CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

Contrôle terminal info2 / Semestre 3

M3101 : Principes des systèmes d'exploitation

Nom du responsable :	**François MERCIOL, Matthieu Le Lain**
Date du contrôle :	**20 octobre 2020**
Durée du contrôle :	**1 heure 30 minutes**
Nombre total de pages :	**10 pages**
Impression :	**Recto – Verso**
Documents autorisés :	**Tous**
Calculatrice autorisée :	**non**
Réponses :	**Sur le sujet (en 2 parties séparées)**

Conseils :

- Indiquez **votre nom** sur cette feuille **dès qu'elle vous aura été donnée**.
- Afin d'éviter la copie, toute réponse non justifiée sera considérée comme nulle.
- **Les parties sont indépendantes** (commencez par la plus simple pour vous).
- Il est demandé des réponses à la fois claires et concises.
- **Lisez en entier** le contrôle avant de commencer à répondre.
- **Ne restez pas bloqué**, vous pourrez revenir sur une question difficile par la suite.
- **Ne brûlez pas toute votre énergie**, des calculs longs peuvent vous rapporter moins de points que la réponse à des questions de réflexion.
- **Conservez du temps pour chaque partie**
  (90 min / 20 pts => pas plus de 4 minutes 30 par point)

Le devoir se compose de 3 parties :

- [4 pts] questions de cours (QCM : Questions à Choix Multiple / 10 min)
- [8 pts] « Faisons la fête » communication réseau (problème / 40 min)
- [8 pts] « Faisons la fête » analyse des messages (exercices / 40 min)

CE CORRIGÉ EST INDICATIF. D'AUTRES RÉPONSES PEUVENT ÊTRE CONSIDÉRÉES COMME JUSTES.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

« Faisons la fête »

Les résultats scientifiques sont indiscutables, l'humain serait un être sociable. Nous proposons d'utiliser une dernière fois le numérique pour l'aider à se retrouver. Pour cela, nous disposons de 2 dispositifs :

- un repas partagé, lieu de rendez-vous convivial
- une proposition de participation.

Traces du terminal A

```
$ java -cp ../build/class party.Meeting 9090
Menu :
felix apporte du fromage

Menu :
laurent apporte du pain
felix apporte du fromage

Menu :
laurent apporte du pain
felix apporte du fromage
tranber apporte la modération
```

Traces du terminal B

```
$ java -cp ../build/class party.HaveFun 127.0.0.1 9090 "du fromage"
Merci pour apporter du fromage
```

La gestion du repas fait appel à 2 classes : Meeting et Menu. La proposition de participation utilise la classe HaveFun.

Dans la première partie, nous nous concentrons sur la communication

Dans la seconde, nous abordons l'analyse de texte.

2.) [8 pts] « Faisons la fête » communication réseau (problème / 40 min)

Nous vous fournissons le code de deux classes : Meeting et HaveFun. Malheureusement un virus s'y est introduit et a fait disparaître certaines lignes de code. Votre mission, pour sauver l'humanité, est de les retrouver.

Les questions se trouvent sur les pages suivantes. Vous devrez répondre sous chaque question. En cas de manque de place vous pourrez utiliser une zone d'extension en fin de partie.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

```
15.public class Meeting {
16.  public static final int maxMsg = 1024;
17.  Menu menu = new Menu ();
18.  DatagramSocket socket;
19.
20.  public void receive ()
21.      throws IOException {
22.    DatagramPacket packet = // …
23.    // …
24.    String received = // …
25.    String answer = menu.parse (received);
26.    byte[] buf = answer.getBytes ();
27.    packet.setData (buf, 0, buf.length);
28.    socket.send (packet);
29.  }
30.
31.  static public void main (String[] args) {
32.    try {
33.      Meeting meeting = new Meeting (Integer.parseInt (args [0]));
34.      for (int number = 0; ; ) {
35.        int newNumber = meeting.menu.getNumber ();
36.        if (number != newNumber)
37.          System.err.println (meeting.menu.getMenu ());
38.        number = newNumber;
39.        Thread.sleep (1000);
40.      }
41.    } catch (Exception e) {
42.      System.err.println ("Usage: java Meeting port");
43.    }
44.  }
45.}
```

```
46.public class HaveFun {
47.  public HaveFun (String address, int port, String msg)
48.      throws IOException {
49.    byte[] buf = (msg).getBytes ();
50.    DatagramPacket packet =
51.      new DatagramPacket (buf, buf.length,
52.                          InetAddress.getByName (address), port);
53.    DatagramSocket socket = new DatagramSocket ();
54.    socket.send (packet);
55.
56.    packet.setData (new byte[Meeting.maxMsg], 0, Meeting.maxMsg);
57.    socket.receive (packet);
58.    String received = new String (packet.getData (), 0, packet.getLength ());
59.    System.err.println (received);
60.  }
61.
62.  static public void main (String[] args) {
63.    try {
64.      String msg = System.getProperty ("user.name")+" "+
65.                   (args.length == 2 ? Menu.menuToken
66.                                     : (Menu.apporteToken+" "+args [2]));
67.      new HaveFun (args [0], Integer.parseInt (args [1]), msg);
68.    } catch (IOException e) {
69.      System.err.println ("Usage: java HaveFun ip port [\"participation\"]");
70.    }
71.  }
72.}
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.A) [2 pt] Quelle communication ?

Indiquez le type de communication (nature de socket) utilisé dans les programmes. Quels en sont les avantages et inconvénients ?

La communication est en mode « non connecté ».
Ce sont des DatagramSocket qui sont utilisé dans ces programmes (1)

L'avantage est que la communication est asynchrone.
L'inconvénient est que l'on a pas de garantie que les messages sont arrivés. (1)

2.B) [2 pt] Bonne réception

Reconstituez les lignes manquantes de la méthode « receive » de la classe Meeting.

Il faut déclarer un message, l'attendre, puis convertir le résultat en chaîne de caractères.

```
public void receive ()
     throws IOException {
  DatagramPacket packet =
       new DatagramPacket (new byte[maxMsg], maxMsg); (1)
  socket.receive (packet); (.5)
  String received =
      new String (packet.getData (), 0, packet.getLength ()); (.5)
  String answer = menu.parse (received);
  byte[] buf = answer.getBytes ();
  packet.setData (buf, 0, buf.length);
  socket.send (packet);
  }
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.C) [2 pt] Se mettre à la tâche

Meeting est actuellement sourd, car rien ne fait appel à la méthode de réception. Créez un constructeur qui crée le serveur socket ainsi qu'une activité qui assure une réception sans fin.

Il faut initialiser le socket et lancer une tâche pour écouter les messages arrivants.

```
public Meeting (int port)
    throws IOException {
  socket = new DatagramSocket (port); (.5)
  (new Thread () { (.5)
    public void run () { (.5)
      try {
        for (;;)
          receive (); (.5)
      } catch (IOException e) {
        e.printStackTrace ();
      }
    }
  }).start ();
}
```

2.D) [2 pt] Variantes

Indiquez ci-dessous deux autres façons de créer une tâche.

Il y a 3 façons de créer une tâche :
*** par dérivation de la classe Thread (.5)**
*** par implantation de la classe Runnable puis création d'un exemplaire de la classe Thread. (.5)**
*** par création d'une classe anonyme dérivé de Thread. (.5)**

(.5 pour code)

Corrigé (Les réponses apparaissent avec ce fond de caractère)

Corrigé (Les réponses apparaissent avec ce fond de caractère)

3.) [8 pts] « Faisons la fête » analyse des messages (exercices / 40 min)

Lors de la communication entre un futur participant et l'organisateur de la fête, il y a l'échange de messages. Voici des exemples de message envoyé par les participants qui proposent leur contribution :

```
laurent apporte du pain
felix apporte du fromage
tranber apporte
kfsqdjh fsdkks=fqj_è'(ç_'è987 fkdsjh 0987ç_#{[
```

Voici des exemples de réponse de l'organisateur :

```
Merci pour apporter du pain.
Ce choix est déjà pris.
Ne venez pas les mains vides.
Pardon ?
```

Il est également possible de demander l'état du menu par :

```
laurent menu
felix menu
```

La réponse du serveur pourra être :

```
Menu :
laurent apporte du pain
felix apporte du fromage
```

Pour vous aider, le squelette de la classe Menu est fourni page suivante.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

```
public class Menu {

  Hashtable <String, String> menu = new Hashtable <String, String> ();

  public synchronized String parse (String msg) {
    Matcher m = // …
    boolean ok = // …
    if (!ok)
      return "Pardon ?";
    String sender = // …
    String cmd = // …
    String arg = // …
    if (apporteToken.equals (cmd)) {
      if (arg != null)
        arg = arg.trim ();
      if (arg == null || "".equals (arg))
        return "Ne venez pas les mains vide.";
      if (add (sender, arg))
        return "Merci pour apporter "+arg+".";
      return "Ce choix est déjà pris.";
    }
    return getMenu ();
  }

  public boolean add (String who, String what) {
    String old = menu.get (who);
    if (old != null && what.equals (old))
       return true;
     if (menu.contains (what))
       return false;
     menu.put (who, what);
     return true;
   }

   public synchronized String getMenu () {
     String result = "Menu :\n";
     for (Map.Entry<String, String> entry : menu.entrySet()) {
       String who = entry.getKey ();
       String what = entry.getValue ();
       result += who+" apporte "+what+"\n";
     }
     return result;
   }

   public synchronized int getNumber () {
     return menu.size ();
   }
}
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

3.A) [1 pt] Expression régulière

Proposez une expression régulière pour reconnaître les demandes de participants (à la fois proposition de participation et demande de menu, sachant que l'on souhaite extraire 3 informations : l'émetteur, la commande et d'éventuels arguments.

Pour gérer les espace, nous utiliserons \s et pour les caractères d'un mot \w. Les parenthèses isolent les composants que l'on souhaite extraire. Les espaces embarqués dans le dernier groupe optionnel seront supprimé en ligne 87. L'expression est donc :

\s*(\w)+\s+(menu|apporte)(\s+.*\w+)?\s*

3.B) [1 pt] Expression régulière dans le code Java

L'on vous demande cette fois d'écrire la même expression régulière mais dans une chaîne de caractères Java.

Dans le code java, l'expression est écrite dans une chaîne de caractères, il faut donc échapper les caractères \. L'expression régulière devient :

"\\s*(\\w)+\\s+(menu|apporte)(\\s+.*\\w+)?\\s*"

3.B) [3 pt] Utilisation dans Java

Proposez le début de la méthode parse (il y a deux classes qui interviennent dans l'analyse d'une expression régulière)

```
final static String menuToken = "menu";
final static String apporteToken = "apporte";
final static String pseudoToken = "\\w+";
final static String cmdToken = menuToken+"|"+apporteToken;
final static String argToken = ".*";
final static String regexp =
  "\\s*("+pseudoToken+")\\s+("+cmdToken+")(\\s+"+argToken+")?\\s*";
final static Pattern msgPattern = Pattern.compile (regexp); (1)

public synchronized String parse (String msg) {
    Matcher m = msgPattern.matcher (msg); (1)
    boolean ok = m.matches (); (1)
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

3.B) [1 pt] Synchronisation

Pourquoi les méthodes parse et getMenu sont-elles synchronisées ?

Car 2 tâches s'exécutent en parallèle : celle qui attend une demande de participant et celle qui trace l'évolution du menu dans la console du serveur. Il pourrait donc y avoir en même temps une modification du menu (avec parse) et une lecture du menu (avec getMenu).

3.B) [2 pt] Et la date ?

Pour une meilleure organisation, on projette d'ajouter une proposition de date au format suivant « AAMMJJ hh:mm:ss »

Indiquez une expression régulière permettant de reconnaître que le format est juste.

En première approximation nous nous limiterons à l'analyse des chiffres et des caractères de séparation. A ce niveau, il n'est pas possible de faire un contrôle de la cohérence (vérification des années bissextiles, ...).

"\\d{6}\\s+\\d{2}:\\d{2}:\\d{2}"

Indiquez une autre façon d'analyser une date avec une classe Java adaptée

A la place d'utiliser les expressions régulières (classes Pattern et Matcher) on utilise la classe du package java.text qui contrôle la sémantique (sens donné aux chiffres).

```
SimpleDateFormat sdfl =
 new SimpleDateFormat ("yyMMdd-HH:mm:ss");
```