Pale INF422 – Composants d'un système informatique
INF422 Final Exam – Components of a Computing System

2012–2013

- L'examen dure 3 heures.
  The exam lasts 3 hours.
- Tous documents autorisés.
  All documents are authorized.
- Il est impératif de commenter les programmes et de justifier vos réponses.
  Programs must be commented and every answer must be justified.

Sujet en français

1 Statistiques sur la mémoire

Question 1.1

La commande `cat /proc/meminfo` produit la sortie suivante, après filtrage pour éliminer les informations dépassant le cadre du cours :

```
MemTotal:        8078332 kB
MemFree:          917388 kB
SwapTotal:      10149884 kB
SwapFree:       10141236 kB
```

Expliquez chaque ligne de cette sortie.

Question 1.2

Les lignes ci-dessous analysent syntaxiquement le fichier `/proc/meminfo` (on appelle cette analyse syntaxique le parsing). Insiprez vous de celles-ci pour programmer l'affichage du total de mémoire utilisable par le noyau et les processus (mémoire physique plus "swap", libre ou occupée).

```
// Ouverture du fichier /proc/meminfo, mise en place du scanner
File file = new File("/proc/meminfo");
Scanner scan_file = new Scanner(file);

scan_file.nextLine();
String second_line = scan_file.nextLine();

// Mise en place d'un second scanner, pour decomposer
// cette ligne en deux elements, delimites par ':'
Scanner scan_line = new Scanner(second_line);
scan_file.useDelimiter(":");

// la premiere partie contient la chaine "MemFree:", la seconde
// partie contient l'information recherchee.
scan_line.next();
String free_memory = scan_line.next();
```

```
// On ferme ce qu'on a ouvert, et en renvoie la qantité de memoire libre
scan_line.close();
scan_file.close();
```

Question 1.3

On souhaite afficher ces statistiques dans une interface graphique disposant d'une boite de texte.

Décrivez un projet d'application Android qui réalise cette interface graphique et ces fonctionnalités, en indiquant les différents fichiers composant le projet et en détaillant le contenu de ces fichiers (code source, disposition et description de l'interface).

Question 1.4

Modifiez votre application en ajoutant un bouton pour rafraîchir l'information affichée : un click simple sur le bouton doit relire le fichier et mettre à jour la boite de texte.

Question 1.5

Programmez la même fonctionnalité que le code Java de la question 2 à l'aide d'un script shell.

Vous pourrez utiliser les commandes `cut` ou `sed` pour isoler des mots ou des nombres dans les lignes du fichier (voir les corrections du TD4 pour un exemple d'utilisation de `cut`).

Question 1.6

Écrivez un nouveau script shell qui prend un entier n en argument, et qui affiche la statistique correspondant à la n-ième ligne de `/proc/meminfo`. Par exemple, si cette n-ième ligne correspond à `MemTotal`, le script doit afficher `8078332` (d'après la sortie de la commande `cat /proc/meminfo` ci dessus).

2 Dégradé de gris

On complète l'exercice 5 du TD7 en réalisant une méthode Java qui lit l'image courante de l'écran dans le frame buffer, et la convertit en dégradé de gris.

Question 2.1

Écrivez une fonction Java qui prend un entier de type `short` codant un pixel au format RGB en 5 :5 :5 :1 bits (le bit de poids faible n'étant pas utilisé), et retourne un entier de type `short` codant le même pixel en dégradé de gris. Pour réaliser une teinte de gris, on effectuera la moyenne des 3 composantes R, G et B que l'on affectera à chaque composante de couleur du pixel résultant.

Question 2.2

Implémentez une fonction Java qui convertit l'ensemble de l'image à l'écran en dégradé de gris.

3 Censure en tous genres

Question 3.1

À l'aide de la commande `sed`, remplacez toutes les occurrences de `tonnerre de Brest` dans une ligne de texte par `######## ## #####`, en préservant les autres mots de la ligne.

Question 3.2

Réalisez la même opération en Java, en utilisant la classe `Scanner` du premier exercice, et les méthodes suivantes pour convertir une chaînes de caractères en tableau de caractères, et réciproquement :

```
String s1 = "hello world";
char[] a1 = s1.toCharArray();

char[] a2 = {'h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd'};
String s2 = new String(a2);
```

Question 3.3

On souhaite modifier le serveur du TD3 (messagerie instantanée) pour analyser le texte des messages et y censurer les mots interdits par l'opérateur de la messagerie. Il serait suffisant de programmer directement le filtrage dans le code du serveur, mais cela compliquerait la maintenance ultérieure du programme. On préfère réaliser une classe `CensoredSocket` qui se comporte en tout points comme la classe `Socket`, sauf que tout mot interdit est automatiquement censuré entre l'émetteur et le recepteur.

Quelle construction du langage Java convient-il d'utiliser pour construire une telle classe sans reprogrammer les fonctionalités d'une socket ?

Comment modifier la classe `Net` du TD3 pour utiliser la classe `CensoredSocket` ?

Quelles méthodes des classes `ServerSocket` et `Socket` faut-il reprogrammer pour que la classe `Net` effectue des communications censurées ?

Les méthodes de la classe `Net` sont rappelées ci-dessous, le code complet est fourni dans les corrections du TD3.

```
final class Net {
    static public ServerSocket createServer(int server_port) { /* ... */ }
    static Socket acceptConnection(ServerSocket s) { /* ... */ }
    static Socket establishConnection(String ip, int port) { /* ... */ }
    static PrintWriter connectionOut(Socket s) { /* ... */ }
    static BufferedReader connectionIn(Socket s) { /* ... */ }
}
```

Question 3.4

Expliquez comment, avec deux sockets que l'on appellera "clair" et "filtré", il est possible d'insérer l'opération de censure de manière transparente entre un émetteur et un recepteur utilisant la classe `CensoredSocket`.

Question 3.5

Comme l'opération de censure est concurrente à l'émission et à la réception de messages, il est nécessaire d'utiliser un thread dédié dans les méthodes identifiées ci-dessus. Ce thread est créé à la création d'une instance (un objet) de la classe `CensoredSocket`, attend des lignes de texte à l'aide d'un `BufferedReader` sur le socket "clair", et répond en émettant les mêmes lignes après censure sur le socket "filtré".

Programmez la classe `CensoredSocket`.