FONDAMENTI DI PROGRAMMAZIONE B

Tempo a disposizione: 2 ore 30 minuti

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

Esercizio 1 [C++] (15pt). Definire una classe templatica `Queue<T>` che realizza il tipo di dato astratto coda di elementi di tipo `T` che permette di inserire un nuovo elemento in coda e di rimuovere un elemento dalla testa della coda.

La classe deve definire:

- un costruttore senza argomenti che crea una coda vuota
- un metodo `enqueue` che, preso come argomento `elem`, aggiunge `elem` in coda
- un metodo `dequeue` che rimuove l'elemento in testa alla coda e lo restituisce. Se la coda è vuota, il metodo deve lanciare un'eccezione.
- un metodo `isEmpty` che controlla se la coda è vuota

Si sovraccarichi l'operatore `<<` in modo tale che stampi gli elementi della coda su uno stream di output `fout` nel formato $[e_1, e_2, \ldots, e_n]$. Non è consentito utilizzare classi della STL. Se necessario, ridefinire gli opportuni metodi, costruttori e/o operatori. Specificare opportunamente eventuali metodi e parametri costanti. Massimizzare incapsulamento e information hiding.

Esercizio 2 [Java] (15pt).

Nel contesto di un sistema di pagamento, si sviluppino le seguenti classi ed interfacce.

- Si implementi un'interfaccia `MetodoPagamento` che contiene un metodo `verificaSaldo` che ritorna un valore di tipo `double` e due metodi chiamati `incrementa` e `decrementa` che prendono come paramentro un valore tipo `double`. Si implementino le classi `CartaDiCredito` e `ContoCorrente`. Entrambe le classi devono implementare l'interfaccia `MetodoPagamento`.
- La classe `CartaDiCredito` è caratterizzata dal numero della carta di credito e dal suo saldo. La classe deve mettere a disposizione un costruttore che prende come parametri il numero della carta e il suo saldo iniziale. Si ridefinisca il metodo `equals`. Due carte di credito sono uguali per il metodo `equals` se hanno lo stesso numero di carta e lo stesso saldo.
- La classe `ContoCorrente` è caratterizzata da un codice IBAN e dal suo saldo. La classe deve mettere a disposizione un costruttore che prende come parametri il codice IBAN e il saldo iniziale. Due conti corrente sono uguali per il metodo `equals` se hanno lo stesso codice IBAN e lo stesso saldo.
- I metodi `decrementa` ed `incrementa` devono decrementare ed incrementare il saldo della carta di credito o del conto corrente su cui è invocato il metodo. Se il saldo non è sufficiente, il metodo `decrementa` deve lanciare un'eccezione di tipo non controllato `SaldoNonSufficienteException`, da implementare.
- Si implementi la classe `Persona` caratterizzata da un nome, un cognome, e un insieme di metodi di pagamento posseduti dalla persona. La classe deve mettere a disposizione un costruttore che prende come parametri il nome e il cognome della persona e li inizializza. Inizialmente, la persona non ha alcun metodo di pagamento. La classe deve mettere a disposizione un metodo `aggiungiMetodoPagamento` che aggiunge un metodo di pagamento alla persona su cui è invocato il metodo. La classe deve mettere a disposizione un metodo `paga` che prende come parametro un metodo di pagamento `m` e un importo `double`: il metodo deve controllare se il metodo di pagamento `m` è posseduto dalla persona su cui è invocato il metodo. Se sì, il metodo deve decrementare il saldo del metodo di pagamento `m` di un importo pari a `importo`. Altrimenti, il metodo ritorna.

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 una implementazione del metodo `hashCode` coerente col metodo `equals` che implementerete.

(+2pt) La classe `ContoCorrente` deve implementare l'interfaccia `Comparable`. Il metodo corrispondente da implementare utilizza il saldo per il confronto.