

Instructions

- **Durée** : 2 heures
- **Note totale / maximale** : 20 points
- **Des points bonus peuvent être cachés !**
- **Les réponses doivent être argumentées, sinon 0**
- **Le code doit compiler (Java 11+) sinon note / 2**
- **Attention aux pièges**

Q1 Programme et processus (2 pt)

Définissez précisément la différence entre un **programme** et un **processus**.
Donnez les différences clés (format/stockage, espace mémoire, exécution, identifiants).

Q2 Commutation de contexte (2 pts)

Qu'est-ce qu'un **context switch** (commutation de contexte) ? Listez les éléments essentiels sauvegardés/restaurés et citez un **coût direct** et un **coût indirect**.

Q3 Copier un fichier (2 pts)

Écrivez une méthode Java qui copie un fichier source vers une destination en gérant correctement la fermeture des ressources, même en cas d'exception. Utilisez `FileInputStream` et `FileOutputStream`. Signature attendue :

```
1 public static void copyFile(Path src, Path dst) throws IOException
```

Q4 Lecture de fichier (4 pts)

On reçoit un flux texte au format PPM (P3) avec des commentaires et des espaces multiples :

```
1 P3
2 # bla bla bla
3 # 6 2 mauvaise dimension
4 3 1
5 # valeur max
6 255
7 255 0 0    0 255 0    0 0 255
```

- Donnez une stratégie robuste pour ignorer les commentaires `#` et les espaces multiples (utilisez `BufferedReader` + `StreamTokenizer` ou regex).
- Quelle structure de données (types Java) utiliseriez-vous pour stocker l'image en mémoire pour un traitement ultérieur (ex. : rotation, filtre) ? Justifiez (performance, accès, taille).
- Implémentez la méthode de lecture avec gestion d'erreurs (format invalide, EOF prématuré).

Signature attendue :

```
1 public static int[][][] readP3(String filename) throws IOException
```

Retourne un tableau 3D : $height \times width \times 3$ (R,G,B).

Q5 Concurrency (3 pts)

Le code suivant incrémente un compteur partagé :

```
1 class Counter {
2     int value = 0;
3 }
4
5 Counter c = new Counter();
6 Thread t1 = new Thread(() -> {
7     for (int i = 0; i < 1_000_000; i++)
8         c.value++;
9 });
10 Thread t2 = new Thread(() -> {
11     for (int i = 0; i < 1_000_000; i++)
12         c.value++;
13 });
14 t1.start(); t2.start();
15 t1.join(); t2.join();
16 System.out.println(c.value); // Attendu : 2_000_000
```

- Quels problèmes potentiels ? Expliquez avec un scénario de *race condition*.
- Proposez une correction (sans changer la sémantique)
- Attention on veut que les boucles s'exécutent **en parallèle**.

Q6 Sémaphore (2 pts)

- Différenciez **sémaphore binaire**, **mutex** et **sémaphore comptant**.
- Donnez un exemple d'usage pour limiter à 3 threads l'accès à une ressource partagée (ex. : connexion DB).
- Écrivez un extrait Java complet avec **Semaphore** (acquisition/libération).

Q7 Adressage (3 pts)

Expliquez le mécanisme d'**adressage des données** d'un *inode* (pointeurs directs et indirects). Avec une taille de bloc de **4 Ko** et des pointeurs de **4 octets** :

- Calculez la capacité couverte par les **12 pointeurs directs**.
- Calculez la capacité couverte par l'**indirection simple** (1 niveau).
- Ignorez double/triple indirection.

Q8 Liens (2 pt)

Faites la différence entre **lien dur** et **lien symbolique**.

Donnez un risque d'utilisation pour chacun (ex. : suppression, déplacement).