Réalisé par:

GHOFRANE BOUALLEGUE - FIE4

Question 2 : Pourquoi l'essuyeur ne peut pas prendre d'assiette dans la pile ?

- La méthode `push` n'était pas synchronisée et il n'y avait pas de contrôle de pile pleine.
- Donc si la pile est vide au moment où l'essuyeur fait `pop()`, il se met en attente (`wait()`) et ne peut rien retirer tant qu'un laveur n'a pas ajouté d'assiette.

Partie 2: Table des philosophes

Question 1 : Comment les accès concurrents sont gérés?

- Chaque baguette est un objet avec des méthodes `take()` et `release()` synchronisées.
- **`synchronized`** garantit qu'un seul philosophe peut prendre ou relâcher une baguette à la fois.
- **`wait()`** et **`notifyAll()`** gèrent l'attente : si la baguette est occupée, le philosophe attend (`wait()`) et dès qu'elle est relâchée, `notifyAll()` réveille les philosophes en attente.
- Donc les accès concurrents sont sécurisés au niveau des baguettes.

Question 2 : Est-ce un deadlock ou une famine ? Pourquoi ce blocage apparaît-il ?

- Le programme peut se **bloquer complètement**.
- Il s'agit d'un **deadlock** pas d'une famine.
- **Pourquoi ?**
    - Chaque philosophe prend d'abord une baguette (gauche ou droite) puis essaie de prendre l'autre.
    - Si tous prennent leur première baguette en même temps, chacun attend la baguette de l'autre : aucun ne peut continuer -> interblocage.
    - C'est un **deadlock classique du problème des philosophes**.