# Exercice : Le problème des Philosophes

## Introduction

Le problème des philosophes a été introduit par Dijkstra pour illustrer les problèmes de synchronisation et d'accès aux ressources partagées en programmation concurrente.

Cinq philosophes sont assis autour d'une table ronde. Entre chaque paire de philosophes se trouve une fourchette. Un philosophe alterne entre deux états :

- Penser
- Manger

Pour manger, un philosophe doit posséder les deux fourchettes situées à sa gauche et à sa droite.

On souhaite éviter :

- L'interblocage (deadlock)
- La famine (starvation)

# 1 Version 1 : Solution centralisée avec superviseur

Dans cette version, un processus supplémentaire appelé **Superviseur** contrôle l'accès aux fourchettes.

## Principe

- Un philosophe doit demander l'autorisation au superviseur avant de manger.
- Le superviseur vérifie si les deux fourchettes sont disponibles.
- Si oui, il accorde l'autorisation.
- Sinon, le philosophe attend.
- Après avoir mangé, le philosophe informe le superviseur qu'il libère les fourchettes.

## Travail demandé

1. Définir les variables d'état du superviseur.
2. Proposer un algorithme en pseudo-code pour le superviseur.
3. Expliquer pourquoi cette solution évite l'interblocage.
4. Cette solution garantit-elle l'absence de famine ? Justifier.

# 2 Version 2 : Solution distribuée sans superviseur

Dans cette version, il n'y a pas de processus central. Chaque philosophe communique uniquement avec ses voisins.

## Principe

- Chaque fourchette est représentée par un processus.
- Un philosophe doit demander la fourchette gauche et la fourchette droite.
- Une fourchette ne peut être utilisée que par un seul philosophe à la fois.

Une stratégie possible :

- Un philosophe prend d'abord la fourchette de numéro le plus petit,
- Puis la fourchette de numéro le plus grand.

## Travail demandé

1. Implémenter en Akka et en Scala la version centralisée et la version distribuée.