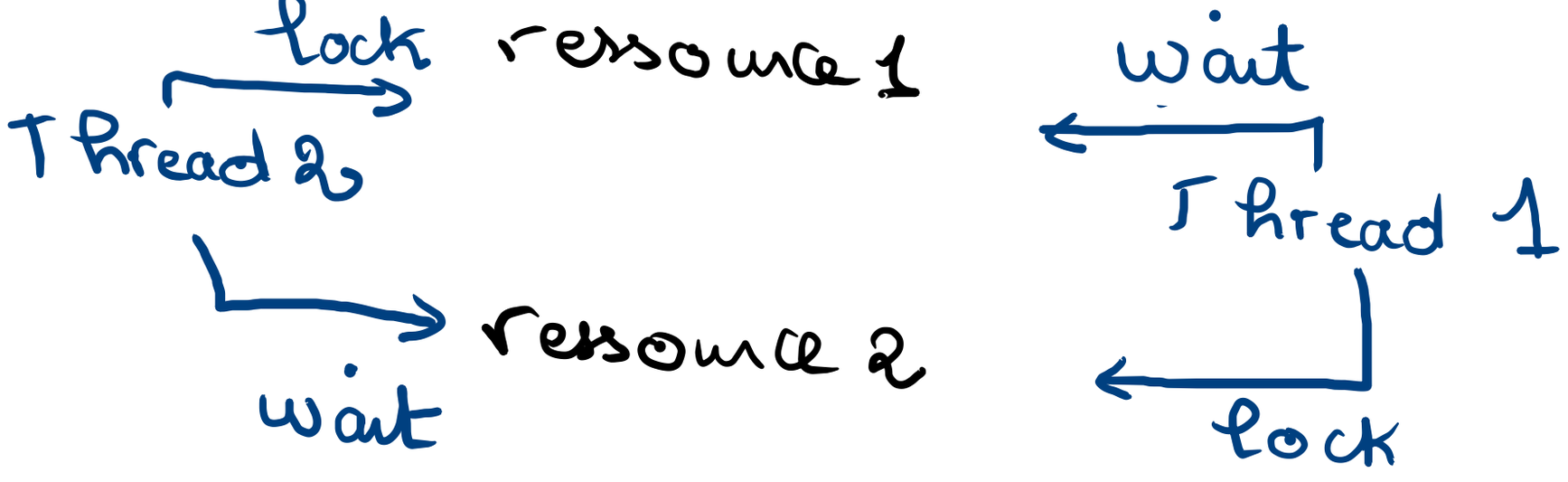
Thread = Une tâche simultanément

- in ressource
- Deadlock (Scénario)



- Solution : - timeout : Deadline
  forcer le thread à enlever lock

Thread en Java

① « classe » Thread

② « Interface » Runnable

→ Redéfinir la méthode run() = la description des tâches effectuées.

Exemple:

```
Main : {
  for(int i = 0; i<10; i++) {
    Thread t = new Thread(){
      Public void run(){
        SOP("Hello Word");}}
    t.start();
  }}
```

// affichage:
Hello Word
(10 fois)

Thread en Java

① « classe » Thread

② « Interface » Runnable

→ Redéfinir la méthode run() = la description des tâches effectuées.

Exemple:

```
Main : {
  for(int i = 0; i<10; i++) {
    Thread t = new Thread(){
      Public void run(){
        SOP("Hello Word");}}
    t.start(); } }
```

// affichage:
Hello Word
(10 fois)

① Déclarer Thread (class, Interface)

② redéfinir la methode run

③ Thread.start() // Lancer le thread.

Thread 1
retrait (somme)

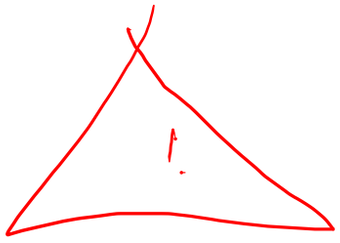
Thread 2
retrait (somme)

Synchronized: pour organiser accès à montant (retrait)

⚠ Ordonnanceur Java qui choisit l'ordre d'exécution

=> Code sur Eclipse.

1. 1000 - 250 ≡ t2 puis t1
2. 1000 - 200 ≡ t1 puis t2



Pour chaque tâches il faut :

1. Créer -> instantiéi new check();
2. affecter les queues.
3. Créer le thread qui lui correspond

en = Thread t = new Thread(check);

4. t.start();

attention ~~check.run();~~