Grands Moulins

19/03/2022 - 15h30-17h30

Initiation à la Programmation (IP2)

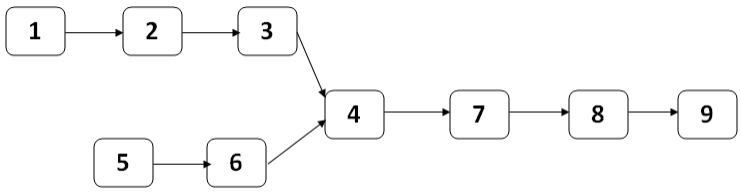
Aucun document autorisé. Dans vos réponses n'utilisez **aucune des librairies de java** qui sont en rapport avec les listes, vous devez écrire tout ce qui vous est utile.
Pensez à dire sans ambiguïté à quelle classe vos méthodes appartiennent. Ne comptez pas sur le correcteur pour le deviner : c'est à vous d'être clair.

Ce sujet est composé de 4 exercices. Ils sont indépendants. Le barème est indicatif, et vous donne une idée du temps à consacrer aux exercices. Si le barème évolue légèrement ce sera pour donner de l'importance aux exercices traités complètement et correctement : il vaut donc mieux s'appliquer à faire les choses bien plutôt que de se précipiter.

Exercice 1 (3 points) On rappelle la structure qui permet de définir les listes

```
public class ListInt {
  private Cellule first;
}
public class Cellule {
  private int val;
  private Cellule suivant;
}
```

Souvent on considère implicitement que les cellules sont séparées d'une liste à l'autre. Or rien n'interdit qu'il y ait un partage. Dans la figure suivante vous voyez qu'une liste commence par la cellule `1`, qu'une autre commence par la cellule `5`, et que plus loin des cellules se retrouvent mises en commun :



On veux détecter si deux listes ont une section commune non vide. Pour cela écrivez les deux méthodes suivantes, **en style itératif** :

- `public static boolean mergeTest(ListInt l1,ListInt l2)`
- `public static boolean mergeTest(Cellule l1, Cellule l2)`

Exercice 2 (6 points)

Dans cet exercice on travaille sur une structure de liste doublement chaînées.

```
public class ListInt {
 private Cellule first;
}
```

```
5 public class Cellule {
  private int val;
7 private Cellule suivant;
  private Cellule precedent;
9 }
```

Pour une raison ou pour une autre, on admet qu'une cellule peut avoir son attribut `precedent` corrompu. Il a alors une valeur fausse sur laquelle on ne peut pas compter. On suppose par contre que l'attribut `suivant` est lui toujours exact.

1. **(1.5 point)** Ecrivez une méthode `void resetPrecRec()` qui redéfini proprement les liaisons `precedent` des cellules d'une liste. On veut ici une solution **récursive**. [1]
2. **(1.5 point)** Ecrivez une méthode `void resetPrecIt()` qui résout le même problème **en itératif**
3. **(1.5 point)** Une fois que les cellules ont été réparées par l'une ou l'autre des méthodes précédente, on veut pouvoir vérifier que le travail a bien été fait. Pour cela, on vous demande d'écrire une méthode `void printBackwardRec()` qui affiche une liste, à l'envers, en se fiant aux liaisons `precedent`. On vous demande ici une solution **de style récursive**.
4. **(1.5 point)** même question pour `void printBackwardIt()` dans un **style itératif**

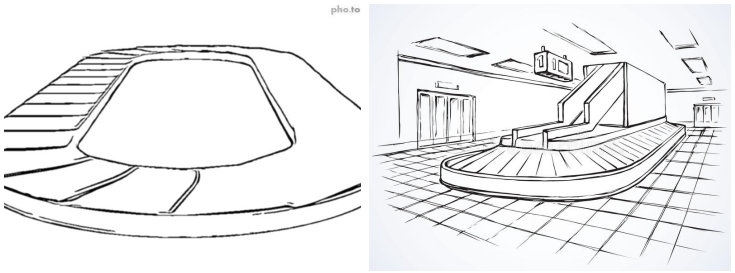
Exercice 3 (4 points) On s'intéresse à la valeur de la cellule "du milieu" dans une liste : le milieu de $l1 = \{5, 7, 30, 45, 50\}$ est 30. Comme cette notion n'est vraiment précise que pour les listes de longueur impaires, on la défini aussi pour les listes de longueur paire en disant que la valeur du "milieu" est la moyenne des deux candidats naturels. Ainsi pour $l2 = \{39, 29, 19, 9\}$, puisqu'il n'y a aucune raison de considérer que 29 serait un meilleur milieu que 19, on retournera $(29 + 19)/2$ c'est à dire 24. Par convention on décide aussi que le milieu de la liste vide est 0.

1. **(2 points)** Ecrivez les classes qui permettent de modéliser une liste doublement chaînée d'entiers, avec un observateur sur le début et la fin de la liste. Tous les attributs doivent être privés. Ecrivez également le constructeur de liste vide, une méthode `void addT(int x)` qui ajoute `x` en tête de liste, `void addQ(int x)` qui ajoute `x` en fin de liste, ainsi que les accesseurs et setter strictement nécessaire dans cette question (et seulement ceux là).
2. **(2 points)** Pour le calcul du milieu, on ne veut pas que vous trouviez votre propre solution, mais on souhaite savoir si vous êtes capable de traduire fidèlement la méthode décrite ci-dessous :
   - le principe consiste à avancer à partir des deux extrémités, en descendant d'un coté et en remontant de l'autre, et ceci tant que les références ne se sont pas croisées.
   - l'essentiel du travail doit être fait dans les cellules, **récursivement**, même si la méthode qui initie le calcul et gère les cas triviaux est appelée sur les listes.
   - une liste transmet à une cellule un argument qui lui sera utile pour savoir où en est le parcours censé remonter à partir de la fin.

   Ecrivez la méthode `int milieu()` de la classe liste, et celle qui lui correspond dans les cellules.

[1] Indication : vous pouvez transmettre un argument lors des appels au niveau des cellules

Exercice 4 [7 points] On s'intéresse à une chaîne d'emballage et d'expédition automatisée. Sa conception s'inspire des tapis roulants circulaires comme vous les connaissez lorsque vous récupérez vos bagages dans les aéroports (un carrousel) :



Le tapis en lui même est constitué de plateaux accrochés les uns aux autres. On peut au besoin l'allonger ou le raccourcir en ajoutant ou en retirant des plateaux.

On aura placé deux bras articulés à certains endroits : ils s'occupent de traiter ce qu'ils voient sur le plateau qui leur fait face. L'un est un *producteur*, et l'autre est un *consommateur*. C'est l'ensemble bras + tapis qui constitue notre chaîne de production. Elle fonctionne en recevant des commandes élémentaires qui peuvent être exclusivement : `prépare()`, `récupère()`, ou `tourne()`

- La commande `prépare()` fait travailler le *producteur* de la façon suivante :
  - s'il n'a rien sur le plateau devant lui, il y dépose un emballage,
  - s'il y avait déjà là un emballage, il y dépose un produit et referme l'ensemble.
  - sinon il insère un nouveau plateau afin d'en avoir un vide devant lui.
- la commande `récupère()` s'adresse au *consommateur* : si en face de lui il y a un produit emballé, il le retire et le renvoie, sinon il ne fait rien.
- la commande `tourne()` fait avancer le tapis d'un cran.

1. **[1 point]** Proposez un modèle qui permet de décrire le système que nous venons de présenter. (Dans cette question vous n'êtes noté que sur les noms des classes que vous choisissez et leurs attributs)
2. **[2 points]** La chaîne de production par défaut est constituée d'un tapis de 6 plateaux avec un producteur et un consommateur qui sont diamétralement opposés. Assurez cette construction.
3. **[1 point]** Ecrivez la méthode `tourne()`
4. **[1 point]** Ecrivez la méthode `récupère()`
5. **[2 point]** Ecrivez la méthode `prépare()`