dati due giocatori ne calcola la distanza.