SINF1121 – Algorithmique et structures de données

Mission 1

Questions complémentaires

Cette liste de questions complémentaires est proposée aux étudiants pour la séance de bilan de la Mission 1. Chaque groupe d'étudiants est invité à résoudre collectivement certaines de ces questions et à en discuter avec son tuteur.

On s'intéresse à la classe `NodeQueue` qui constitue une implémentation du type abstrait `File` (Queue, en anglais). Cette implémentation utilise une liste ***simplement*** chaînée ***circulaire***, c'est-à-dire pour laquelle la dernière position de la liste fait référence, comme position suivante, à la première position de la liste. L'ajout d'un nouvel élément dans la file (méthode `enqueue`) se fait en fin de liste et le retrait (méthode `dequeue`) se fait en début de liste. La classe `NodeQueue` ***ne*** fait ***pas*** usage de position sentinelle et dispose d'***une seule*** référence `marker` à une position particulière (à vous de déterminer laquelle !) au lieu des deux références usuelles `head` et `tail`. ***Toutes les opérations*** sur la file ***doivent*** se faire en $\mathcal{O}(1)$.

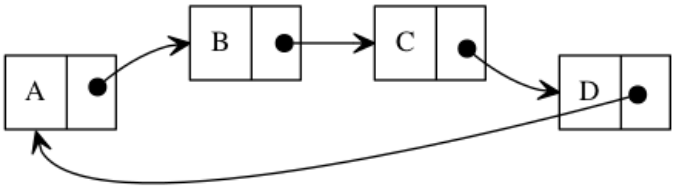

FIGURE 1 – Liste simplement chaînée circulaire.

Par convention, on considère que la position la plus à gauche correspond au début de la liste et la position la plus à droite à la fin de la liste.

Q 1 La classe `NodeQueue` implémente l'interface `Queue`. On vous demande de compléter les méthodes de cette implémentation ***sans*** ajouter ou supprimer de variables d'instance dans cette classe. Vous veillerez à ***commenter succintement et précisément*** votre code.

```
public class Node<E> {
 private E element;
 private Node<E> next;
 public Node(E e, Node<E> n) {
   element = e;
   next = n;
 }
 public E getElement() {return element;}
 public Node<E> getNext() {return next;}
 public void setNext(Node<E> n) {next = n;}
}

public class NodeQueue<E> implements Queue<E> {
  // Variables d'instance
  private Node<E> marker;
  private int size;
```

Q 1.1

```
  // Constructeur
  public NodeQueue() {
  ..........................................................................
  ..........................................................................
  ..........................................................................
  ..........................................................................
  }
```

```
  public int size() { return size; }
  public boolean isEmpty() { return (size==0); }
```

Q 1.2

```
  public E front() throws QueueEmptyException {
  ..........................................................................
  ..........................................................................
  ..........................................................................
  ..........................................................................
  ..........................................................................
  ..........................................................................
  }
```

Q 1.3

```
public void enqueue (E element) {
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
}
```

Q 1.4

```
public E dequeue() throws QueueEmptyException {
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
}
```

```
} // Fin de la classe NodeQueue
```

Q 2 L'examen de LSINF1121 comporte une partie programmation des structures de données et algorithmes sur ordinateur. Pour réussir cette partie, il est indispensable de programmer régulièrement. Les erreurs triviales de Java seront perçues par le correcteur comme un manque de participation de la part de l'étudiant aux missions de programmation. Nous vous proposons d'étudier ici quelques erreurs ou problèmes rencontrées fréquemment.

Q 2.1

Nous envisageons d'implémenter une Stack avec un tableau. Il faudra donc redimensionner le tableau si la taille de la stack atteint la taille maximale. Est-ce que Java dispose d'une méthode efficace pour redimensionner/recopier un tableau ? Si oui donnez un exemple de code Java pour réaliser cette opération.

Q 2.2

Est-ce que Java propose une classe pour les *Stack, Vector, List* ? Si oui dans quel package ? A votre avis, est-ce intéressant de bien connaître ce package pour l'examen ? *List* est-elle une interface ou une classe ? Comment créer un object de type *List*.

Q 2.3

Quelle est la différence entre un Iterable et un Iterator ?

Q 2.4

Quelle est l'erreur dans le code suivant où l'étudiant cherche à créer un tableau de 5 listes et ensuite insérer l'entier 4 dans la 3e liste ? Corrigez le code.

```
public List<Integer>[] myList = new List<Integer>[5];
myList[2].add(4);
```

Q 2.5

Quelle est l'erreur dans le code suivant où l'étudiant cherche à créer un objet Iterable ? Corrigez le code.

```
public Iterable<Integer> myIterable = new Iterable<Integer>();
```

Q 2.6

Quelle est l'erreur dans le code suivant où l'étudiant cherche à définir un constructeur ?

```
public class ADT {
  private int n = 4;
  private ADT myAdt;
  public ADT(int n) {
    n = n;
    myAdt = new ADT(4);
  }
}
```