Corrigé du TD d'algorithmique avancée n°3

1 SAUVEZ-LES TOUS !

Corrigé dans le TD précédent...

2 4CHAN 

4chan est un forum assez populaire sur Internet. Sa particularité est de garantir l'anonymat complet de ses contributeurs : aucune adresse IP n'est conservée en mémoire, et les sujets les moins populaires sont remplacés par les plus populaires (pas de stockage à long terme des messages).

Nous allons tenter de reproduire le comportement de ce forum avec des listes. Dans un premier temps, nous définirons une liste **SujetDeDiscussion**, qui permettra de stocker tous les messages relatifs à un sujet de conversation précis. Puis, nous devrons trouver un moyen de stocker au maximum 150 **SujetDeDiscussion**, et prévoir les fonctions permettant d'ajouter un **SujetDeDiscussion** tout en en supprimant un.

1. Définissez la structure **ListeMessage** qui contient un champ *Auteur* (de type String), un champ *Contenu* (de type String), et ?... Puis, définissez la structure **SujetDeDiscussion**, qui contient un champ *DateDeDerniereContribution* (de type entier), un champ *Sujet* (de type String), et un champ *Messages* (de type **ListeMessage**) qui pointera vers le premier message du sujet de discussion.

On suit les instructions... tout d'abord, la structure ListeMessage. Comme c'est une liste, il ne faut pas oublier un suivant !

```
public class ListeMessage
{
    String message;
    String auteur;
```

```
    ListeMessage message_suivant;
}
```

Le sujet de discussion permettra d'apporter des informations sur le sujet en cours... A quand remonte le dernier message de ce sujet, et quels sont les messages de ce sujet.

```
public class SujetDiscussion
{
    int dateDeDerniereContribution;
    String sujet;
    ListeMessage messages;
}
```

2. Faîtes une fonction **AjouterMessage**, qui prend en paramètre un **SujetDeDiscussion**, un nom d'auteur et un message, et qui ajoute un message au sujet de discussion (vous devrez utiliser la fonction fictive Instant(), qui renvoie un entier avec la date et l'heure du jour (à la milliseconde près).

```
public static void AjouterMessage(SujetDiscussion sd, String mess, String aut)
{
    //On rajoutera le nouveau message à la fin de la liste de messages de
    //sd (qui est un SujetDiscussion)

    //D'abord, on créé un nouveau message
    ListeMessage m = new ListeMessage();
    m.auteur = aut;
    m.message = mess;
    m.message_suivant = null; //Ce nouveau message sera placé à la fin de la
liste

    //On parcourt la liste de messages de SujetDiscussion pour ajouter notre
    //message à la fin
    //MAis attention, il faut vérifier s'il n'y a AUCUN message dans la liste...
    //si c'était le cas, on rajoute simplement notre message

    //La liste est-elle vide ?
    if(sd.messages != null) //non
    {
        ListeMessage parcours = sd.messages;
        while(parcours.message_suivant != null) //on parcourt jusqu'au dernier
élément
        {
            parcours=parcours.message_suivant;
        }
        //et on rajoute notre élément à la fin
        parcours.message_suivant=m;
    }
    else //oui
    {
        sd.messages = m;
    }

    //enfin, on met à jour la date de dernière contribution avec cette fonction
fictive
    //qui n'existe pas en Java...
    sd.dateDerniereContribution = Instant();
}
```

3. Vous devez trouver maintenant un moyen de stocker au plus 150 messages de discussion dans une structure. Avant de vous lancer, réfléchissez au fait que vous devrez faire une fonction **AjouterSujetDeDiscussion**, qui prendra en paramètre un sujet, un auteur et un message, qui recherchera dans la structure choisie le sujet de discussion avec la date de contribution la plus ancienne, et le remplacera par le nouveau sujet de discussion.

On pourrait faire un tableau ici (de 150 cases). La fonction AjouterSujetDeDiscussion recherchera dans le tableau le sujet de discussion le plus vieux (la valeur de dateDernierContribution sera la plus petite) et le remplacera par notre sujet de discussion.

On définit d'abord une fonction permettant de trouver, dans un tableau de SujetDiscussion, le sujet le plus ancien.

```
public static int position_plus_ancien(SujetDiscussion[] tablo)
{
    int position_ancien = 0;

    for(int i=1; i<tablo.length; i=i+1)
    {
        if(tablo[i].dateDeDerniereContribution <
tablo[position_ancien].dateDeDerniereContribution)
        {
            position_ancien = i;
        }
    }

    return position_ancien;
}
```

Puis, on fait notre fonction.

```
public static void AjouterSujetDiscussion(SujetDiscussion[] t, String sujet,
String message, String auteur)
{
    //on créé un nouveau sujet de discussion
    SujetDiscussion s = new SujetDiscussion();
    s.sujet = sujet;

    //on y ajoute le message que l'on veut
    AjouterMessage(s, message, auteur);

    //Puis, on recherche dans le tableau le plus ancien sujet
    int plus_ancien = position_plus_ancien(t);

    //et on remplace le sujet le plus ancien par notre sujet
    t[plus_ancien] = s;
}
```

4. Imaginons que chaque **SujetDeDiscussion** possède un nouveau champ *NumeroIndentiteUnique*, qui l'identifie de façon unique par rapport aux autres **SujetDeDiscussion**. De quoi auriez vous besoin pour faire une fonction qui ajoute rapidement un message à un sujet de discussion qu'on ne possède pas (c'est à dire que l'on ne possède pas la structure **SujetDeDiscussion** à proprement dit, mais on peut posséder autre chose, comme par exemple le *NumeroIndentiteUnique* du sujet de discussion auquel on souhaite ajouter un message.

Attention : prenez en compte le fait que pleins de gens sont connectés en même temps sur votre site et y font des opérations.

Il faudrait que le tableau de sujet de discussion soit trié selon leur numéro d'identité, et l'on

pourrait faire une recherche dichotomique. Il faut aussi penser au fait que, pendant qu'on recherche notre sujet, d'autres personnes ajoutent des messages dans le tableau, et ce dernier va être modifié (des sujets de discussion pourraient être effacés). Il faut en fait éviter que plusieurs opérations puissent être faites en parallèle sur le tableau pour éviter ces conflits...