UCLouvain

École polytechnique de Louvain

LINFO1104

Concepts, paradigmes et sémantique des langages de programmation

Auteurs :
Nicolas Jeanmenne
Tania Shafiei

Formation académique :
Bachelier en sciences informatiques

Année académique :
Bac 2

2021-2022

Préface

Cette synthèse ne remplace pas le cours et ne constitue en aucun cas un substitut des CM ou des TP. Ce n'est qu'une version synthétique et vulgarisée du cours de *Concepts, paradigmes et sémantique des langages de programmation* réalisé par des étudiants pour des étudiants.

Il va de soi, que cette synthèse n'a pas été par un génie, par conséquent si une chose dite en cours et/ou aux TP, est contraire à ce que dit cette synthèse, **c'est l'explication du cours/TP qui prime**.

Cette synthèse est open source, le code LaTeX étant disponible sur GitHub. Les éventuelles contributions ou corrections peuvent donc être faite via *pull request* en suivant le lien ci-dessous.

https://github.com/Nicojmn/LINFO1104-synthese

Table des matières

Chapitre 1

Introduction

Contenu du chapitre

1.1 Qu'est-ce qu'un paradigme ?

Dans le monde du développement informatique, il y a des centaines de langages de programmation, maitriser tous ceux-ci est impossible en un cycle d'étude, voir en *une vie*. C'est pourquoi nous allons apprendre le concept de paradigme.

Un langage de programmation utilise **un ou plusieurs paradigmes**. Pour fournir un exemple, Java est un langage mono-paradigme qui utilise la programmation orienté objet et Python est multi-paradigmes (impératif, POO, ...). Nous pouvons résumer tout ceci sous forme de schéma :

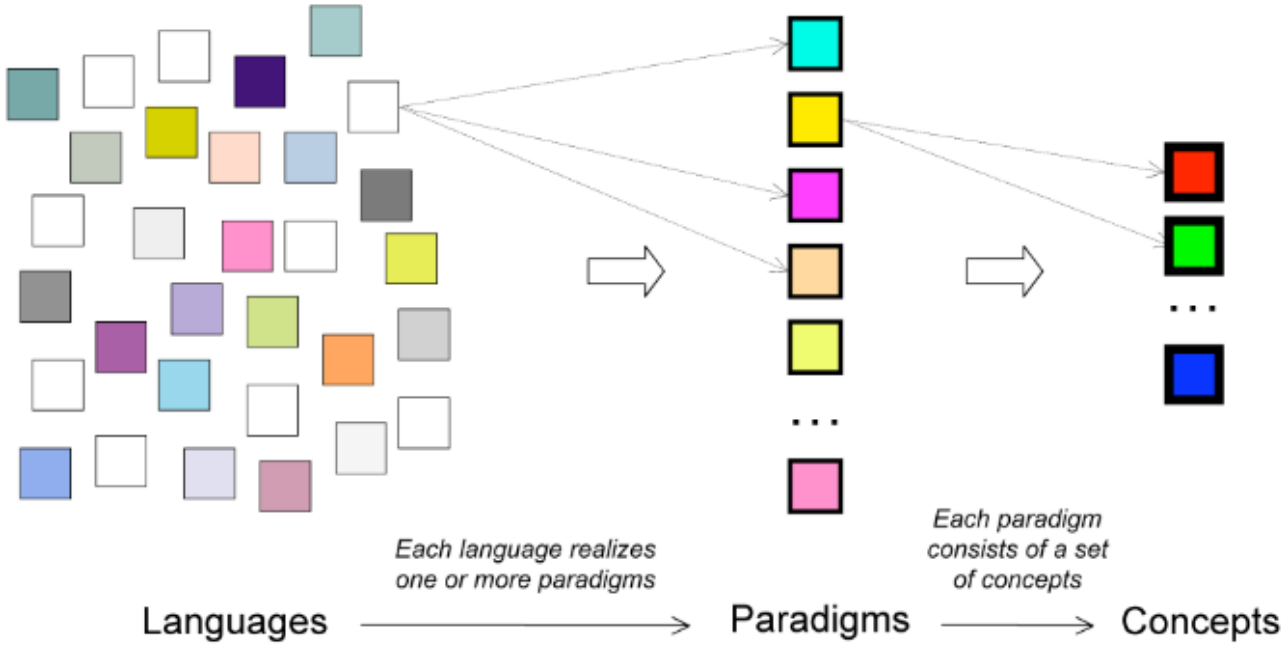

FIGURE 1.1 – Schéma du concept de paradigme

Dès lors, vu que **tous** les langages implémentent un ou des paradigmes, il suffit d'apprendre les principaux paradigmes pour avoir des connaisances de bases dans tous les langages existants et à venir.

Définition formelle d'un paradigme

Un paradigme de programmation est une approche de programmation d'ordinateur basée sur un ensemble cohérent de principes / règles ou des théories mathématiques.

Un programme est écrit pour résoudre des problèmes

- Un programme réaliste doit savoir résoudre différents types de problèmes
- Chaque type de problème a besoin de son paradigme propre
- On a donc besoin de plusieurs paradigmes à combiner dans un même programme

1.2 Étudier plusieurs paradigmes

1.2.1 Comment étudier plusieurs paradigmes ?

Comme la plupart des languages ne supportent qu'un paradigme et que nous devons apprendre plusieurs d'entre eux, nous allons utiliser le langage de recherche Oz qui combinent plusieurs paradigmes avec sa propre syntaxe et sémantique (un langage par paradigme étant beaucoup trop lourd pour un seul cours). Nous verrons aussi Erlang pour le paradigme *actor dataflow programming*.

1.2.2 Combiner créer un programme multi-paradigmes ?

- Chaque paradigme est une approche différente de pensée
- Un concept clé pour les implémenter est le langage noyau

Langage noyau

Le langage noyau est à la base de tous les paradigmes. Celui fonctionne d'une manière simple, il contient les concepts essentiels auquel chaque paradigme va ajouter ses propres concepts un par un. De cette façon il est aussi facile de traduire un langage compliqué en langage noyau. En plus, les langages noyaux des différents paradigmes ont souvent beaucoup de concepts communs (exemple : conditions structurelles if/else, récursion, ...).

1.3 Paradigmes principaux

Nous verrons dans ce cours les 5 paradigmes les plus importants. Il y existe bien sur plein d'autres pour d'autres types de problèmes (constraint programming, ...) que nous ne verrons pas ici.

Paradigmes principaux

- Functional programming
- Object-orientied programming (Java)
- Functional dataflow programming
- Actor dataflow programming (multi-agent) (Erlang)
- Actives objects

Ces paradigmes sont liés entre-eux comme le montre l'image ci-dessous (ils s'emboitent comme des legos)

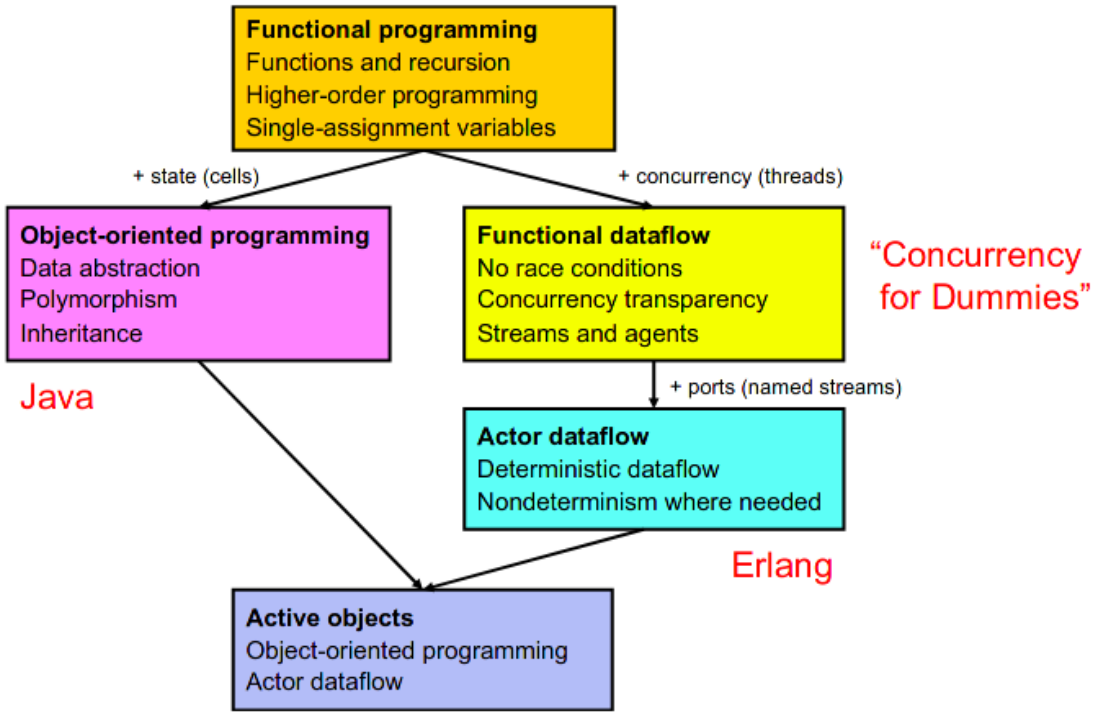

FIGURE 1.2 – Liens entre les paradigmes

1.4 Principes de base

Première partie

Functional programming

Chapitre 2

Introduction à la programmation fonctionnelle

Contenu du chapitre

2.1 Caractérisques

C'est le premier paradigme que nous allons voir. Voici ses caractéristiques :

- Paradigme le plus simple
- Fondement de tous les autres paradigmes
- C'est une forme de declarative programming : *say what, not how*

- Introduction pour les concepts de programmation
- Introduction au langage noyau
- Sert à expliquer l'invariant programming
- Sert à expliquer la symbolic programming
- Sert à expliquer la higher-order programming
- Sémantique formelle basée sur le langage noyau