

CHAPITRE 0 :

## *INTRODUCTION A LA POO*

### **Objectif spécifique**

➔ Rappel sur les types abstraits de données

### **Eléments de contenu**

**I.** Rappel : les types abstraits de données

**II.** Exemple : comment penser en OO

### **Volume Horaire :**

**Cours :** 1 heure 30

**TD :** 0 heure

## **0.1 Rappel : Les Types Abstraites de données**

### **0.1.1 Définition :**

Un type de données abstraites (TDA) peut être représenté comme l'abstraction sur les techniques de représentation des données : fonctionnalité (opérations) avant tout. On peut dire qu'un TDA est constitué d'un modèle mathématique et les opérations définies sur ce modèle.

Généralement, nous voulons construire des algorithmes en termes de TDA, mais la mise en œuvre d'un algorithme dans un langage de programmation nécessite que nous trouvions un moyen de représenter les TDA à l'aide des types de données et des opérations prédéfinis dans le langage lui-même. Pour représenter le modèle mathématique sous-jacent à un TDA : il faut recourir à des structures de données, qui sont des ensembles de variables, à priori de types différents, reliées de multiples façons.

L'incarnation d'un TDA est la traduction, dans les termes d'un langage de programmation, la déclaration qui définit une variable comme appartenant à ce type de données abstrait, et l'écriture dans un même langage d'une procédure pour chaque opération relative à un TDA.

Une incarnation nécessite le choix d'une structure de données pour représenter le TDA : chaque structure de donnée est construite à partir de types prédéfinis du langage de programmation hôte, à l'aide de ses possibilités de structuration des données.

Par exemple les tableaux (array) et les enregistrements (record) sont deux outils particulièrement importants en Pascal.

En réalité, tout informaticien utilise déjà les TADs, parfois sans le savoir. En effet, dans un algorithme il doit manipuler des nombres entiers ou réels, il ne se préoccupe pas de la représentation de ces nombres, mais uniquement des propriétés intrinsèques des quatre opérations habituelles (+, -, \*, /).

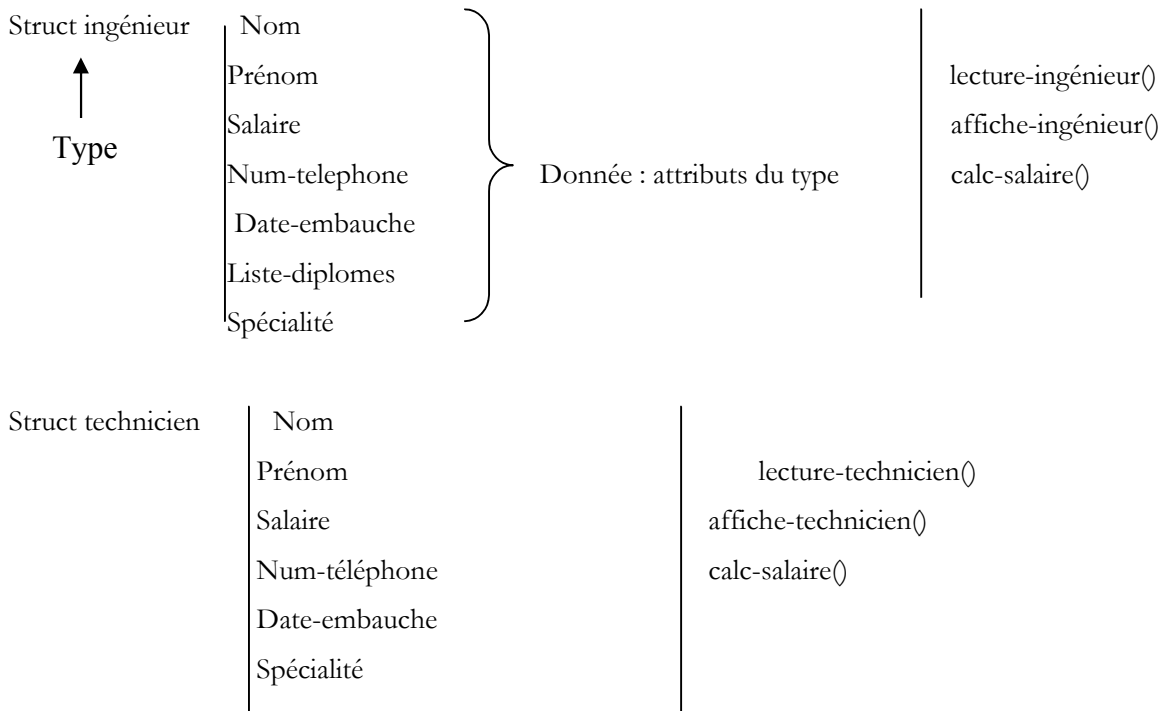
On présente ici par l'exemple la spécification d'un type bien connu, celui des booléens : le TDA booléen.

<b>type booléen</b>	
<b>opérations</b>	
vrai	: → booléen
faux	: → booléen
¬ _	: booléen → booléen
_ ∧ _	: booléen × booléen → booléen
_ ∨ _	: booléen × booléen → booléen
<b>sémantique</b>	
pour tout a, b : booléen	
¬vrai	= faux
¬¬a	= a
vrai ∧ a	= a
faux ∧ a	= faux
a ∨ b	= ¬(¬a ∧ ¬b)

## 0.2 Exemple : concept POO

**Consigne :** on veut faire un programme de gestion du personnel d'une entreprise informatique. On y trouve des ingénieurs, des techniciens, des directeurs et des ouvriers d'entretiens (concierge, femme de ménage...).

**1<sup>ère</sup> solution :** Programmation procédurale classique => Que veut on faire ?



Laboratoire

Struct directeur	Nom	lecture-directeur()
	Prénom	affiche-directeur()
	Salaire	calc-salaire()
	Num-téléphone	
	Num-fax	
	Num-bureau	
	Service	

**2<sup>ème</sup> solution :** Programmation orientée objet => De quoi parle-t-on ?

