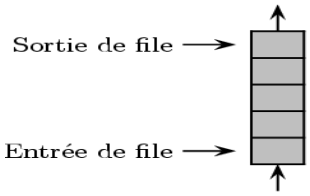
Examen Final de programmation Orientée Objet – 2 Heures

Documents autorisés

Barème indicatif, non définitif



Soit une **classe File** utilisée pour une fonctionnalité de file d'attente : le premier élément entré est le premier sorti (FIFO : first-in first-out). Le nombre d'éléments en file peut varier au cours du temps suivant les entrées et les sorties. La taille max de la file est fixe (nombre d'éléments au maximum en file).

Soit la classe File :

```
public class File {

	// Attributs :
	private Object[] tab; // tab est utilisé pour stocker les éléments de la file
	private int nbElements; // nombre d'éléments dans la file
	public static final int TAILLE_INIT = 50; // taille d'allocation par
                                                  défaut du tableau

	// Constructeurs :
	public File() { /* TODO : Q3*/ }
	public File(int _tailleInit) { /* TODO : Q3*/ }

	// Fonctions :
	// retourne le nombre d'éléments de la file
	public int getTaille() { /* TODO : Q4*/ }
	// Enfile l'objet et retourne vrai si la file n'est pas pleine, sinon renvoie faux
	public boolean enfiler(Object _o) { /* TODO : Q4*/ }
	// Défile un objet si la file est non vide, renvoie null sinon
	public Object defiler() { /* TODO : Q4*/ }
	// Vide la file
	public void vider() { /* TODO : Q4*/ }
}
```

I / La classe FILE (6 Pts)

Q1 (1 Pt) / Est-ce qu'un objet de type File peut enfiler les objets d'une classe Polygone ? Pourquoi ?

Q2 (1 Pt) / Quelle est la différence entre **`tab.length`** *et* **`nbElements`** *? Décrire textuellement la façon dont les éléments sont enfilés et défilés selon votre proposition (vous pouvez ajouter des attributs si besoin).*

Q3 (1 Pt) / Proposer une implémentation pour les deux constructeurs suivant :
Par défaut, le tableau **`tab`** est initialisé à la taille **`TAILLE_INIT`**. Lorsque le paramètre **`_tailleInit`** est précisé, le tableau est initialisé à **`_tailleInit`**. *Essayer de factoriser si possible (utiliser :* **`this(…)`***).*

Q4 (3 Pts) / Proposer une implémentation des fonctionnalités restantes.

II / Utilisation de la classe File (7 Pts)

Soit la fonction utilitaire suivante :

```
Class Math
		public static int rnd() ; // retourne un nombre aléatoire entier
```

Q5 (1.5 Pt) / Ajouter à la classe **`Math`** *une fonction utilitaire permettant d'initialiser aléatoirement (et dynamiquement) un tableau d'entiers de taille* **`n`** *(donner la signature* ***et*** *le code de la fonction).*

Q6 (2 Pts) / Dans le programme principal de notre application (le « main »), implémenter le traitement suivant :

Soit deux tableaux d'entiers **`t1`** et **`t2`** de taille 10 initialisés aléatoirement et un tableau d'entier **`t3`** de taille 20. *Enfiler consécutivement les éléments de* **`t1`** *et* **`t2`**, *pour ensuite défiler les éléments dans* **`t3`**.

Q7 / On souhaite réaliser le même traitement en utilisant trois processus distincts (un pour **`t1`**, *un pour* **`t2`**, *un pour* **`t3`**).

- *(1 Pt) Faut-il synchroniser ? Pourquoi ? Si oui, comment ?*
- *(2.5 Pt) Écrire le traitement multi-thread [rq. : question assez longue, plutôt en fin d'examen ...]*

III / Évolutions des fonctionnalités de la File (7 pts)

Indication : Les objets de type **`FileTypeUnique`** et **`FilePrioritaire`** sont aussi du type **`File`** …

Spécialisation 1 : Filtre sur les éléments autorisés

On souhaite créer une classe **`FileTypeUnique`** qui autorise uniquement l'ajout d'éléments de même type. Ainsi à chaque entrée la méthode **`valider`** va vérifier que le nouvel objet est issue de la même classe que le dernier objet ajouté (utiliser la fonction ci-dessous). Si l'objet est compatible, on l'enfile, sinon on retourne une exception de votre choix (prévoir le cas du premier objet ajouté).

```
Classe Object
      public Class getClass() {…} // retourne la classe de l'objet
```

Q8 (1.5 Pts) / Proposer l'implémentation de la fonction **`valider`** *ainsi que la redéfinition de la fonction* **`enfiler`** *pour la classe* **`FileTypeUnique`** *(signatures de fonction incluses).*

Spécialisation 2 : File avec priorités

On souhaite à présent créer une nouvelle classe **`FilePrioritaire`** qui, en plus d'autoriser uniquement l'ajout d'éléments de même type, réponde aux fonctionnalités suivantes :

- Tous les objets manipulés sont des **`Comparable`**
- Lorsque l'on enfile un objet, celui-ci peut « doubler » les autres éléments de la file d'attente, tant qu'il leur est supérieur

Q9 (1 Pt) / Question de cours : Que peut-on déduire de la phrase « Tous les objets manipulés sont des **`Comparable`** *» ?*

Q11 (2 Pts) / Sachant que **`File`**, **`FileTypeUnique`**, **`FilePrioritaire`** *ont des fonctionnalités proches, comment les mettre en relation dans le modèle objet ? Donner le diagramme de classe global pour la partie III (indice : modèle avec 4 classes intéressant). Documenter, justifier.*

Q10 (1.5 Pts)/ Proposer une implémentation des fonctions **`enfiler`** *et* **`valider`** *pour la classe* **`FilePrioritaire`** *(donner la signature* ***et*** *le code de la fonction).*

Q12 (1 Pt) / Soit la classe **`FilePrioritaireGenerique`** proposée ci-dessous qui utiliserait les types génériques :

```
Classe FilePrioritaireGenerique<T extends Comparable> {
            private T[] tab;          … }

Quel est l'intérêt d'utiliser des types génériques ?
```