PROGRAMMAZIONE II - a.a. 2019-20

Esercitazione 24 Ottobre 2019

Esercizio

Si consideri il tipo di dato *buffer* che è un contenitore di dati generici. Un buffer è una sequenza lineare e finita di elementi di uno specifico tipo generico. Le proprietà strutturali di un buffer sono la sua *capacità*, il *limite* e la *posizione*.

- La capacità di un buffer è il numero di elementi che contiene. La capacità di un buffer non è mai negativa e non cambia mai.
- Il limite di un buffer è l'indice del primo elemento che non può essere letto o scritto. Il limite di un buffer non è mai negativo e non è mai superiore alla sua capacità.
- La posizione di un buffer è l'indice dell'elemento successivo da leggere o scrivere. La posizione di un buffer non è mai negativa e non è mai superiore al suo limite.

Supponiamo che la definizione del tipo di dato `Buffer<T>` contenga tra gli altri i metodi

```
interface Buffer<T> {

   /* rende il buffer disponibile per una nuova sequenza di operazioni di lettura o scrittura */
   void clear();

   /* rende il buffer disponibile per rileggere-riscrivere i dati che contiene */
   void rewind();

   /* inserisce l'intero contenuto dell'array di elementi generici nel buffer*/
   void put(T[] src);

    /* trasferisce gli elementi del buffer nell'array generico*/
   T[]  get();
   }
```

1. Assumendo di adottare una strategia di programmazione difensiva, si completi il progetto del tipo di dato astratto `Buffer<T>` , definendo le clausole REQUIRES, MODIFIES, e EFFECTS di ogni metodo, indicando le eccezioni eventualmente lanciate e se sono checked o unchecked.
2. Si consideri la seguente struttura di implementazione per la classe `MyBuffer<T>`

   ```
   private Vector<T> elems;
   private int capacity;
   private int limit;
   private int position;
   ```

   Si definisca l'invariante di rappresentazione per l'implementazione di `MyBuffer<T>`.
3. Si fornisca l'implementazione del costruttore e dei metodi `rewind`, `clear` e `put` e si dimostri che le implementazioni proposte preservano l'invariante di rappresentazione.
4. Si consideri una classe `AlwaysReadWrite<T>` che estende `myBuffer<T>` permettendo di leggere-scrivere sempre gli elementi del buffer. Giustificando la risposta, si dica come deve essere modificata la struttura di implementazione del buffer.