Corso di Laurea in Ingegneria Informatica
Prova scritta di *Fondamenti di informatica II*
29 gennaio 2024
Tempo a disposizione: 1 ora e 10 minuti

Esercizio 1

Si implementi in Java una classe *Sistema* che fornisca metodi per l'analisi dei dati riguardanti viaggi in autobus e conducenti. Si supponga che siano definite le classi *Viaggio* e *Conducente*, che forniscono i seguenti metodi:

Classe *Viaggio*:

- *public String getCodice()*, che restituisce il codice che identifica il viaggio.
- *public String getDestinazione()*, che restituisce la destinazione del viaggio.
- *public int getDurata(),* che restituisce la durata (in ore) del viaggio.
- *public int getNumPasseggeri(),* che restituisce il numero di passeggeri che partecipano al viaggio.
- *public boolean equals(Object o)*.
- *public String toString().*

Classe *Conducente*:

- *public String getNome()*, che restituisce il nome che identifica il conducente.
- *public LinkedList<String> getViaggi()*, che restituisce la lista dei codici dei viaggi gestiti dal conducente.
- *public boolean equals(Object o).*
- *public String toString().*

La classe *Sistema* contiene le liste *listaViaggi* dei viaggi e *listaConducenti* dei conducenti. Oltre ad eventuali metodi che si ritengano necessari, si includano almeno i seguenti metodi nella classe:

- *public boolean verifica(int oreMin, int passeggeriMax),* che restituisce *true* se e solo se sono soddisfatte entrambe le seguenti condizioni:
  - nessun conducente ha un numero complessivo di ore inferiore a *caricoMin* (il numero complessivo di ore di un conducente è la somma delle durate dei viaggi gestiti dal conducente stesso);
  - nessun viaggio ha un numero di passeggeri maggiore di *passeggeriMax*.
- *public LinkedList<String> destinazioniRichieste(int passeggeriMin).* Il metodo restituisce la lista delle destinazioni richieste – una destinazione *d* è richiesta se soddisfa le seguenti condizioni:
  - almeno 2 viaggi hanno destinazione *d*;
  - il numero totale di passeggeri che partecipano a viaggi aventi destinazione *d* è maggiore di *passeggeriMin*.
- *public LinkedList<String> conducentiDiversi(String conducente).* Il metodo restituisce la lista dei nomi distinti dei conducenti i cui viaggi non hanno destinazioni in comune con i viaggi gestiti dal conducente con nome *conducente*.

Esempio. Si assuma che i dati a disposizione siano i seguenti:

- Viaggi:
  - {"V1", "Roma", 5, 70}
  - {"V2", "Roma", 6, 40}
  - {"V3", "Salerno", 4, 50}
  - {"V4", "Salerno", 4, 70}
  - {"V5", "Reggio Calabria", 3, 50}
  - {"V6", "Roma", 5, 40}
- Conducenti:
  - {"Rossi", ["V1", "V3", "V6"]}
  - {"Bianchi", ["V2"]}
  - {"Verdi", ["V4", "V5"]}

Allora:

- *verifica(6, 80)* restituisce *true* perché:
    - nessun conducente ha un numero complessivo di ore inferiore a 6 (Rossi ne ha 14, Bianchi 6 e Verdi 7);
    - nessun viaggio ha un numero di passeggeri maggiore di 80.
- *destinazioniRichieste(130)* restituisce la lista ["Roma"] perché:
    - Roma è la destinazione di 3 viaggi e il numero totale di passeggeri che partecipano a viaggi aventi destinazione Roma è 150;
    - Salerno è la destinazione di 2 viaggi ma il numero totale di passeggeri che partecipano a viaggi aventi destinazione Salerno è 120;
    - Reggio Calabria è la destinazione di un solo viaggio.
- *conducentiDiversi("Bianchi")* restituisce la lista [“Verdi”] perché:
    - Bianchi gestisce un solo viaggio, con destinazione Roma;
    - Roma non è tra le destinazioni dei viaggi gestiti da Verdi (Salerno e Reggio Calabria);
    - Roma è tra le destinazioni dei viaggi gestiti da Rossi (Roma e Salerno).

Esercizio 2

Si arricchisca la classe *ListaConcatenataInt* sviluppata durante il corso con un metodo *verificaSomme(int k)* che:

- considera le sottoliste separate da elementi aventi valore zero e non vuote;
- restituisce *true* se e solo se tutte tali sottoliste hanno somma pari a *k*.

Si assuma che la lista non sia vuota e contenga almeno uno zero. Il metodo *verificaSomme* dovrà essere ricorsivo o invocare un opportuno metodo ricorsivo sulla classe *NodoInt*.

Esempio. Se la lista contiene i valori [3, 2, 0, 2, 2, 1, 0, 0, 5] allora *verificaSomme(5)* restituisce *true*, perché le sottoliste [3, 2], [2, 2, 1] e [5] hanno somma pari a 5.