Seconda Prova di esonero – I turno A – 11 gennaio 2024

Creare un workspace avente nome *Esonero2Cognome*, e.g. *Esonero2Rossi*. In tale workspace si crei un progetto Java avente lo stesso nome del workspace.
Nel progetto creato si realizzino due package: *esonero2.catalogo* ed *esonero2.test*.
Nel package *esonero2.catalogo* si realizzino due classi che fungano da Catalogo di Abitazioni. In particolare:

- Catalogo di abitazioni (di qualunque tipo esse siano);
- Catalogo di ville (le ville sono abitazioni di un particolare tipo).

Classe Catalogo (valida per ogni classe Catalogo sopra elencata)
Rappresenta un contenitore di abitazioni, può contenere istanze di abitazioni ed è così definita:

- Attributi:
  - abitazioni: contenitore per memorizzare vari tipi di abitazioni. **Non sono ammesse abitazioni che abbiano lo stesso codice**.
- Operazioni:
  1. boolean catalogaAbitazione(Abitazione abitazione): *true* se aggiunge un'abitazione al contenitore, *false* altrimenti. Non c'è un limite al numero di abitazioni.
  2. boolean rimuoviAbitazioneByCodice(String codice): true se l'abitazione, identificata mediante codice, è stata rimossa, false altrimenti.
  3. Abitazioni getElenco(descrizione, filtro, operazione) dove:
     - descrizione: breve descrizione dell'operazione eseguita; tale descrizione sarà visualizzata sul terminale.
     - filtro: il filtro da applicare alla lista di elementi;
     - operazione: l'operazione da eseguire su ciascun elemento che soddisfa il filtro.
     - Abitazioni: Collezione di abitazioni contenente il risultato ottenuto a seguito dell'applicazione del *filtro* e dell'esecuzione dell'*operazione*.

     Tramite l'applicazione di una specifica coppia (filtro, operazione) si dovranno ottenere:
     3.1. lista di abitazioni il cui costo è superiore al valore definito in input ed ordinata in ordine ascendente per costo dell'abitazione (a partire dal meno costoso);
     3.2. lista di abitazioni il cui *luogo* cui sia uguale al valore definito in input all'applicazione;
  4. Abitazione getAbitazione(descrizione, operazione) dove:
     - descrizione: breve descrizione dell'operazione eseguita; tale descrizione sarà visualizzata sul terminale.
     - operazione: l'operazione da eseguire.
     - Abitazione: Abitazione che soddisfa il risultato ottenuto a seguito dell'esecuzione dell'*operazione*.

     Tramite l'applicazione di una specifica operazione si dovrà ottenere:
     4.1. Abitazione il cui costo è il più grande rispetto a tutte le altre.

Requisiti:

1. Per la realizzazione delle operazioni 3.1, 3.2 e 4.1 è obbligatorio l'uso delle API Streams e Lambdas.
2. Le operazioni di catalogazione e rimozione di un'abitazione saranno molto frequenti.

Qualunque tipo di abitazione sarà caratterizzata da:

- Attributi: codice (int), luogo (String), costo (double)
- Operazioni:
  - Abitazione(int codice, String luogo, double costo): costruttore per inizializzare gli attributi.
  - String getCodice(): restituisce il codice dell'abitazione.
  - String toString(): rappresentazione testuale di un'abitazione.
  - double calcolaCosto(): calcola e restituisce il costo dell'abitazione.

Le ville si caratterizzano per avere in più rispetto ad una generica abitazione:

- Attributi: metriGiardino (double)
- Metodi:
  - Villa(int codice, String luogo, double costo): costruttore per inizializzare gli attributi.
  - double calcolaCosto(): il costo dell'abitazione è aumentato del 150% se il valore di *metriGiardino* è superiore a 50,0.
  - setMetriGiardino(metriGiardino double): definisce i metri quadri del giardino.

Requisiti informativi delle abitazioni

- Il *costo* non può mai essere inferiore a 5.000, quando ciò accade un'istanza di abitazione non può essere creata.
- Il valore predefinito di *metriGiardino* è 5,0. Se per *metriGiardino* viene specificato un valore inferiore a 5,0 allora sarà utilizzato il valore predefinito.
- L'attributo *codice,* obbligatorio per ogni abitazione*,* identifica univocamente un'abitazione.

Esecuzione del TEST

Nel package *esonero2.test* si realizzi una classe di Test, denominata *CatalogoTest*, che esegua le seguenti **ATTIVITÀ**:

1. Acquisizione, mediante inserimento da tastiera di tutti le abitazioni i cui valori sono indicati nella sezione **INPUT** della presente traccia.
2. Aggiunta delle abitazioni create al punto 1. in un'istanza della classe Catalogo denominata catalogo.

3. Esecuzione delle operazioni 3.1, 3.2 e 4.1. Per ogni esecuzione si producano le stampe dei risultati ottenuti dalle operazioni. Per le operazioni 3.1 e 3.2 i valori del costo e quello del luogo devono essere acquisiti da tastiera.
4. Rimozione dell'oggetto denominato *villa1* dal catalogo con stampa a video dell'esito. Aggiunta dell'oggetto denominato *villa1* con stampa a video dell'esito.

INPUT

1. **abitazione1** (nome della variabile - abitazione generica):
   - Codice: 1; Luogo: Milano; Costo: 50000
2. **abitazione2** (nome della variabile - abitazione generica):
   - Codice: 2; Luogo: Roma; Costo: 70000
3. **villa1** (nome della variabile – abitazione di tipo villa):
   - Codice: 3; Luogo: Firenze; Costo: 100000
4. **villa2** (nome della variabile - abitazione di tipo villa):
   - **Codice: 2**; Luogo: Campobasso; Costo: 10000 euro;
5. **abitazione3** (nome della variabile - abitazione generica):
   - Codice: 4; Luogo: Isernia; **Costo: 4999**

VALUTAZIONE

Premessa

- Se il programma presenta errori di compilazione il punteggio finale sarà 0 a prescindere da quanto sia stato implementato.
- La valutazione di un'attività che solleva eccezione sarà 0.
- Se un'attività, pur non sollevando eccezione, non è soddisfatta completamente sarà valutata 0.

Punteggi

- Le attività 1 e 2 saranno valutate al massimo 1 punto ciascuna.
- L'attività 3 sarà valutata al massimo 3 punti. Ogni singola operazione testata da tale attività sarà valutata al massimo 1 punto.
- L'attività 4 sarà valutata al massimo 1 punto.