Final LO43 A15

durée 2h, 14/01/16
Aucun document autorisé

Episode 1: La menace fantôme: les lambda expressions

Les lambda expressions ont fait leur apparition avec la version 8 du jdk. Expliquez en quoi elles consistent, comment elles s'écrivent et quels sont les cas les plus évidents pour lesquels on peut les utiliser.

Episode 2: L'attaque des clones.

Le clonage en général est la création d'un nouvel objet (une instance d'une classe) à partir d'une instance déjà existante. En java le clonage sert à faire la copie d'un objet dans un autre.

Question 1 : si x et y sont des objets de même type que fait la commande x=y ;

Question 2 : pour contourner ce problème on définit une méthode particulière pour pouvoir dupliquer l'objet appelant dans un autre objet. Quels sont les prérequis sur la classe qui veut permettre le clonage ?

Question 3 : Qu'affiche le code suivant :

```
class Cellule implements Cloneable {
      int i = 0 ;
      int [] t = {1,2} ;
      public Object clone(){
            try {
                  return super.clone() ;
            }
            catch (CloneNotSupportedException e) {
                  throw new InternalError();
            }
      }

      public void afficher() {
            System.out.println(i+" " + t[0]+" "+t[1]);
      }
}

class TestClone {
      public static void main(String args[]) {
            Cellule x= new Cellule() ;
            x.afficher() ;
            Cellule y= (Cellule) x.clone() ;
            y.afficher() ;
      }
}
```

Question 4 : On ajoute à la classe Cellule la méthode suivante :

```
public void changeMe() {
      i=10 ;
      t[0]=11 ;
      t[1]=12 ;
}
```

Quel est le résultat du code suivant ? Que se passe-t-il ?

```
x.changeMe() ;
x.afficher() ;
y.afficher() ;
```

Question 5 : Proposer des modifications pour éviter ce problème.

Episode 3: La revanche des Sith : l'ordre 66

On suppose que les valeurs de retour des méthodes qui sont négatives correspondent à des comportements anormaux du code. Par contre, les valeurs positives ou nulles correspondent à un fonctionnement normal.

```
class Ordre66 {
      public static void main(String argv[]) {
            int a = eliminationDesJedi () ;
            if (a==0) System.out.println(« il faut redoubler d'effort !
      »);
            if (a==-1) System.out.println(« mission accomplie »);
      }

      public static int eliminationDesJedi (String argv[]) {
      int i ;
      for (i=0 ; i<100; i++) {
       int a = rapportDeMission () ;
      if (a==0) {
            System.out.println(«Pas de nouvelle entrée») ;
            i = i+25 ;
                  }
      if (a==-1) {
            System.out.println(« Yoda est introuvable ») ;
            i = i+5 ;
                  }
      if (a==-2) {
            System.out.println(« Certains Jedi ont pris la fuite »);
                  }
      if (a==-3) {
      System.out.println(« tous les jedi sont élmininés ») ;
      return –1 ;
                  }
            }
      return 0 ;
      }

      public static int rapportDeMission () {
            System.out.println(«De quelle mission voulez vous le rapport
      (0-3)»);
            int i = KeyBoard.getInt();
            return –i ;
      }
}
```

Question 1 : Définir les exceptions correspondant à chacune des situations.

Question 2 : Réécrire le code en utilisant les exceptions.

Episode 4: Un nouvel espoir

Il est temps pour le jeune Skywalker de se faire de nouveaux amis. Comment est gérée l'amitié en Java ?

Episode 5: L'empire contre attaque

Avant d'attaquer la base rebelle sur Hoth, les troupes de l'empire se demandent comment fonctionnent les quadripodes AT-AT. Un robot quadripode est composé d'un corps, d'une tête et de 4 pattes articulées lui permettant de se déplacer. Le mouvement d'une patte se décompose en deux phases : en **rétraction**, le pied de la patte est posé sur le sol et le mécanisme de la patte effectue un mouvement de l'avant vers l'arrière, participant ainsi à la propulsion du robot. En **protraction**, la patte lève le pied du sol, le ramène en position antérieure et repose le pied au sol. Une classe Java **Patte** est fournie (ne pas la coder !), munie des méthodes:

public Patte() - le constructeur
public void retracter() qui effectue la phase de rétraction
public void protracter() qui effectue la phase de protraction

Chaque objet **Patte** est contrôlé par un objet de la classe **ContrôleurPatte** qui en invoque les méthodes **retracter()** et **protracter()** . Afin d'assurer l'équilibre du robot il est nécessaire de synchroniser les phases de **protraction** entre les différentes pattes. La règle est la suivante: une patte a le droit de lever le pied du sol uniquement si ses deux voisines ont le pied posé. Cette synchronisation est assurée par un objet de la classe **Privileges** qui gère les états des 4 pattes sous forme d'un tableau de valeurs booléennes (patte en protraction ou non). Avant de commencer la protraction d'une **Patte**, un **ContrôleurPatte** doit en obtenir l'autorisation de la part l'objet **Privileges**. Si elle ne peut être accordée, c'est-à-dire si une des deux pattes voisines est levée, le **ContrôleurPatte** est mis en attente.

Question 1 : donner un diagramme d'objet d'un robot quadripode.

Question 2 : Donnez le code Java des classes **ContrôleurPatte** et **Privileges**. (Note : il est nécessaire que chaque objet **ContrôleurPatte** connaisse son identifiant, c.-à-d. l'indice de la patte qu'il contrôle.)

Question 3: Donnez le code Java d'une classe **ATAT** dont la seule méthode … **live (...)** instancie correctement les objets Patte, Privileges et ContrôleurPatte.

Question 4: On met tous les marcheurs AT-AT dans un conteneur de type ArrayList. Ecrire la classe principale du programme permettant de faire fonctionner tous les quadripod du conteneur.

Question 5: Maintenant que vous connaissez le fonvtionnement d'un ATAT, comment peut-on l'empécher d'avancer?

Episode 6: Le retour du Jedi

Il est temps de décider comment attaquer l'étoile de la mort. Pour cela, il faut organiser les discussions de façon à être sûr que tous les points de vue ont bien été abordés. Les membres du conseil de la rébellion sont autour d'une table. Leur nombre, tenu secret, sera noté N. Chacun possède devant lui un communicateur spécial qui sert également d'interprète. De part et d'autre du communicateur se trouvent deux interrupteurs partagés avec les voisins. Chaque membre du conseil peut réfléchir pendant une durée indéterminée, avoir besoin de prendre la parole (pendant un temps déterminé et fini sinon il s'en va et arrête les discussions), prendre la parole pendant un temps déterminé et fini (sinon il risque de faire fuir les autres membres). Le système possède plusieurs contraintes, lorsqu'un membre du conseil veut parler, il va se mettre en état « recherche à prendre la parole » et attendre que les deux interrupteurs soient libres. Pour parler, il a besoin des deux interrupteurs. Si il n'arrive pas à utiliser un interrupteur, il reste dans l'état « recherche à prendre la parole » jusqu'à une prochaine tentative. Le problème consiste à trouver un ordonnancement permettant à tout le monde de prendre la parole.
Question 1 : Ecrire le diagramme de classe de ce problème.

Question 2 : Ecrire le diagramme d'état transition pour un membre du conseil.

Question 3 : Ecrire un diagramme de séquence représentant une configuration possible d'échange de parole.

Question 4 : De quelle classe doit hériter chaque classe MembreDuConseil pour pouvoir réaliser ceci en programmation ?

Question 5 : Quelles sont les ressources partagées dans cet exemple ? Comment s'assurer que celles-ci sont accédées de façon exclusive par un seul membre à la fois ?

Question 6 : Imaginons que les membres du conseil soient tous des jedi (ils héritent de la Classe Jedi) comment peut-on faire pour résoudre ce problème étant donné qu'il n'y a pas d'héritage multiple en Java ?

Question 7 : Comment pourrait-on transformer la classe Jedi en un autre élément du langage Java afin de conserver l'héritage mis en place lors de la question 4 ?

Episode 7 : Le réveil de la lambda expression:

Voici un code fait avec l'ancienne méthode :

```
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Le bouton a été cliqué selon une méthode de
l'ancienne république");
    }
});
```

Modifier le code de façon à ce qu'il utilise une lambda expression.

Une autre saga cinématographique

Question bonus : Dans Retour vers le futur 3, Marty et le Doc cherchent par tous les moyens à propulser la Delorean à 88 miles à l'heure avec des chevaux ou un train alors qu'il y a du carburant disponible pas loin. Savez vous où ?