Una software house deve ristrutturare una applicazione che simula le dinamiche di un parco montano. Nel parco (per ora) ci sono due tipologie di animali: Roditori e Rapaci, modellati dalle corrispondenti classi. Il parco e' modellato dall'omonima classe: questa classe gestisce due mappe che associano ad una posizione la presenza di un animale (roditore o rapace).

Roditori e rapaci hanno diverse proprieta' in comune (peso, cibo, probabilita' di riproduzione, anni, posizione), ma differiscono per il comportameto. Entrambi si riproducono e muoiono (per mancanza di cibo o per vecchiaia), tuttavia i rapaci cacciano, mentre i roditori si spostano nel parco (se non lo fanno non si nutrono). La caccia dei rapaci consiste nel cercare nelle proprie adiacenze un animale piu' piccolo (con minor peso); se lo trovano, lo mangiano: ne prendono il posto e aumentano il proprio cibo. NB: come si puo' osservare dal codice, per semplicita' i pesi dei rapaci e dei roditori sono costanti (fissate al momento della creazione degli oggetti) e i rapaci sono piu' pesanti dei roditori.

Ci si e' resi conto che il software non e' ben progettato.

- DOMANDA 1 (5%): nelle mappe `posizione2rapace` e `posizione2roditore` viene sollevato un errore. Perche'? (scriverlo in un commento di max due righe, all'inizio della classe `Simulatore`). Risolvere il problema.
- DOMANDA 2 (20%): nella classe `Parco` scrivere il metodo `Map<Integer, Integer> cibo2numeroRoditori()`: restituisce una mappa che ha per chiave valore di cibo, e per valore il numero di roditori con tale valore di cibo
- DOMANDA 3 (20%): scrivere una classe di test con almeno tre metodi di test per il metodo `List<Roditore> roditoriOrdinatiPerEtaCibo()` della classe `Parco` (vedi DOMANDA 4).
- DOMANDA 4 (15%): nella classe `Parco` scrivere il metodo `List<Roditore> roditoriOrdinatiPerEtaCibo()`: deve restituire una lista dei roditori presenti nel parco ordinata in ordine crescente di eta, a parita' di eta, di cibo. E' possibile introdurre nuove classi.
- DOMANDA 5 (40%): ci sono molte linee di codice in comune tra `Roditore` e `Rapace`, cosi' come nella classe `Parco` ci sono molti metodi che differiscono solo per la tipologia di animale. Un progettista esperto ci fa notare che e' opportuno introdurre una classe astratta `Animale`, che deve essere estesa da `Rapace` e `Roditore`, e che nella classe `Parco` si deve usare una sola mappa, che associa una posizione ad un animale. Implementare le modifiche suggerite dal progettista. Anche i metodi delle domande 2 e 4 dovranno essere cambiati per poter lavorare su oggetti `Animale` anzicche' su oggetti `Roditore`; in pratica si avranno i metodi `List<Roditore> roditoriOrdinatiPerEtaCibo()`, e `Map<Integer, Integer> cibo2numeroRoditori()` (non e' necessario modificare la classe di test).
  NB: (facoltativo, molto semplice) per far funzionare il simulatore con la nuova implementazione e' necessario modificare anche il medoto `run()` della classe `Simulatore` (facile) e usare il codice commentato nella classe `Visualizzatore` (banale).