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 **19 Giugno 2012**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di un gioco basato su blocchi virtuali attivi componibili. Una configurazione è costituita da un nome (una stringa) e da un insieme di blocchi, ciascuno appartenente alla sola propria configurazione. Ogni blocco ha un codice (una stringa) ed è collegato a un insieme di blocchi in ingresso ed un insieme ordinato di blocchi in uscita. Sia i blocchi in ingresso che in uscita devono appartenere alla stessa configurazione del blocco in questione. I collegamenti sono unidirezionali. Nella configurazione si distinguono l'insieme dei blocchi iniziali e finali. Si noti che, tranne che per il ruolo specifico nella configurazione, tali blocchi sono esattamente come gli altri.

Un blocco è inizialmente spento e può ricevere l'evento *toggle* da uno dei blocchi in ingresso. A fronte di tale evento si accende e manda a sua volta un evento *toggle* a uno dei suoi blocchi in uscita non ancora accesi, se ne ha (la funzione di scelta del blocco si può assumere data). Un blocco acceso può ricevere anch'esso un evento *toggle*, e a fronte di tale evento si spegne mandando un evento *toggle* a tutti i suoi blocchi in uscita (si noti manda molteplici eventi).

Siamo interessati alla seguente attività che prende come parametro una configurazione senza blocchi, legge da input un insieme di blocchi (per esempio 20), e aggiunge i blocchi alla configurazione collegando tra loro i blocchi in modo arbitrario (secondo una funzione che si può assumere data). L'attività poi procede concorrentemente con le seguenti due sottoattività, una di gioco *(i)* e una di verifica *(ii)*. La sotto attività di gioco *(i)* inizia il gioco, attraverso l'invio a tutti i blocchi iniziali dell'evento *toggle* seguendo una sequenza di attivazione arbitraria e ci si mette in attesa (attraverso un'opportuna attività di input/output) del comando di fine esecuzione da parte dell'utente, che interrompe l'esecuzione del gioco. La sottoattività di verifica *(ii)* calcola e restituisce in output l'insieme dei blocchi non iniziali che non hanno blocchi in ingresso. Una volta che tali sottoattività sono completate, stampa un messaggio di saluto e termina.

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare la fase di 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 *Blocco*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica della attività principale e 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 effettuate.

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

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

- La classe *Blocco* (con eventuale classe *BloccoFired*) e le eventuali *associazioni* che legano la classe *Blocco* a se stessa.
- L'*attività principale*, e le *sottoattività NON atomiche*.