Corso di Laurea in Ingegneria Informatica
Prova scritta di *Fondamenti di informatica II*
17 giugno 2022
Tempo a disposizione: 1 ora e 10 minuti

Esercizio 1

Si implementi in Java una classe *Sistema* che fornisca metodi per l'analisi di dati relativi a prenotazioni di voli gestiti da una compagnia aerea. Si supponga che le classi *Volo* e *Prenotazione* siano già disponibili e forniscano i seguenti metodi:

Classe *Volo*:

- *public String getPartenza()*, che restituisce il nome dell'aeroporto di partenza del volo.
- *public String getArrivo()*, che restituisce il nome dell'aeroporto di arrivo del volo.
- *public int getPrezzoBusiness()*, che restituisce il prezzo di un posto in classe business.
- *public int getPrezzoEconomica()*, che restituisce il prezzo di un posto in classe economica.
- *public boolean equals(Object o).*
- *public String toString().*

Classe *Prenotazione*:

- *public LinkedList<String> getPercorso()*, che restituisce il percorso per cui è stata effettuata la prenotazione, cioè la lista dei nomi degli aeroporti attraversati. Ovviamente,
  - il primo aeroporto nel percorso è quello di partenza;
  - l'ultimo aeroporto nel percorso è quello di destinazione;
  - un aeroporto compare al più una volta nel percorso;
  - ogni coppia consecutiva di aeroporti nel percorso corrisponde ad un singolo volo.
- *public String getNomeCliente()*, che restituisce il nome del cliente che ha effettuato la prenotazione.
- *public String getClasse()*, che restituisce la classe del posto prenotato ("business" o "economica").
- *public boolean equals(Object o).*
- *public String toString().*

La classe *Sistema* contiene le liste dei voli e delle prenotazioni effettuate. Oltre ad eventuali metodi che si ritengano utili, si includano nella classe almeno i seguenti metodi:

- *public boolean verificaPrenotazioni().* Il metodo restituisce *true* se e solo se tutte le prenotazioni sono corrette. Una prenotazione è corretta se esiste un volo per ogni coppia consecutiva di aeroporti nel percorso prenotato.
- *public Volo voloMax().* Il metodo restituisce il volo per il quale è stata incassata la somma maggiore. Nel caso in cui più voli soddisfino la proprietà, il metodo restituisce uno qualsiasi di essi.
  L'incasso di un volo è dato dalla formula
  $$nPrenBusiness \times prezzoBusiness + nPrenEconomica \times prezzoEconomica$$
  dove *nPrenBusiness* e *nPrenEconomica* sono i numeri di prenotazioni in classe economica e business effettuate per il volo, mentre *prezzoBusiness* e *prezzoEconomica* sono i prezzi del volo in ciascuna delle due classi.
- public *LinkedList<String> destinazioneComune(String cliente).* Il metodo restituisce la lista dei nomi dei clienti che hanno almeno una destinazione in comune con il cliente *cliente*, cioè i clienti che hanno effettuato almeno una prenotazione avente come destinazione una delle destinazioni presenti nelle prenotazioni effettuate dal cliente *cliente*.

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

- Voli:
  - {Partenza "Roma", arrivo "Milano", prezzo business 300, prezzo economica 150}
  - {Partenza "Lamezia", arrivo "Roma", prezzo business 200, prezzo economica 120}
  - {Partenza "Lamezia", arrivo "Milano", prezzo business 240, prezzo economica 130}
  - {Partenza "Roma", arrivo "Londra", prezzo business 450, prezzo economica 250}
  - {Partenza "Milano", arrivo "Parigi", prezzo business 350, prezzo economica 200}

- Prenotazioni:
    - {Percorso ["Lamezia","Roma","Londra"], cliente "Rossi", classe "business"}
    - {Percorso ["Roma","Milano","Parigi"], cliente "Rossi", classe "business"}
    - {Percorso ["Milano","Parigi"], cliente "Bianchi", classe "economica"}
    - {Percorso ["Lamezia","Milano","Parigi"], cliente "Bianchi", classe "economica"}
    - {Percorso ["Lamezia","Roma"], cliente "Verdi", classe "economica"}

Allora:

- *verificaPrenotazioni()* restituisce *true* perché tutte le prenotazioni sono corrette.
- *voloMax()* restituisce il volo {Partenza "Milano", arrivo "Parigi", prezzo business 350, prezzo economica 200} perché per tale volo è stata incassata la somma maggiore (750).
- *destinazioneComune("Rossi")* restituisce la lista ["Bianchi"] perché le destinazioni del cliente "Rossi" sono "Londra" e "Parigi" e l'unica destinazione del cliente "Bianchi" è "Parigi", mentre l'unica destinazione del cliente "Verdi" è "Roma".

Esercizio 2

Si arricchisca la classe *ListaConcatenataInt* sviluppata durante il corso con un metodo *contaElementiSpeciali(int b)* che conta quanti elementi speciali sono presenti nella lista; un elemento è detto *speciale* quando ha valore maggiore della differenza tra *b* e la somma degli elementi precedenti all'elemento stesso. Ad esempio, se *b* = 10 e la lista contiene i valori [5,3,1,0,**2**,-2,**4**,**6**,-9], allora il metodo restituisce 3 perché:

- 2 > 10 – (5+3+1+0);
- 4 > 10 – (5+3+1+0+2-2);
- 6 > 10 – (5+3+1+0+2-2+4);
- nessun altro elemento soddisfa la condizione.

Il metodo *contaElementiSpeciali* dovrà essere ricorsivo o invocare un opportuno metodo ricorsivo sulla classe *NodoInt*.