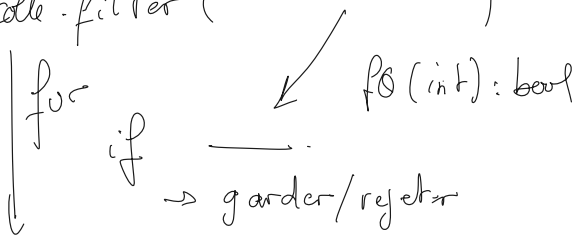
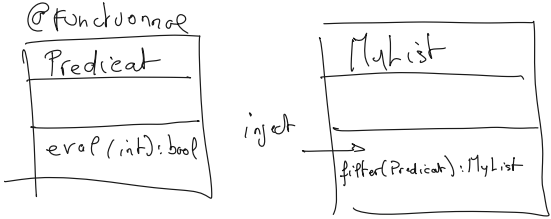
Exercice :

- Faire une classe qui stocke un Tableau de int
- Cette classe implémente une méthode Filter qui prends un prédicat fonctionnel :



qui renvoie un autre tableau filtré

ex : tab.filter(i → i > 0);



MyList mL1 = ...
MyList mL2 = mL1.filter(i -> i>0);

---

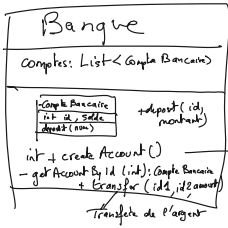
fabriquer un Objet → marquer
ex → data, couleur, n°, Set is ...

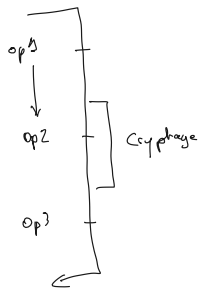
Objet → nom → ce que vous voulez
. marque

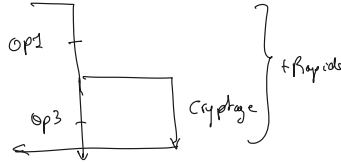
fabrique → Supplier
ex: Supplier sup = () -> new O();

Consumer cs = (u) -> u.setMarque(new Date...());

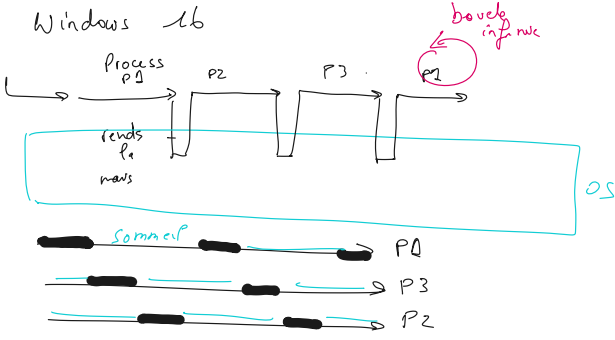
classes internes

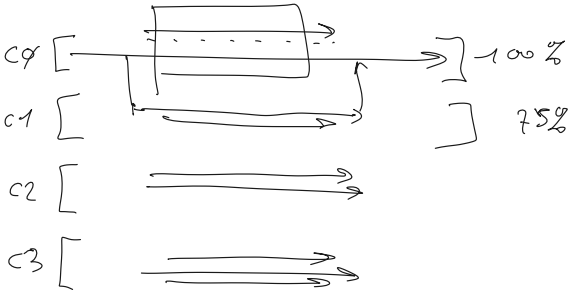






Windows 16





Priorité de Process

Windows

niveaux
- CRITICAL
- HIGHEST
- HIGH
- Above normal
- NORMAL
- below normal
- Low
- Lower
- IDLE

⇒ exécute niveau le + élevé, si cycles CPU relachés, on passe un niveau en dessous.

Linux

1 process = 1 nombre
par défaut = 100

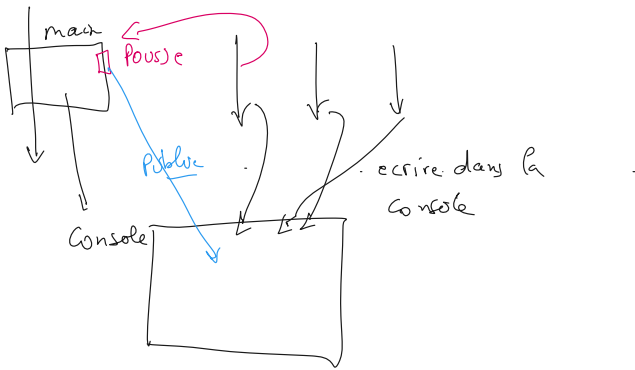
On découpe les allocations par niveau du Process / 5 des niveaux

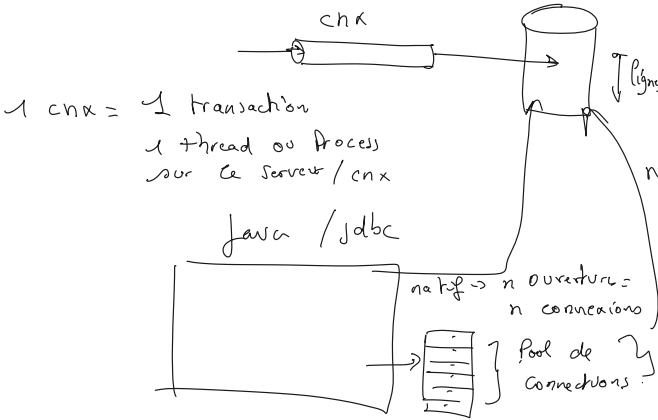
Dans un Process, chaque thread est

- | fore ground
  Back

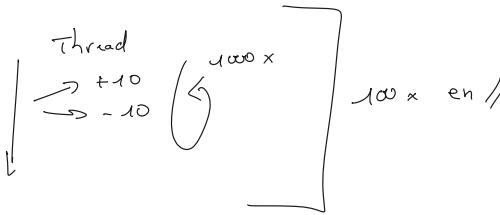
→ le process sort quand il n'y a plus de thread foreground

- peuvent avoir des priorités ≠




cnx
1 cnx = 1 transaction
1 thread ou Process
sur ce Serveur / cnx
lignes
n
Java / Jdbc
natif → n ouverture =
n connexions
Pool de
Connexions

Solde ⌀

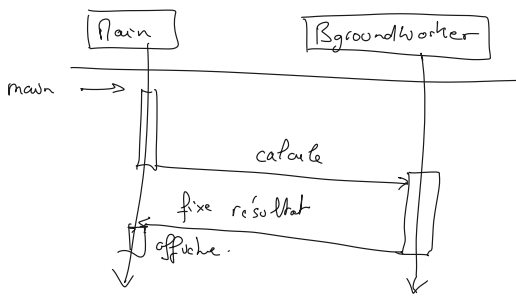


(a) → implémenter et lancer sans synchro
→ résultat = ?

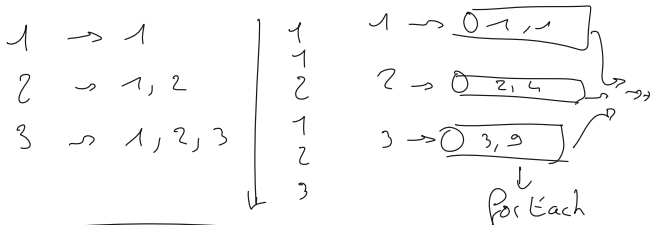
(b) → Synchroniser au niveau méthode

(c) → " ou plus fin. est-ce + rapide que (1)?

Exercice : Depuis le Thread principal, demander le calcul d'une factorielle en Background, et afficher le résultat



flux 1 → n



---

map → générer des flux de flux (boucle)

flatMap → plus simple pour mettre à plat un flux de flux et n'avoir qu'un flux à traiter