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 **18 Febbraio 2014**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una versione semplificata del gioco Cloud & Sheep 2.0 per tablet e smartphone. Un gregge ha un nome (una stringa) ed è costituito da un insieme di pecore (più di due) di cui una è la pecora guida. Una pecora ha un livello di felicità (un intero positivo incluso lo zero). Le pecore si distinguono in quelle di genere maschile e femminile. Le pecore di genere opposto dello stesso gregge possono formare una famiglia, costituita da un *lui* (pecora di genere maschile), una *lei* (pecora di genere femminile) e un insieme di figli (altre pecore di qualsiasi genere) anche essi dello stesso gregge. Ogni pecora può essere al più un *lui*/una *lei* di una sola famiglia.

Una pecora è inizialmente in uno stato di attesa. Quando è in attesa e riceve l'evento di *inizio-gioco* si pone nello stato di gioco. In questo stato può ricevere tre comandi, *salta*, *innamorati*, *riproduciti*. Il comando *salta* serve a far giocare la pecora ed aumenta di 1 il suo livello di felicità. Il comando *innamorati* contiene come parametro una pecora del genere opposto, e se entrambe le pecore fanno parte dello stesso gregge, hanno un livello di felicità superiore a 5 e non sono già un *lui*/una *lei* di una famiglia, costituiscono una famiglia con un insieme vuoto di figli, altrimenti non succede nulla. Il comando *riproduciti* se ha come destinatario una *lui* o una *lei* di una famiglia, crea e aggiunge un figlio alla famiglia ed al gregge dei genitori, con livello di felicità pari alla media dei livelli del padre e della madre, altrimenti non succede nulla. Dallo stato di gioco, con l'evento *fine-gioco*, si torna nello stato di attesa.

Siamo interessati alla seguente attività principale. L'attività prende in input un gregge e inizia attivando concorrentemente le seguenti due sottoattività: *(i)* gioca, e *(ii)* analisi. La sottoattività di gioco *(i)* avvia il gioco inviando l'evento inizio-gioco a tutte le pecore. Poi si mette in attesa del comando di fine-gioco da parte dell'utente che interrompe il gioco. La sottoattività di analisi *(ii)* calcola il genere della pecora guida e lo stampa. Una volta che tali sottoattività sono state completate, si calcola e si stampa il numero di pecore del gregge che non sono una *lui*/una *lei* di una famiglia.

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 *Pecora*, 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 `Pecora` con classe `PecoraFired`, le sottoclassi e le classi JAVA per rappresentare le eventuali *associazioni* che legano queste classi alla classe `Famiglia`.
- L'*attività principale* e le sue eventuali sottoattività NON atomiche.