Una software house sta sviluppando un software di simulazione a supporto di una organizzazione di che svolge attività di propaganda, con lo scopo di stimare i costi ed i benefici di una campagna di informazione pagando degli incaricati. Nello specifico si vuole studiare quanti "contatti" avvengono tra persone che vengono assoldate con lo specifico obiettivo di massimizzare i contatti con altre persone (persone definite *Predicatori*) rispetto alle altre persone.

Allo stato corrente, il codice prevede la presenza in un solo tipo di persone, senza alcun particolare comportamento: si muovono confinate all'interno di un ambiente (modellato dalla classe **`prop.sim.Ambiente`**) e si spostano occupando delle celle scelte casualmente, anche coincidenti, disposte su posizioni (modellate dalla classe **`prop.sim.Coordinate`**) di un piano cartesiano.

DOMANDA 1 (5%)
Modificare il codice della classe **`prop.sim.Coordinate`** affinche' i test presenti nelle classi **`CoordinateTest`** comincino ad avere successo. Gia` dopo aver effettuato questa correzione, e` possibile verificare il corretto funzionamento dell'intera simulazione eseguendo il metodo **`main()`** della classe **`prop.Main`**.

(N.B. Per una piu' agevole comprensione della descrizione che segue, si consiglia di provare ad eseguire il metodo `main()` della classe `prop.sim.Main`, osservare l'animazione della simulazione gia` dopo aver risposto a questa prima domanda, e premere il tasto ESCape dopo qualche secondo per fermare la simulazione. *La simulazione stampa, a fine esecuzione, alcune statistiche. Queste saranno oggetto delle domande successive: e' possibile premere il tasto ESCape per anticipare la fine della simulazione e la stampa delle statistiche in qualsiasi momento, anche senza attendere la terminazione dell'intera simulazione.)*

Ciascuna persona possiede una posizione di partenza e ad ogni passo della simulazione (vedi il metodo **`prop.pers.Persona.mossa()`**) si sposta in una cella adiacente. Nel codice fornito la scelta della cella su cui spostarsi e', al momento, ancora casuale. Quando più persone occupano la medesima posizione viene modellato il *contatto* tra di esse tramite la creazione di oggetti di tipo **`prop.sim.Contatto`**, conservati per il calcolo delle statistiche finali (vedi metodo **`prop.sim.Simulatore.simula()`**). Tutte le persone che sono coinvolte in tale contatto vengono registrate nell'oggetto stesso di tipo **`Contatto`**.

Un progettista esperto fa notare che, nell'ottica di modellare i diversi tipi di comportamenti delle persone, conviene introdurre nuove classi (**`Predicatore`** e **`NonPredicatore`**), per poi ristrutturare il codice trasformando la classe **`Persona`** in una classe astratta che le generalizzi entrambe**.**

DOMANDA 2 (50%)
Pertanto il progettista esperto suggerisce di ristrutturare l'applicazione come segue:

a) **(25%)** Ridenominare la classe **`Bianca`** in **`NonPredicatore`** e la classe **`Rossa`** in **`Predicatore`**. Rifattorizzare contestualmente il codice introducendo la classe **`prop.pers.Persona`**, superclasse di entrambe le precedenti. Assicurarsi in questo modo che i predicatori abbiano l'icona rossa (vedi anche **`prop.gui.CostantiGUI`**) e i rimanenti quella bianca. Ogni persona deve possedere un identificatore ("**`id`**") progressivo intero assegnato base 0 (0, 1, 2, …) dipendentemente dalla propria tipologia. Verificare ed eventualmente correggere il codice principale ed i test presenti nella classe **`prop.persone.PersonaTest`** affinché abbiano sempre successo e verifichino la semantica desiderata per l'assegnazione degli id. Implementare le modifiche suggerite dal progettista esperto, quindi verificare il funzionamento dell'applicazione anche dopo le modifiche. Per far ciò e' necessario modificare la popolazione iniziale della simulazione per farvi apparire sia predicatori che non predicatori, i secondi in numero quadruplo rispetto ai primi, come specificato nella costante **`prop.sim.CostantiSimulazione.NUMERO_INIZIALE_PER_TIPOLOGIA`**.

b) **(25%)** Assegnare ad ogni tipo di persona una nuova strategia per decidere il proprio movimento. Modificare il metodo **`mossa()`** di modo che i *predicatori* si muovano (una cella alla volta) sempre le verso persone piu' lontane che non siano del loro stesso tipo (le informazioni riguardo le celle della simulazione e da chi sono occupate si trovano nella classe **`prop.sim.Ambiente`**). Le altre persone invece si muovono ("fuggono") nella cella adiacente più lontana a persone del tipo opposto. In presenza di un contatto con un predicatore, una persona (che non sia a sua volta un predicatore) potrà venire "convinta" con probabilità **`prop.gui.CostantiGUI.PROBABILITA_CONVERSIONE`** (vedere anche il metodo **`GeneratoreCasuale.siVerificaEventoDiProbabilita()`**). Le persone che sono state "convinte" dopo il contatto con un predicatore verranno mostrate con un icona gialla pur rimanendo di tipo dinamico **`NonPredicatore`**.
Suggerimento: notare che gli oggetti **`Contatto`** sono creati nel metodo **`prop.sim.Simulatore.eseguiPassoDellaSimulazione();`** nella versione iniziale del codice, dopo ogni passo della simulazione, viene invocato il metodo **`Bianca.avvenuto(Contatto)`** per notificare agli oggetti che modellano le persone i contatti che li vedono direttamente coinvolte.

Le domande che seguono richiedono il completamento del corpo di alcuni metodi nella classe **`prop.stats.Statistiche`**: questi sono dedicati al calcolo delle statistiche prodotte al termine di ciascuna simulazione. Sono anche gia` forniti a supporto dei metodi di stampa dei loro risultati per facilitarne la verifica del corretto funzionamento. Si suggerisce di studiare il sorgente della classe **`prop.stat.Statistiche`** ed in particolare il metodo **`stampaFinale()`** per i dettagli.

DOMANDA 3 (25%)
Dopo aver completato il punto precedente:

- completare il metodo **`Map< Object, SortedSet< Object > > produciStatistiche (List< Contatto >)`** nella classe **`Statistiche`**. In particolare questo metodo deve scandire la lista degli oggetti di tipo **`Contatto`** che riceve come parametro, e generare una mappa che metta in relazione il passo della simulazione con l'insieme dei contatti che sono avvenuti in quel passo ordinato per la dimensione del contatto stesso, ovvero per il numero di persone coinvolte. E' necessario infine modificare la segnatura del metodo, sostituendo tutti i riferimenti **`Object`** con il tipo piu' opportuno.
- Per poter calcare correttamente le statistiche può essere necessario modificare anche le classi **`Contatto`** e/o **`Persona`**, soprattutto in virtu' del fatto che oggetti di entrambe le classi siano inseriti all'interno di insiemi.
- completare il corrispondente test-case **`testProduciStatistica()`** all'interno della classe **`StatisticheTest`**

DOMANDA 4 (25%)
Scrivere una o piu' classi di test (posizionandole corrispondentemente alla classe sotto test, e denominandole di conseguenza: ovvero all'interno della directory **`src/test/`** e chiamandola, ad esempio, **`prop.pers.`*`NomeClasse`*`Test`** se ***`NomeClasse`*** è il nome della classe testata), con test-case *minimali* per verificare il corretto funzionamento dei metodi che si occupano della scelta della prossima destinazione di tutte le tipologie di persone. In particolare si richiede di scrivere prioritariamente test in ambienti minimali (piccoli) e con un numero minimale di persone coinvolte ma sufficiente a verificare il comportamento atteso.

E' possibile, ma solo se ritenuto conveniente e senza compromettere il resto del progetto, modificare anche il codice della classi oggetto della precedente domanda 2 per favorirne la *testabilita*`.