SAPIENZA Università di Roma
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Informatica
Corso di Progettazione del Software
Esame del **23 giugno 2009**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una variante del gioco dell'oca in cui si inizia in caselle qualsiasi, diverse caselle fanno terminare il gioco non avendo successori, e una stessa casella può essere la casella successiva di diverse caselle. Più precisamente, ogni casella ha un disegno (una stringa che denota il file jpg) ed ha al più una casella "successiva". Le caselle sono suddivise in caselle "normali" e caselle "salto". Le caselle "salto" sono legate ad esattamente una casella: la casella in cui si salta (che a sua volta può essere una casella "normale" o una casella "salto"). Le caselle possono essere occupate dai giocatori, ogni giocatore è caratterizzato dal nome (una stringa) e da un "avatar" (una piccola immagine rappresentata da una stringa che denota il file "animated jpg" che si giustappone al disegno della casella) e occupa al più una casella. Nelle caselle ci possono essere anche molti giocatori contemporaneamente, ma solo per le caselle normali è importante ricordare il giocatore che ci è entrato per primo.

Siamo interessati a progettare l'attività di preparazione del gioco, che prende come parametro un insieme di caselle contenente almeno una casella "normale" e fa quanto segue. Inizialmente verifica che ogni casella dell'insieme ha come casella "successiva" una casella ancora appartenente all'insieme. Se la verifica va a buon fine si procede altrimenti si termina con un messaggio d'errore. Poi contemporaneamente si procede a *(i)* verificare che ogni casella "salto" nell'insieme abbia come casella in cui si salta una casella ancora appartenete all'insieme; *(ii)* posizionare i giocatori nelle caselle normali. Per fare ciò si legge da input quanti giocatori inserire, e per ognuno si crea l'oggetto giocatore e lo si inserisce in una casella "normale" tra quelle dell'insieme, ricordandosi, nel caso una casella sia occupata di più di un giocatore, il giocatore posizionato per primo. Una volta terminate entrambe queste due sottoattività si procede a stampare un messaggio di preparazione avvenuta, se entrambe sono andate a buon fine, altrimenti si stampa un messaggio d'errore.

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, diagramma delle attività, specifica delle attività atomiche che fanno side effect sul diagramma delle classi (la specifica delle attività di I/O è opzionale), 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 le responsabilità su tutte le associazioni del diagramma delle classi ed il progetto dell'algoritmo dell'*attività atomica di verifica delle caselle "salto"*.

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:

- Le classi *Casella* e *Giocatore*, e le eventuali *associazioni* che le legano (non le eventuali sottoclassi).
- L'*attività principale*, l'*attività atomica di verifica delle caselle "salto"*. Le altre sottoattività non vanno realizzate.