Metodi Avanzati di Programmazione, Corso A, 3 Aprile 2024

1)Fornire le specifiche algebriche minimali e complete (semantiche e di restrizione), **in forma di equazioni**, per il tipo astratto **Vettore** di cui si forniscono le seguenti specifiche sintattiche:

Tipi: Vettore, Elemento Intero, Booleano

crea(Intero)→ Vettore //crea un vettore inizialmente vuoto di dimensione pari all'intero passato come parametro

assegna(Vettore, Intero, Elemento)→Vettore // assegna l'elemento (terzo argomento) alla cella del vettore (primo argomento) identificata dalla posizione passata come secondo argomento

modifica(Vettore, Intero, Elemento)→Vettore // rimpiazza l'elemento (terzo argomento) nella cella del vettore (primo argomento) identificata dalla posizione passata come secondo argomento. Solleva errore se la posizione non è valida o la cella è valida, ma è vuota

prefisso(Vettore, Intero) → Vettore // restituisce il prefisso composto dai primi elementi (in numero pari all'intero passato come secondo argomento) del vettore. Solleva errore se l'intero passato è negativo o più grande della dimensione del vettore

conta(Vettore, Elemento) → Intero // conta le occorrenze di elementi memorizzati nel vettore (primo argomento) che sono più grandi dell'elemento passato come secondo argomento

uguale(Vettore, Vettore, Elemento) → Booleano // restituisce vero se i due Vettori passati come primo e secondo argomento contengono lo stesso numero di occorrenze di elementi più grandi dell'elemento passato come terzo argomento, falso altrimenti.

(11 punti)

2) Descrivere in maniera **esaustiva** il modello concettuale **ereditarietà** nel paradigma **Object Oriented** fornendo e commentando per **ciascuna forma** un ragionevole **esempio UML**. Indicare, per ogni forma, se è supportata in **Java**. Per le forme di ereditarietà eventualmente non supportate da Java descrivere come è possibile **modificare l'esempio UML** affinché sia implementabile in Java.

(11 punti)

3) Descrivere l'uso di **Comparable** in Java. Mostrare come **Comparable** può essere usato per cercare l'elemento **minimo** di un **array** di elementi (array non ArrayList).

Scrivere in Java la classe **Esame** (campi: codice, voto, numeroCFU) e la classe **Libretto** (campi: c **array** di **Esame**). Scrivere il metodo della classe **Libretto** che realizza la specifica sintattica min(Libretto)→Esame (usando **Comparable**). Scrivere un main che popoli una istanza di Libretto con almeno 3 esami e usare il metodo min per stampare a video l'esame in cui si è raggiunto il voto più basso e l'esame con numero di CFU associati più bassi. **Commentare il codice scritto**

*Gli iscritti che avrebbero dovuto frequentare l'esame di MAP negli AA **maggiore uguale** di 2017-2018 (perché allora iscritti al secondo anno) DEVONO rispondere ANCHE alla domanda:*

Descrivere l'operazione di riduzione **reduce** nella estensione funzionale di Java. Scrivere il metodo di **Libretto** che usi una pipeline con reduce per contare il numero di esami con voto maggiore di 28. **Commentare il codice scritto**

Gli iscritti che avrebbero dovuto frequentare l'esame di MAP negli AA precedenti al 2017-2018 DEVONO rispondere ANCHE alla domanda precedente o in ALTERNATIVA alla seguente domanda:

Descrivere in maniera esaustiva classi astratte e interfacce nel paradigma OO con esempi in UML e mostrare analogie e differenze tra questi due modelli concettuali. **(11 punti)**