FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 30 minuti

Nome ..... Cognome ..... Matricola .....

Esercizio 1 [C++] (15pt). Definire una classe templatica `Insieme<T>` che realizza il tipo di dato astratto insieme di elementi di tipo `T`. La classe deve definire un costruttore senza argomenti che crea un insieme vuoto. Il metodo `add` aggiunge un elemento di tipo `T` all'insieme. Se l'elemento è già presente, il metodo deve lanciare un'eccezione. Il metodo `contains` verifica se un elemento appartiene all'insieme. Il metodo `size` ritorna la cardinalità dell'insieme. Si sovraccarichi l'operatore `+` in modo tale che restituisca un nuovo insieme ottenuto dall'unione dell'insieme su cui è chiamato il metodo e l'insieme passato come parametro. Si sovraccarichi l'operatore `<<` in modo tale che stampi gli elementi dell'insieme su uno stream di output `fout` nel formato $\{e_1, e_1, \ldots, e_n\}$. Non è consentito utilizzare classi della STL. Se necessario, ridefinire gli opportuni metodi, costruttori e/o operatori. Specificare opportunamente eventuali metodi costanti.

Esercizio 2 [Java] (15pt).

- Si implementi una classe `GreenPass`, contenente il codice fiscale della persona per cui è stato generato e una data di scadenza. Tale classe deve sovrascrivere i metodi `toString` e `equals`. Due green pass sono uguali per il metodo `equals` se sono stati generati per la stessa persona e hanno la stessa data di scadenza.
- Si implementi una classe `GreenPassVaccino`, sottoclasse di `GreenPass`, che rappresenta un green pass ottenuto tramite vaccino. La classe deve contenere l'informazione sul tipo di vaccino effettuato. I possibili tipi di vaccino sono `VACCINO1, VACCINO2, VACCINO3`. Tale classe deve sovrascrivere i metodi `toString` e `equals`. Due green pass ottenuti tramite vaccino sono uguali per il metodo `equals` se sono stati generati per la stessa persona, hanno la stessa data di scadenza e riguardano lo stesso tipo di vaccino.
- Si implementi poi una classe `GreenPassTampone`, sottoclasse di `GreenPass` che rappresenta un green pass ottenuto tramite tampone. La classe deve contenere l'informazione sul tipo di tampone effettuato. I possibili tipi di tampone sono `TAMPONE1, TAMPONE2, TAMPONE3`. Tale classe deve sovrascrivere i metodi `toString` e `equals`. Due green pass ottenuti tramite tampone sono uguali per il metodo `equals` se sono stati generati per la stessa persona, hanno la stessa data di scadenza e riguardano lo stesso tipo di tampone.
- Si implementi una classe `PersonaConGreenPass`, che modella una persona che ha zero o più green pass. Una persona può avere più di un green pass ottenuto tramite tampone ma un unico green pass ottenuto tramite vaccino. La classe deve contenere le informazioni riguardanti il codice fiscale della persona e i green pass posseduti da quella persona. La classe deve definire un unico costruttore che prende come parametro un codice fiscale. Il metodo `addGreenPass` prende come parametro un green pass e lo aggiunge all'insieme di green pass posseduti dalla persona su cui è invocato il metodo. Se il codice fiscale associato al green pass è diverso dal codice fiscale della persona su cui è invocato il metodo, il metodo deve lanciare un'eccezione **di tipo controllato** `GreenPassException` (da implementare). Inoltre, se il green pass passato come parametro è un'istanza di `GreenPassVaccino` e la persona ha già un green pass ottenuto tramite vaccino, il metodo deve lanciare la stessa eccezione `GreenPassException` implementata precedentemente. La classe `PersonaConGreenPass` deve sovrascrivere i metodi `toString` e `equals`. Due persone sono uguali per il metodo `equals` se hanno lo stesso codice fiscale e gli stessi green pass.

N.B. Massimizzare incapsulamento e information hiding. Non è richiesta l'implementazione del metodo `hashCode` per le classi richieste. Per ciascuna classe, è possibile supporre di avere a disposizione un implementazione del metodo `hashCode` coerente col metodo `equals` che implementerete.

(+2pt) La classe `PersonaConGreenPass` deve implementare l'interfaccia `Comparable<T>`. Il metodo corrispondente da implementare utilizza il numero di green pass posseduti da una persona per il confronto.

Per le date, si utilizzi la seguente classe già implementata `Data`.

```
public class Data {
  private final int giorno;
  private final int mese;
  private final int anno;

  public Data(int giorno, int mese, int anno) {
    if (valida(giorno, mese, anno)) {
      this.giorno = giorno;
```

```
      this.mese = mese;
      this.anno = anno;
    } else
    throw new RuntimeException();
  }

  private boolean valida(int giorno, int mese, int anno) {
    // controllo sulla validita' di giorno, mese e anno
  }

  @Override
  public String toString() {
    return giorno + "/" + mese + "/" + anno;
  }

  @Override
  public int hashCode() {
    // implementazione di hashCode coerente con il metodo equals
  }

  @Override
  public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null)
      return false;
    if (getClass() != obj.getClass())
      return false;
    Data other = (Data) obj;
    return anno == other.anno && giorno == other.giorno && mese == other.mese;
  }
}
```