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

Fondamenti di Informatica II: Progettazione del Software

Esame del **13 febbraio 2017**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di grafi orientati colorati connessi. Un grafo è costituito da un nome (una stringa) e da un nodo che costituisce il suo nodo iniziale. Ogni nodo ha un codice (una stringa) ed è collegato a un insieme di nodi successori. Tali collegamenti sono gli archi orientati del grafo. I nodi sono partizionati in due soli tipi: *rossi* e *blu*. Per i primi interessa la gradazione (una stringa), mentre per i secondi interessa l'intesità (un intero). Ogni nodo rosso ha tra i suoi successori esattamente un nodo speciale blu. Similmente ogni nodo blu ha tra i suoi successori esattamente un nodo speciale rosso.

Un nodo n è inizialmente spento. Quando un nodo n spento riceve l'evento *toggle*, si accende e manda a sua volta un evento *toggle* ad una selezione di nodi successori. Per scegliere a quali nodi successori mandare l'evento *toggle*, conta i nodi di cui è successore (i predecessori): se n è rosso e i suoi predecessori sono in maggioranza blu manda l'evento ai suoi successori blu, altrimenti lo manda a tutti i suoi successori rossi; se n è blu si comporta allo stesso modo ma invertendo i colori. Quando un nodo acceso riceve l'evento *toggle* si spegne.

Siamo interessati alla seguente attività che prende come parametro un grafo, calcola l'insieme dei codici del nodo iniziale e dei suoi successori (diretti) e lo stampa, indicando quali tra questi è il nodo speciale. 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'inizializzazione dell'environment e l'invio al nodo iniziale della componente connessa dell'evento *toggle* e poi si mette in attesa del comando di fine esecuzione da parte dell'utente (attraverso un'opportuno segnale di input), che interrompe l'esecuzione del gioco. La sottoattività di verifica *(ii)* calcola il numero di nodi rossi e di nodi blu che formano il grafo visitandolo a partire dal nodo iniziale. Una volta che tali sottoattività sono completate, stampa un messaggio di saluto riportando il numero di nodi rossi e di nodi blu calcolato dalla precedente verifica 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 *Nodo*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica della 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 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 `Nodo`, con classe `NodoFired`, le sue *sottoclassi*, e le classi per rappresentare le *associazioni* (con eventuali ruoli) di cui `Nodo` e le sue sottoclassi sono responsabili.
- L'*attività principale*.