CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

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

U.B.S. - I.U.T. de Vannes

Département Informatique
Contrôle de 2e année
Date : 20/10/2014

M3101 - Principe des systèmes d'exploitation

Devoir surveillé

Responsable(s) et auteur(s) : F. Merciol – M. Le Lain

Documents personnels et calculatrice autorisés
Téléphone interdit — Durée : 2h
Les barèmes sont donnés à titre indicatif — 11 page(s) de texte

---

Conseils :

- Indiquez **votre nom** sur chaque feuille à rendre **dès qu'elles vous sont données**.
- 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**
  (120 min / 20 pts => pas plus de 6 minutes par point)

Le devoir se compose de 3 parties :

1) [4 pts] questions de cours (QCM : Questions à Choix Multiple)
2) [8 pts] résolution de problème sur les tâches (en java)
3) [8 pts]

Ce corrigé est donné à titre indicatif. D'autres réponses peuvent être considérés comme justes.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

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

1.) [4 pts] question de cours (QCM : Questions à Choix Multiple)

✗ Cochez la case réponse

Question	✾	✱	❀	✲
Le mode "a+" ouvre un fichier en ✾) lecture au début ✱) écriture au début en fin ❀) écriture seule ✲) lecture en fin	○			
Bélády a montré ✾) l'efficité du swap ✱) des lenteurs avec plus de mémoire ❀) l'efficacité des caches ✲) l'efficacité de la segmentation		○		
Les données d'une mémoire DRAM ✾) s'effacent par UV ✱) s'effacent rapidement ❀) sont permanentes ✲) sont d'accès lent		○		
Les premières mémoires sans consommation pour la conservation des bits, utilisaient des ✾) boucles en cuivre ✱) rayons de lumière ❀) anneaux magnétiques ✲) petites réserves d'eau			○	
Le nombre de machines maximum d'une classe A est de ✾) 16 777 214 ✱) 65 534 ❀) 2 097 152 ✲) 254	○			
La référence ISO 8859-15 désigne ✾) l'alphabet cyrillique ✱) l'alphabet latin avec € ❀) le format de CD ✲) l'alphabet latin sans €		○		
En Java, l'instruction wait ✾) bloque les méthodes ✱) attend une notification ❀) signale une anomalie ✲) bloque les méthodes synchronisées		○		
La méthode "notifyall" indique au moniteur ✾) de réactiver les tâches en attentes ✱) d'annuler la synchronisation ❀) de libèrer les sémaphores ✲) de bloquer les sémaphores	○			
Un périphérique est un ✾) logiciel ✱) contournement de bus de données ❀) composant matériel ✲) algorithme			○	
GPT est ✾) une structure de disque ✱) l'ordonnancement de tâche ❀) une architecture de processeur ✲) un ensemble de processus	○			
Sous Unix "kill" ✾) arrête un programme ✱) bloque les E/S ❀) libère la mémoire ✲) envoie un signal				○
Avec la commande ls -l le mode s désigne ✾) un périphérique spécial ✱) un socket ❀) un lien symbolique ✲) un pipe nommé		○		
Pour lancer une nouvelle tâche, il ne faut jamais ✾) créer un Thread ✱) afficher un objet Swing ❀) créer un Runnable ✲) invoquer la méthode run				○
Le nombre de processus actifs est —— nombre de processeurs. ✾) supérieur au ✱) indépendant du ❀) opposé au ✲) inférieur ou égal au				○
Un processus terminé dont le père n'a pas encore lu le testament se nomme : ✾) accidenté ✱) orphelin ❀) en attente ✲) zombie				○
Sous Unix, un numéro mineur désigne ✾) un périphérique particulier ✱) une version instable ❀) une expression irrégulière ✲) un développement négligeable	○			
Pour que les données échangées avec le disque soient régulières, la vitesse de rotation d'un disque est ✾) plus rapide au milieu ✱) constante ❀) plus rapide au centre ✲) plus rapide à l'extérieur		○		
Le modèle de financement par la publicité sur l'Internet ✾) enrichit les compagnies étasuniennes ✱) diminue les prix pays ❀) aide les pays pauvres ✲) donne l'accès à tous aux produits vitaux	○			
L'algorithme le plus utilisé intervenant dans les défauts de page est : ✾) LFU ✱) OPT ❀) FIFO ✲) aléatoire			○	
Le SFINX est ✾) un animal mythique ✱) le protocole d'échange égyptien ❀) un routeur aléatoire ✲) l'interconnexion des réseaux en France				○

Corrigé (Les réponses apparaissent avec ce fond de caractère)

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

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

2.) [8 pts] Résolution de problème sur les tâches (en java)

« Comme une boule de flipper »

Quoi de plus hypnotique que de regarder des auto tamponneuse s'entrechoquer. Nous nous proposons de créer des rectangle de couleur qui vont bouger dans un espace fermé. Attention, il ne s'agit pas de la même approche que le TP que vous avez rendu. Ici nous utiliseront directement des objets graphiques Swing.

Cette partie se décompose en plusieurs temps : compréhension du code fourni, proposition de solution, écriture de code. Les codes sont fourni sans nom de paquetage et l'on suppose qu'il y a eu les importations des classes standards nécessaire.

Les objets de couleurs sont autonomes. Ils se déplacent suivant une direction. En cas, de collision avec les murs, ils rebondissent. En cas de collision avec d'autres rectangles, ils attentent. Si une étreinte fatale (deadlock) est détectée, ils changent de direction.

Les mobiles

```
public class Mobile extends JLabel {
    Panneau panneau;
    Color color;
    int vx, vy;
    public Mobile (Panneau panneau, Color color,
                           int x, int y, int width, int height,
                           int vx, int vy) {
            this.panneau = panneau;
            this.color = color;
            setLocation (x, y);
            setSize (width, height);
            panneau.add (this);
            this.vx = vx;
            this.vy = vy;
            startSingle ();
    }
    public void flip () {
            vx *= -1;
            vy *= -1;
    }
    public void paint (Graphics g) {
            Graphics2D g2 = (Graphics2D) g;
            g2.setBackground (color);
            g2.clearRect (0, 0, getWidth (), getHeight ());
    }
    public static void pause (int millis) {
            try {
                Thread.sleep (millis);
            } catch (InterruptedException e) {
            }
    }
    public void startSingle () {
    }
}
```

Compréhension du code.

2.A) [½ pt] Donnez le rôle des attributs ? Quelles attributs indispensable à la représentation du mobile ne sont pas déclaré mais hérités ?

Corrigé (Les réponses apparaissent avec ce fond de caractère)

color est la couleur du rectanglevx et vy sont respectivement les vitesses de déplacement sur les axes horizontal et vertical.Les coordonnées x et y sont hérités de JPanel.width et height sont également hérité et représente la taille.

2.B) [½ pt] Donnez le rôle des méthodes ?

Mobile est le constructeur qui initialise les attributs et lance une activité (voir plus loin).flip change la direction de déplacement.paint déssine un rectangle couleur représentant le mobile.pause permet de suspendre l'activité pendant un temps en milliseconde passé en argument.

On suppose l'existence d'une méthode dans la classe Panneau qui prend de nouvelle coordonnées pour un mobile et si la place est libre, elle change les coordonnées.

```
public void tryMove (Mobile mobile, int x, int y) {
                    // il y a du code
                    mobile.setLocation (x, y);
                    // est la aussi
}
```

2.C) [1,5 pt] Ecrire la méthode startSingle qui toutes les 10 millisecondes, déplace le mobile. En cas de sortie de du panneau la vitesse est inversée dans le même axe.

```
    public void startSingle () {
            (new Thread () {
                    public void run () {
                        for (;;) {
                                pause (10);
                                int nx = getX ()+vx;
                                int ny = getY ()+vy;
                                if ((nx < 0 && vx < 0) ||
                                    (nx+getWidth () > panneau.getWidth () && vx > 0)) {
                                    vx = -vx;
                                    continue;
                                }
                                if ((ny < 0 && vy < 0) ||
                                    (ny+getHeight () > panneau.getHeight () && vy > 0)) {
                                    vy = -vy;
                                    continue;
                                }
                                panneau.tryMove (Mobile.this, getX ()+vx, getY ()+vy);
                            }
                        }
                }).start ();
    }
}
```

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

2.D) [½ pt] Quelle instruction Java vous a permit de créer une activité propre au mobile dans la question précédente ?

new Thread () pour l'activité / start pour la démarer / run pour le comportement à dérouler.

2.E) [½ pt] Pourquoi un mobile ne peut pas hérité d'une Java dans cette situation ?

Car c'est déjà un objet qui hérite de JPanel et qu'il n'y a pas d'héritage multiple en Java.

Nous passons à la seconde classe du problème : le panneau

```
public class Panneau extends JPanel {

    public static int PW = 400, PH = 200;
    public static int MW =  50, MH =  10;

    public static void main (String[] arg) {
            Panneau panneau = new Panneau (PW, PH);
            testComponent ("Panneau", panneau);

            new Mobile (panneau, Color.BLUE,          10,     PH/2, MW, MH, -1,  1);
            new Mobile (panneau, Color.GREEN,       PW/2,       10, MW, MH,  3,  1);
            new Mobile (panneau, Color.RED,            0,     PH/4, MW, MH,  1,  0);
            new Mobile (panneau, Color.MAGENTA,     PW/4,       10, MW, MH,  0, -2);
            new Mobile (panneau, Color.RED,            0, (PH*3)/4, MW, MH,  2,  0);
            new Mobile (panneau, Color.MAGENTA, (PW*3)/4,       10, MW, MH,  0, -1);
    }

    public Panneau (int width, int height) {
            super (null);
            Dimension size = new Dimension (width, height);
            setMinimumSize (size);
            setPreferredSize (size);
    }

    private Hashtable<Component, ArrayList<Component>> allCollisions =
            new Hashtable<Component, ArrayList<Component>> ();
    public ArrayList<Component> collisions (Component mobile, int x, int y) {
            ArrayList<Component> result = new ArrayList<Component> ();
            Rectangle newRect = new Rectangle (x, y, mobile.getWidth (), mobile.getHeight ());
            for (Component component : getComponents ()) {
                if (component == mobile)
                        continue;
                if (component.getBounds ().intersects (newRect))
                        result.add (component);
            }
            return result;
    }

    public enum CollisionType { No, Collision, DeadLock; }
    public CollisionType mustWait (Component mobile, int x, int y) {
            ArrayList<Component> collisions = collisions (mobile, x, y);
```

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

```
        if (collisions.size () == 0) {
            allCollisions.remove (mobile);
            return CollisionType.No;
        }
        allCollisions.put (mobile, collisions);
        for (Component component : getComponents ()) {
            ArrayList<Component> constraints = allCollisions.get (component);
            if (constraints != null && constraints.contains (mobile))
                    return CollisionType.DeadLock;
        }
        return CollisionType.Collision;
    }
```

2.F) [1 pt] Quel est le rôle de la méthode collisions ? Même question pour la méthode mustWait ?

Collision vérifie qu'il n'y a pas de superposition entre la future position du mobile et les autres objets graphics dans le JPanel

mustWait indique si un objet n'a aucune collision (No). Dans le cas contraire, on vérifie qu'aucun autre objet ne dépendrait de ce blocage se qui impliquerait une étreinte fatale.

La méthode tryMove réalise le comportement suivant :

- s'il y a une étreinte fatale la méthode change le sens de déplacement du mobile puis retourne sans rien faire.
- si il y a un risque de collision avec un autre mobile, la méthode attend puis réessaye le déplacement
- sinon, elle déplace le mobile

2.G) [1,5 pt] Ecrire la méthode tryMove.

```
    public synchronized void tryMove (Mobile mobile, int x, int y) {
            for (;;) {
                switch (mustWait (mobile, x, y)) {
                case No:
                        notifyAll ();
                        invalidate ();
                        mobile.setLocation (x, y);
                        validate ();
                        return;
                case Collision:
                        try {
                            wait ();
                        } catch (InterruptedException e) {
                        }
                        break;
                case DeadLock:
                        mobile.flip ();
                        return;
                }
            }
    }
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.H) [½ pt] Par quel mécanisme Java le déplacement d'un mobile est-il interrompu ? Indiquez les mots clefs et leur place dans tryMove

C'est l'instruction wait () qui bloque l'activité dans le "case collision".

2.I) [½ pt] Comment un mobile peut-il repartir ? Indiquez les mots clefs et le code correspondant dans tryMove.

C'est l'instruction notifyAll () qui permet de relancer l'évaluation de toute les conditions de blocage.

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

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

3.) [8 pts] Expressions régulières

Nous nous proposons d'analyser les lignes produites en interrogeant la commande netstat. Elles sont constituées par les informations suivantes : protocole, trames reçues, trames envoyées, adresse locale, adresse distante, état de la connexion. Voici un exemple d'exécution.

```
tcp4  0  0  15.15.158.193.53943  serv3.univ-ubs.imaps  ESTABLISHED
tcp4  0  0  15.15.158.193.43944  serv3.univ-ubs.imaps  ESTABLISHED
tcp4  0  295  15.15.158.193.53946  cache.google.com.https  ESTABLISHED
tcp4  0  0  15.15.158.193.29345  ax2-150-20-130-5.https  ESTABLISHED
tcp4  30  0  15.15.158.193.53945  ax2-150-20-110-6.https  ESTABLISHED
tcp4  0  0  15.15.158.193.30485  mail.monmail.net.imaps  ESTABLISHED
tcp4  0  0  15.15.158.193.53945  mail.monmail.net.imaps  ESTABLISHED
tcp4  255  0  15.15.158.193.34945  cache.google.com.https  ESTABLISHED
tcp4  0  148  15.15.158.193.39275  mail.monmail.net.imaps  ESTABLISHED
tcp4  0  0  15.15.158.193.53759  14.126.192.56.37562  ESTABLISHED
tcp4  0  0  15.15.158.193.59260  serv6.univ-ubs.imaps  ESTABLISHED
```

3.A) [4,5 pt] Proposer un algorithme permettant de vérifier en plusieurs expressions régulières la composition des lignes ci-dessus (une expression par élément de ligne).

```
String exprProt = "(\\w{4})\\s* '';
String exprTrReceived = "(.*)\\s*(\\d+)\\s*(.*)\\s* '';
String exprTrSent = "(.*)\\s*(\\d+)\\s*"(.*)\\s* '';
String exprIpLocal = "(.*)\\s*(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}:\\d{1,5})\\s*(.*) '';
String exprIdDistant = "\\s*(.*)\\s*";
String exprState= "(.*)\\s*(\\w{9})\\s*";
static public void parse (String expr, String test) {
            System.err.println ("expr: "+expr+" test :"+test);
            Pattern p = Pattern.compile (expr);
            Matcher m = p.matcher (test);
            boolean b = m.matches ();
            System.out.println ("matches: "+b);
}
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

3.B) [1 pt] Proposer une expression régulière permettant de vérifier la totalité de la structure des lignes (une expression pour la ligne).

```
String exprTotal=
"(\\w{4})\\s*(\\d+)\\s*(\\d+)\\s*(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}:\\d{1,5})\\s*(.*)"\\s*(\\w{9});
```

3.C) [2,5 pt] Proposer un algorithme avec une expression régulière permettant de vérifier la véracité d'un identifiant pour une connexion sur un site web. Les identifiants sont des adresses mails provenant uniquement de France, éventuellement commerciales, ne contenant que des lettres ou des chiffres, avec un maximum de 10 caractères en premières partie, et 5 caractères pour le nom de domaine..

```
String exprMail : ''(\\w{10})@(\\w{5}.(fr | com))'';
```