PC3 – SIMULATEUR DE RÉSEAU

MODULE TC131D : BASES DE LA PROGRAMMATION OBJET
UV TC131 : OUTILS DE L'INGÉNIEUR

SOMMAIRE

Prérequis 1

Approfondissements 1

Préambule 1

I. Sujet 2

II. Identification des classes 3

III. Exercices pas à pas 4

IV. Traitement des erreurs de transmission 6

PRÉREQUIS

Connaissances de base de la programmation objet et de Java étudiées dans le cours « Découverte de Java »

APPROFONDISSEMENTS

À la suite de ce TD, vous pouvez améliorer vos compétences en travaillant les programmes sur machine.

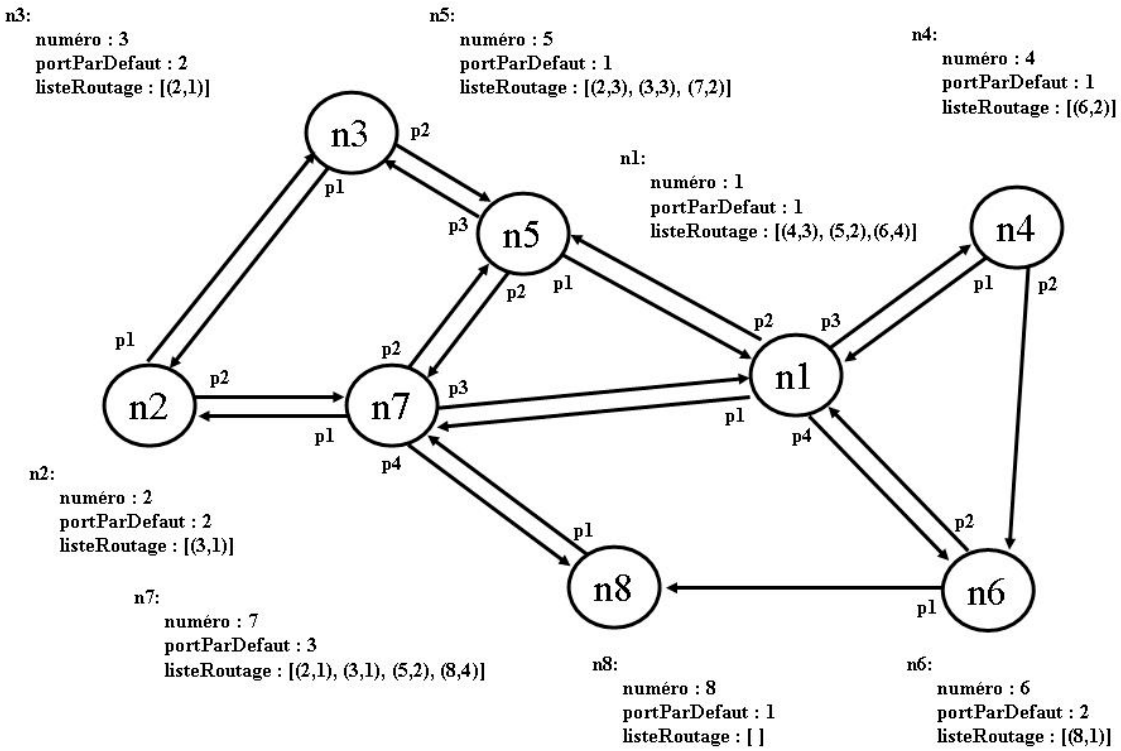
PRÉAMBULE

L'objectif de cette petite classe est de prendre en main les constructions simples de la programmation orientée objet en Java et la réutilisation de classes existantes.

CC BY NC SA

I. SUJET

Durant cette séance, nous allons écrire une partie d'un logiciel de simulation d'un réseau de communication. Elle consistera principalement à réaliser un certain nombre de méthodes permettant l'acheminement d'un message à l'intérieur du réseau. Voici un exemple de réseau, sur le schéma :



Dans un réseau, chaque nœud a un nom, un numéro, un port par défaut et une liste de routage. Le nom d'un noeud correspond au numéro du nœud précédé de la lettre n. Par exemple, le numéro du nœud n5 de l'exemple, est 5.

La liste de routage d'un nœud est utilisée par celui-ci pour décider du port à utiliser pour router un message. Par exemple, pour le nœud n5, la liste de routage dit que pour envoyer un message vers le nœud 2 il faut utiliser le port 3. Et enfin le port par défaut est celui utilisé lorsque le nœud destination n'est pas connu dans la liste de routage. Par exemple, pour notre nœud n5, s'il a un message dont le destinataire est n4, il l'enverra par le port par défaut.

Chaque nœud du réseau connaît son numéro. À la réception d'un message le nœud doit alors:

- Afficher le contenu du message lorsqu'il est le destinataire de ce message.
- Transmettre le message vers un autre nœud lorsqu'il n'est pas le destinataire de ce message. Il dispose pour cela :
    - d'une liste (listeRoutage), qui stocke les correspondances entre numéro de nœud destinataire et numéro de port à utiliser pour émettre en "direction" de ce nœud destinataire

CC BY NC SA

- d'un numéro de portParDefaut pour tous les numéros de nœud destinataire inconnus de la liste de routage.

II. IDENTIFICATION DES CLASSES

À partir du sujet, identifiez les classes candidates et leurs attributs. Spécifiez ensuite au moins les méthodes nécessaires pour assurer le fonctionnement du simulateur. Déterminez, enfin, dans quelle classe doit être codée chacune de ces méthodes.

CC BY NC SA

III. EXERCICES PAS À PAS

Sachant que pour réaliser un logiciel il est possible de trouver plusieurs solutions correctes, dans cette suite d'exercices nous allons vous guider pas à pas pour aboutir à une solution facile à implémenter et respectant les principes de la programmation orientée objet.

Pour cela vous venez de recevoir un code à trou d'une solution que nous vous proposons. Pour les questions qui suivent pensez à utiliser les commentaires des différentes classes, pour comprendre ce qui doit être fait.

A. CODE À TROU

Chaque nœud de ce réseau est une instance de la classe NoeudReseau donnée ci-après :

```
import java.util.LinkedList;

public class NoeudReseau {
      private int numero;                                                    // numero du
NoeudReseau
      private int portParDefaut;                                     // numéro du port par défaut
d'émission d'un message
      private LinkedList <ElementListeRoutage> listeRoutage ;      // LinkedList d'
ElementListeRoutage

         /** Ce constructeur  construit une instance d'un objet  NoeudReseau en lui donnant
         *  un numéro. listeRoutage sera instanciée comme une liste vide.
         *  @param numero numéro du noeud
         *  @param portParDefaut port physique sur lequel seront émis physiquement  tous
         *  les messages  pour lesquels le destinataire n'est pas dans listeRoutage.
         */
         public  NoeudReseau (int numero, int portParDefaut) {
            …
         }

         /** Cette méthode renvoie le numéro  du nœud
         */
         public int getNumero() {
            …
         }

         /** Cette méthode met à jour la valeur de portParDefaut du nœud courant
         *  @param portParDefaut port physique sur lequel devront être émis physiquement  tous
tous
         *  les messages  pour lesquels le destinataire n'est pas dans listeRoutage.
         */
         public void  setPortParDefaut(int portParDefaut) {
               …
         }

         /**  Cette méthode ajoute un élement à la liste listeRoutage (après vérifications)
         */
         public void  majListeRoutage(int destinataire, int port) {
               …
         }


         /** Cette méthode émet un Message "m".
         *  Elle utilisera la listeRoutage du noeud courant pour déterminer le port associé
         *  au destinataire de "m" et émettra "m" en utilisant la méthode émettrePhysiquement
         *  @param m  le Message à émettre
         */
         public void emettre(Message m) {
         …
         }

         /** Cette méthode appellera recevoirPhysiquement pour récupérer un  Message sur un
port
```

```
         *  affiche le Message reçu si le noeud courant est destinataire du message "m"
         *  Elle devra émettre le message m si le noeud courant n'est pas destinataire du
message
         *  @param port  le port ou recevoir
         */
         public void recevoir(int port) {
               …
         }

         private void emettrePhysiquement(int port, Message m) {
               …
         }

     private Message recevoirPhysiquement(int port) {
           …
     }
}
```

Les messages à acheminer seront des instances de la classe Message :

```
public   class Message {

      private int emetteur;
      private int destinataire;
      private String contenu;

       public Message(int emetteur, int destinataire, String contenu) {
         this.emetteur = emetteur;
         this.destinataire = destinataire;
         this.contenu = new String(contenu);
      }

       public int getDestinataire() {
         return destinataire;
      }

       public int getEmetteur() {
         return emetteur;
      }

       public String getContenu() {
         return contenu;
      }

   }
```

Les éléments de la liste listeRoutage d'une instance de NoeudReseau seront des instances de la classe ElementListeRoutage :

```
public class ElementListeRoutage {
   private int destinataire;
   private int port;

   public ElementListeRoutage(int destinataire, int port) {
      this.destinataire = destinataire;
      this.port = port;
   }

   public int getDestinataire() {
      return destinataire;
   }

   public int getPort() {
      return port;
   }
}
```

CC BY NC SA

Un constructeur et quelques méthodes de la classe générique LinkedList

```
public class LinkedList  <E> {
      // Constructs and returns an empty list.
      public LinkedList <E> () { …}

      // Appends the specified element o to the end of this list.
      public boolean  add(E o) {…}

      // Returns the element at the specified position in this list.
      public E get(int index)  {…}

      // Replaces the element at the specified position in this list with the specified
element.
         // Returns the element previously at the specified position.
      public E set(int index, E element)  {…}

      // Returns the number of elements in this list.
     public int size() {…}

      // …
}
```

B. LES QUESTIONS

Complétez le constructeur de la classe NoeudReseau

Écrivez la méthode getNumero de la classe NoeudReseau

Écrivez la méthode setPortParDefaut de la classe NoeudReseau

Écrivez la méthode majListeRoutage de la classe NoeudReseau

Écrivez les méthodes emettre et recevoir de la classe NoeudReseau

IV. TRAITEMENT DES ERREURS DE TRANSMISSION

Les messages précédents transitent sur des supports physiques de transmission perturbés. On souhaite augmenter le degré de confiance que l'on a dans le contenu du message en y ajoutant un attribut "controleTransmission" de type int. Il ne s'agit que de se prémunir de la mauvaise qualité des supports physiques de transmission et non pas de manœuvres malveillantes.

Définissez les attributs et le constructeur de Message.

La méthode d'instance hashCode() de la classe String renvoie une valeur int qui est fonction du texte de la String.

Le constructeur de Message fera qu'à l'instanciation d'un Message, l'attribut int controleTransmission soit initialisé à : emetteur + destinataire + contenu.hashCode().

Définissez une méthode d'instance checkMessage dans la classe Message.

Cette méthode d'instance (sans paramètre) vérifiera si le message a de "bonnes chances" d'être valide. Cette méthode vérifie la correspondance entre la valeur de l'attribut controleTransmission du message et la valeur calculée à partir des autres attributs du message. Dans le cas où il y a égalité entre la valeur calculée et l'attribut controleTransmission du message elle renverra "true", et "false" autrement.

CC BY NC SA

Il reste à modifier la méthode recevoir de la classe NoeudReseau de façon à ce qu'elle ne fasse rien si le message reçu est invalide.

Modifiez la méthode recevoir de la classe NoeudReseau.

CC BY NC SA