Examen

Sujet 1

Aucun document autorisé. Seul matériel autorisé : stylos, crayons et règle. Aucun autre matériel autorisé. Le barême est donné à titre indicatif. Durée de l'épreuve : 2h00.

Exercice 1 : [Analyse de code] (5 points)
Soit le programme suivant.

Début `Thread1.java`

```
class Thread1 {
    private int num;

    private Thread1(int n) {
        num = n;

        Travail1 tl1 = new Travail1(this);
        Thread t1 = new Thread(tl1);
        t1.start();

        Travail2 t2 = new Travail2(this);
        t2.start();

        try { t1.join(); }
        catch(InterruptedException e) { System.exit(-1); }

        (new Travail3(tl1.getNum(),t2.getNum())).start();
    }

    int getNum() { return num; }

    void setNum(int n) { num = n; }

    public static void main(String[] args) {
        try {
            new Thread1(Integer.parseInt(args[0]));
        } catch(NumberFormatException e) {}
    }
}

class Travail1 implements Runnable {
```

```
    private Thread1 t;
    private boolean fin = false;
    private int monNum = 0;

    Travail1(Thread1 t) {
        this.t = t;
    }

    public int getNum() { return monNum; }

    public void run() {
        while (!fin) {
            while (((t.getNum()%2) == 0) && (t.getNum() != 1))
                try { Thread.sleep(50); }
                catch(InterruptedException e) {}
            if (t.getNum() == 1) {
                System.out.println("T1 : fin => " + monNum);
                fin = true;
            } else {
                int n = t.getNum();
                System.out.println("T1 : " + n + " -> " + (n-1));
                monNum++;
                t.setNum(n - 1);
            }
        }
    }
}

class Travail2 extends Thread {
    private Thread1 t;
    private boolean fin = false;
    private int monNum = 0;

    Travail2(Thread1 t) {
        this.t = t;
    }

    public int getNum() { return monNum; }

    public void run() {
        while (!fin) {
            while (((t.getNum()%2) == 1) && (t.getNum() != 1))
                try { Thread.sleep(50); }
                catch(InterruptedException e) {}
            if (t.getNum() == 1) {
                System.out.println("T2 : fin => " + monNum);
                fin = true;
            } else {
                int n = t.getNum();
                System.out.println("T2 : " + n + " -> " + (n/2));
                monNum++;
                t.setNum(n/2);
            }
```

```
        }
    }
}

class Travail3 extends Thread {
    private int i1, i2;

    Travail3(int n, int m) {
        System.out.println("T3 : " + n + " et " + m);
        i1 = n;
        i2 = m;
    }

    public void run() {
        System.out.println("T3 : " + (i1+i2));
    }
}
```

Fin `Thread1.java`

Donner l'affichage effectué par l'exécution de ce programme résultant de l'appel suivant : `java Thread1 15`.

Exercice 2 : [Client-Serveur] (15 points)

On s'intéresse à la programmation d'un serveur de variables partagées. Celui-ci fonctionne de la façon suivante. Le serveur stocke des variables qui sont ensuite partagées par tous les clients connectés. Les variables sont identifiées par des chaînes de caractères de taille quelconque et peuvent contenir uniquement des entiers (de type `int`). Chaque client peut demander au serveur les services suivants : créer une nouvelle variable, affecter une valeur à une variable et récupérer la valeur d'une variable. Si un client demande la création d'une variable qui existe déjà, le serveur doit l'informer de ce fait. L'affectation ou la récupération auprès de variables n'existant pas doit être signalée au client par un message approprié. Il peut y avoir un nombre quelconque de clients connectés au serveur.

On rappelle que la méthode `substring(int beginIndex)` appelée sur une chaîne de caractères renvoie la sous-chaîne commençant à la position `beginIndex` (inclu) et se terminant à la fin de la chaîne initiale. Par exemple, si `str` contient la chaîne `"bonjour"`, alors `str.substring(2)` renvoie la chaîne `"njour"`. La méthode `substring(int beginIndex, int endIndex)` renvoie la sous-chaîne comprise entre les indices `beginIndex` (inclu) et `beginIndex` (exclu).

1. Définir un protocole de communication.
2. Donner le code complet du serveur (gestion des connexions, des variables partagées, de la communication avec les clients, etc.)
3. **[Bonus]** Que faudrait-il modifier pour que le serveur puisse supprimer les variables qui ne sont plus utilisées.