SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica
Corso di Progettazione del Software
Esame del **19 febbraio 2021**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione riguarda un videogioco sul mondo del crimine. Una gilda ha un nome ed è composta da un certo numero di assassini, minimo 3. Un assassino è una persona (vedi dopo) e appartiene a esattamente una gilda. A un assassino può essere commissionato un tentativo di omicidio con un codice (una stinga), una persona che è la vittima ed un'altra persona che è il committente. Un omicidio ha due campi booleani per memorizzare rispettivamente se concluso o meno e in caso sia concluso se è riuscito o fallito. Se è concluso e riuscito allora è di interesse memorizzare il tempo impiegato per eseguirlo, in secondi. Tutte le persone hanno nome, cognome e indirizzo, e tra queste gli assassini hanno anche un'arma preferita (una stringa), e gli assassini esperti hanno anche una seconda arma.

Siamo interessati al comportamento dell'assassino. Un assassino è inizialmente a *riposo*. Se riceve dalla sua gilda un messaggio *uccidi* con payload un omicidio commissionato a se stesso ed il tempo corrente, notifica la gilda con un messaggio e si mette nello stato *missione*. Quando riceve l'evento *fine* inviato dalla propria gilda con payload il tempo corrente e l'esito (omicidio fallito o riuscito) aggiorna le informazioni calcolando, se riuscito, il tempo impiegato nell'omicidio e torna a *riposo*. Per calcolare il tempo impiegato eseguire la differenza tra il tempo corrente nel evento *uccidi* e il tempo corrente nell'evento *fine*.

Siamo interessati alla seguente attività principale che prende come parametro un insieme di omicidi. Questa attività prima verifica che nessun committente di un omicidio nell'insieme sia anche la vittima di un omicidio dell'insieme. Se questa verifica fallisce manda un segnale di errore e termina. Altrimenti, esegue concorrentemente due sottoattività: nella prima vengono tentati tutti gli omicidi, nella seconda vengono richiesti i pagamenti ai committenti. I dettagli di queste due sottoattività non interessano, salvo il fatto che la prima attività produce il sottoinsieme degli omicidi riusciti e la seconda il totale dei soldi raccolti. Quando entrambe le sottoattività si concludono, si esegue un'attività che calcola la differenza fra i soldi raccolti e quelli previsti come pagamento degli omicidi riusciti e la si stampa.

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 *Assassino*, diagramma delle attività, specifica del diagramma stati e transizioni, riportando solo gli stati, e le variabili di stato ausiliarie, ma non la specifica delle transizioni; la segnatura delle attività complesse, delle attività atomiche e dei segnali di input/output. Motivare, qualora ce ne fosse bisogno, le scelte di progetto.

Domanda 2. Effettuare la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate. In particolare definire SOLO le responsabilità sulle associazioni del diagramma delle classi (nella tabella, inserire anche il motivo di ognuna delle responsabilità).

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 *Assassino* (con classe *AssassinoFired*), eventuali *superclassi* e *sottoclassi* e le classi per rappresentare le *associazioni* di cui la classe *Assassino* è responsabile.
- L'*attività principale*, e la sua *attività atomica di verifica*.