| Constantine, le 12/02/2020 | |
|----------------------------|--|
| Nom de l'étudiant(e) : | |
| Prénom: | |
| Groupe: | |

| 1-Quelles son | t les | différentes | méthodes | pour | implémenter | les | threads | en | java? | Justifier | quand | est | ce | qu'on |
|----------------|-------|-------------|----------|------|-------------|-----|---------|----|-------|-----------|-------|-----|----|-------|
| utilise chaque | méth | ode (2pts) | | | | | | | | | | | | |

| |
|------|
| |
| |
| |
| |

2- Pour chacune de ces questions, cocher les affirmations qui sont vraies (0,5 point par bonne réponse, -0,5 point par mauvaise réponse) (5 pts)

Pour chaque question, cocher les affirmations qui sont vraies (0,5 point par bonne réponse, -0,5 point par mauvaise réponse)

- 1- Quelle est la méthode qui permet de démarrer un thread :
- \square init();
- \square start();
- \square resume();
- \square run();
- 2- Pour assurer la synchronisation du point de rendez-vous suivant, déterminer les solutions correctes Pour Pt de rdv (init(SA,0), init (SB,0)):

 \prod

| Processus A | Processus B |
|-------------|-------------|
| ••• | ••• |
| I1; | J1; |
| Pt de rdv | Pt de rdv |
| I2 ; | J2; |
| ••• | ••• |

| Processus A | Processus B |
|-------------|-------------|
| ••• | ••• |
| I1 ; | J1; |
| I2 ; | J2; |
| ••• | ••• |

 Processus A
 Processus B

 I1;
 J1;

 V(SA);
 V(SB);

 P(SB);
 P(SA);

 I2;
 J2;

| Processus A | Processus B |
|-------------|-------------|
| I1 ; | J1; |
| P(SB); | V(SB); |
| V(SA); | P(SA); |
| I2 ; | J2; |

| Processus A | Processus B |
|-------------|-------------|
| I1 ; | J1; |
| P(SB); | P(SA); |
| V(SA); | V(SB); |
| I2; | J2; |

| Constantine, le 12/02/2020 | |
|----------------------------|--|
| Nom de l'étudiant(e) : | |
| Prénom: | |
| Groupe: | |
| | |

| 3- Qu'implémente la classe Semaphore : □ Object ; □ Thread ; □ Runnable ; □ Serializable ; | | | | | | | |
|---|--|--|--|--|--|--|--|
| 4- Parmi les propositions suivantes déterminer celles qui sont correctes : □ Le problème d'accès concurrent aux variables peut être résolu en déclarant ces synchronized ; □ Les méthodes synchronized sont remplacées par les méthodes lock(), unlock() etc. de | | | | | | | |
| Les methodes synchronized sont remplacees par les methodes lock(), unlock() etc. de l'interface Lock; Dans une classe ayant une partie du code synchronisé, plusieurs threads peuvent accéder à l'autre partie du code non synchronisé; Pour une barrière utilisée une seule fois, il vaut mieux utiliser CountDownLatch; notify() et notifyall() peuvent être appelées dans un contexte non-synchronisé; Les variables à conditions sont gérées par le paquetage java.util.concurrent.lock; La barrière est cyclique mais pas réutilisable; Les sémaphores en java sont gérées par les méthodes acquire() et signal() de la classe Semaphore; Une méthode statique ne peut pas être synchronisée; Une barrière permet de bloquer plusieurs thread en un point jusqu'à ce qu'un nombre prédéfini de threads atteigne ce point; Une barrière permet de bloquer un thread en un point jusqu'à ce qu'un nombre prédéfini de threads atteigne ce point; Quand le thread est endormi, il libère ses verrous; | | | | | | | |
| <pre>5- Que va afficher ce code ? class Test extends Thread{ public void run() { for(int i = 0; i < 30; i++) System.out.println("LMD GL"); } public static void main(String[] args) { Test t = new Test(); } }</pre> | | Rien du tout. 30 fois "LMD GL". Ce code ne compilera pas ! 29 fois "LMD GL | | | | | |

- 6- Quand un thread est-il considéré comme mort ?
- a. Lorsqu'il est mis en attente.
- b. Lorsqu'il est bloqué par un autre thread.
- c. Lorsqu'il a dépilé la méthode run() de sa pile d'exécution.

| Constantine, le 12/02/2020 | | | |
|----------------------------|--|--|--|
| Nom de l'étudiant(e): | | | |
| Prénom: | | | |
| Groupe: | | | |

Exercice 1 : Les Sémaphores (6pts)

Soient deux villes VILLE1 et VILLE2 sont reliées **par une seule voie** de chemin de fer. Les trains peuvent circuler dans **un seul sens à la fois** de la VILLE1 vers la VILLE2 ou de la VILLE2 vers la VILLE1. On considère deux classes de processus : les trains allant de la VILLE1 vers la VILLE2 (Train V1V2) et les trains allant de la VILLE2 vers la VILLE1 (Train V2V1). Ces processus se décrivent comme suit :

```
Train V1V2 {

Demande d'accès à la voie par VILLE1 () ;

Circulation sur la voie de la ville1 vers

la ville 2 () ;

Sortie de la voie par la VILLE2 ();

}
```

```
Train V2V1{

Demande d'accès à la voie par VILLE2 () ;

Circulation sur la voie de ville2 vers la ville1 () ;

Sortie de la voie par VILLE1 ();

}
```

1- Proposer une solution permettant de respecter les règles de circulation sur la voie unique en utilisant les sémaphores, utilisez deux compteurs pour calculer le nombre des trains voulant aller de V1V2 et de V2V1. (6pts)

| /* Sémaphores et initialisation | Demande d'accès à la voie par la VILLE2 () { |
|--|--|
| • | |
| | |
| Demande d'accès à la voie par la VILLE1 () { | |
| | |
| | |
| | } |
| | Sortie de la voie par la VILLE1 (){ |
| } | |
| Sortie de la voie par la VILLE 2 (){ | |
| | |
| | |
| | |
| | } |
| | |
| | |

| | Constantine, le 12/02/2020 |
|---|----------------------------|
| | Nom de l'étudiant(e): |
| | Prénom: |
| | Groupe: |
| _ | |

Exercice 2 : Les moniteurs en java (7pts)

Soit le problème du producteur/consommateur abordé au cours, avec un tampon de taille fixe N et circulaire, deux pointeurs : **queue** pour y produire et **tete** pour y consommer.

-Proposer le <u>code java</u> des méthodes : public void produire (int m) et public int consommer () en utilisant les Moniteurs de deux façons différentes.

| Moniteur | |
|------------|---------------|
| java < 1.5 | |
| Ü | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| Moniteur | |
| java > 1.5 | |
| juvu 110 | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | DON COUD A CE |
| | BON COURAGE |