Examen

Sujet 4

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 `Test.java`

```
class Test {
    private boolean var;

    private Prog1 ex1;
    private Prog2 ex2;

    private Thread moi;

    private Test(int n) {
        // La methode Thread.currentThread() permet de recuperer un objet de
        //  la classe Thread representant le thread en train de s'executer.
        moi = Thread.currentThread();
        ex2 = new Prog2(this);
        ex2.start();
        ex1 = new Prog1(this,n);
        ex1.start();

        System.out.println("Valeur = " + n);
        var = true;
    }

    void put(int n) {
        if (var) ex1.set(n);
        else ex2.set(n);
        var = !var;
    }

    boolean get() {
        return var;
    }

    void attendre() {
```

```
        try { moi.join(); }
        catch(InterruptedException e) {}
    }

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

class Prog1 extends Thread {
    private Test t;
    private int num1, num2 = 12;

    Prog1(Test t, int n) {
        this.t = t;
        num1 = n;
    }

    public void set(int n) {
        num2 = n;
    }

    public void run() {
        t.attendre();
        while (num2 > 1) {
            while (t.get())
                try { Thread.sleep(50); }
                catch(InterruptedException e) {}
            int n = num1 / num2;
            System.out.println("Resultat :  " + n);
            num1 = num1 % num2;
            t.put(0);
        }
    }
}

class Prog2 extends Thread {
    private Test t;
    private int num1;

    Prog2(Test t) {
        this.t = t;
        num1 = 4;
    }

    public void set(int n) {
        num1 = num1 - 1;
    }

    public void run() {
        t.attendre();
```

```
        while (num1 > 0) {
            while (!t.get())
                try { Thread.sleep(50); }
                catch(InterruptedException e) {}
            t.put(calcul());
        }
    }

    private int calcul() {
        int r = 1;
        for (int i = 0; i < num1; i++)
            r = r * 2;
        return r;
    }
}
```

Fin `Test.java`

Donner l'affichage effectué par l'exécution de ce programme résultant de l'appel suivant : `java Test 19`.

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

On s'intéresse à la programmation d'un serveur de connexions qui permet de mesurer le trafic. Ce serveur, que l'on nommera `ServConnect` dans la suite, fonctionne de la façon suivante. Lorsqu'un client se connecte à `ServConnect`, celui-ci indique d'abord à quel serveur distant il souhaite se connecter, puis sur quel port. Notre serveur `ServConnect` établit alors la communication avec ce serveur distant puis, au fur et à mesure, transmet les messages venant du serveur distant vers le client, et réciproquement les messages du client vers le serveur distant. On supposera que la conversation entre le client et le serveur distant est toujours parfaitement alternée : le client envoie un message, puis attend un message du serveur distant, etc. La communication avec le serveur distant se fait sous forme de chaînes de caractères terminées par un `'\n'`. C'est toujours le client qui commence la conversation. Lorsque le client le souhaite, il peut mettre fin à la communication avec le serveur distant en envoyant à `ServConnect` un message approprié. Plusieurs client peuvent utiliser `ServConnect` en même temps pour se connecter à différents serveurs distants. Notre serveur `ServConnect` doit mesurer le trafic en comptant le nombre total de messages envoyés par tous les clients confondus.

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 `endIndex` (exclu).

1. Définir un protocole de communication entre le client et `ServConnect`. Ne pas traiter la validité des communications entre le client et le serveur distant.
2. Donner le code complet du serveur `ServConnect` (gestion des connexions, de la communication avec les clients, comptabilité des messages envoyées, etc.)
3. **[Bonus]** Que faudrait-il modifier/supposer pour que les conversations soient plus complexes qu'un message reçu par message envoyé.