STI 4e année – POO avancée: Java

Examen final

Salwa SOUAF

Nom et Prénom : _____ Date : _____

1 QCM (Entourez la bonne réponse)

Question 1 Quelle affirmation est vraie à propos de Java ?

A. C'est un langage de programmation dépendant de la plateforme
B. C'est un langage de programmation dépendant du code
(C) C'est un langage de programmation indépendant de la plate-forme
D. C'est un langage de programmation dépendant de la séquence

Question 2 Quel composant est utilié pour compiler, déboguer et exécuter un programme java ?

A. JSE
B. JVM
(C) JDK
D. JRE

Question 3 Quel composant est responsable de la conversion du byte-code en code spécifique à la machine ?

A. JSE
(B) JVM
C. JDK
D. JRE

Question 4 Lequel des mots clés non valide avec la méthode principale main() ?

A. public
B. static
(C) private
D. final

Question 5 Prenant en considération cet extrait de code, quelle ligne de code convient pour démarrer un thread ?

```
class MyRunnable implements Runnable
{
    public static void main(String args[])
    {
        /* Some code */
    }
    public void run() {}
}
```

A. Thread t = new Thread(MyRunnable);
B. Thread t = new Thread(MyRunnable); t.start();
(C) MyRunnable run = new MyRunnable(); Thread t = new Thread(run); t.start();
D. Thread t = new Thread(); MyRunnable.run();

2 Questions de cours

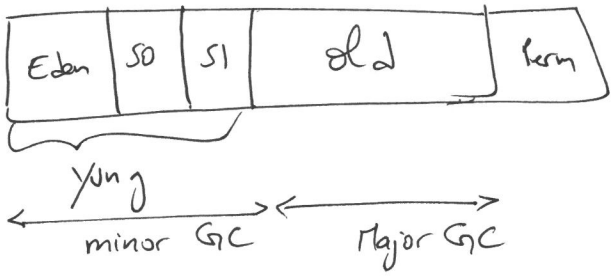
Question 6 Expliquez le principe de modularisation et ses apports

... La modularisation de la JVM :
- Classpath sous forme d'arbre de dépendances
- Vérification de la présence de tous modules nécessaires
- renforcement de la sécurité (package exportés ...)
- JVM modulaire : rt.jar découpé en modules

...

Question 7 Expliquez le fonctionnement du garbage collector GC1

...

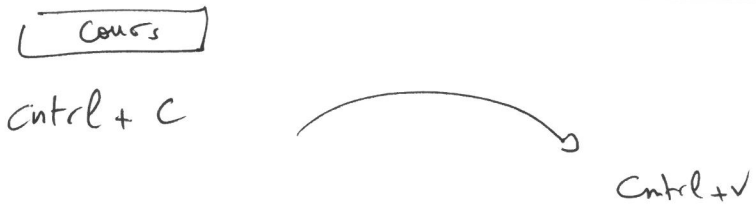


... etc

...

Question 8 Donnez la différence entre les processus lourds et légers

...



...

Question 9 Définir un Scheduler, mentionnez les catégories de Schedulers et donnez deux exemples d'algorithmes utilisés dans chaque catégorie.

...

Schedulers coopératifs
- FCFS
- Shortest Job Next
- Priority scheduling

Schedulers préemptifs
- Round - Robin
- Shortest Remaining Time

...

Question 10 Donnez la différence entre les deux méthodes : wait() et sleep()

...

sleep() : met le thread en pause pendant la durée précisée

wait() : thread en attente d'être redémarrer par un autre

Major difference : wait() relache les monitors
sleep() ne relache pas les monitors

...

3 Exercices

Exercice 11 Choisissez la pile d'opérandes équivalente au code source suivant, en justifiant votre choix.

```
int test() {
      int a= 1; byte c= 2; byte d= 3; int b= 4;
      int f= (a+b);
      return f;
   }
```

A.

```
iconst_1
istore_1
iconst_2
istore_2
iconst_3
istore_3
iconst_4
istore_4
iload_1
iload_4
iadd
i2b
istore_5
iload_5
ireturn
```

B.

```
iconst_1
istore_1
iconst_2
istore_2
iconst_3
istore_3
iconst_4
istore_4
iload_1
iload_4
iadd
istore_5
iload_5
ireturn
```

C.

```
iconst_1
istore_1
iconst_2
istore_2
iconst_3
istore_3
iconst_4
istore_4
iload_2
iload_3
iadd
istore_5
iload_5
ireturn
```

...

i2b : no need for conversion

...

Exercice 12

i - Donnez la sortie du programme (page suivante)
ii - Que modifier pour démarrer les Threads correctement?
iii - Donnez la sortie du programme une fois cette modification faite.

```
public class Test {
    public static void main(String[] args)
    {
  final StringBuffer a = new StringBuffer();
  final StringBuffer b = new StringBuffer();

          new Thread()
          {
              public void run() {
                try {
                synchronized(a) {
                        sleep(1000);
                        System.out.print(a.append("A"));
                        synchronized(b){
                          sleep(3000);
                          System.out.print(b.append("B"));
                       }
                }
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                                              }
                }
          }.run();

          new Thread()
          {
              public void run() {
                try {
                synchronized(b){
                  sleep(3000);
                  System.out.print(b.append("C"));
                  synchronized(a){
                    System.out.print(a.append("D"));
                                          }
                  }
                } catch (InterruptedException e) {
                        e.printStackTrace();
                                              }
                }
          }.run();
    }
}
```

i - avec run() on exécute les méthodes mais on lance pas les threads

Sortie : AB BC AD

ii -

change run() to start()

iii - AC => then deadlock due to the synchronized blocks and the sleep times

...

Exercice 13 Prenez en considération les deux classes suivantes. Donnez le résultat de l'exécution de main en justifiant votre réponse.

```
public class Mother {
  private String MyFunction() {
    String r = "I am " + this.getClass().getName() + " and this is my function !";
    return r;
  }
}
public class Daughter extends Mother {
  @Override
  String MyFunction() {
    String newR = "I am " + this.getClass().getName() + " and this is My function now !";
    return newR;
  }
}
public static void main(String [] args)
    {
        Daughter d = new Daughter();
        System.out.println(d.MyFunction());
        Daughter d2 = new Daughter();
        Mother m = (Mother) d2;
        System.out.println(m.MyFunction());
    }
```

...

Prop de compilation : on ne peut appeler une méthode privée

...

Exercice 14 On veut implémenter une file d'attente d'accès à un service, pour limiter le nombre d'utilisateurs du service (i.e. à tout moment pas plus de **n** utilisateurs peuvent manipuler le service). Expliquez comment cela peut être fait, en détaillant comment vous vous y prenez lors du codage. Vous pouvez inclure des extraits de code si vous le souhaitez.

...

Utilisation de Sémaphore avec n jeton

...

...

4 Bonus

Exercice 15 Donnez la sortie du programme suivant :

```
public class Bonus
{
    public static void main(String[] args)
    {
        System.out.println(2+0+1+9+"/"+1+1+"/"+1+2);
        System.out.println((2+0+1+9)+"/"+1+1+"/"+1+2);
        System.out.println("Today "+2+0+1+9+"/"+1+1+"/"+(1+2));
    }
}
```

```
...
    12 / 11 / 12
    12 / 11 / 12
    Today 2019 / 11 / 3
...
```