Chapitre 1

Introduction à la Programmation Orientée Objets

On parle de *Programmation Orientée Objet*, ou plus simplement de *Programmation par Objets*. Cette manière de programmer consiste à définir ou utiliser des *objets informatiques*. Il existe de la même manière des objets mathématiques, ou des objets du quotidien ... Comme un objet du quotidien, un objet informatique est prévu pour un certain usage, et ne peut être manipulé qu'au travers des opérations prévues pour lui : avec une télé, on peut changer de chaîne, monter le volume, etc. mais on en peut pas laver du linge, ni téléphoner !

Pour se figurer ce qu'est un objet informatique, pensez à un bouton sur une interface graphique : son comportement est toujours le même quelque soit l'application qui l'utilise. Il a toujours la même forme, on peut cliquer dessus, il le signale par un changement de couleur, ... On peut paramétrer le texte qu'il contient, les actions qu'il déclenche, etc.

En tant que programmeurs, vous serez amenés à envisager les objets de deux façons :
- vous apprendrez à programmer de nouveaux objets informatiques,
- vous apprendrez aussi à programmer en utilisant des objets déjà existants.

Commençons par situer l'activité de programmation dans son ensemble, avant de situer la programmation par objets proprement dite.

1 L'activité de programmation d'une machine (d'un ordinateur)

1.1 Qu'est-ce que la programmation ?

Programmation. Programmer consiste à donner des instructions à un ordinateur pour qu'il calcule la solution d'un problème.

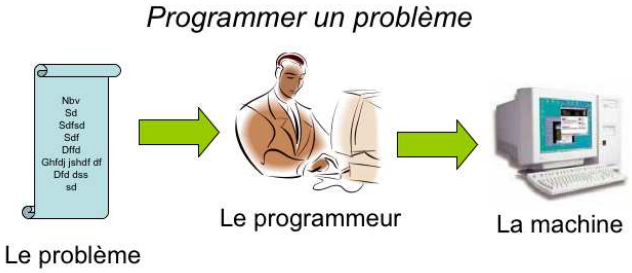

FIG. 1.1 – L'activité de programmation d'un problème

Il y'a d'une part le problème, d'autre part l'ordinateur. Entre les deux : le programmeur (voir Fig. 1.1).

Le programmeur imagine ou connaît une stratégie de résolution du problème, et la traduit en un programme (= une suite d'instructions) qui sera compris et exécuté par l'ordinateur.

Exemple. Un problème : déterminer si un nombre entier N est un nombre premier.

Une stratégie de résolution (pas la plus efficace) : essayer de diviser N successivement par tous les entiers strictement supérieurs à 1 et plus petits que lui. Dès qu'un de ces entiers divise N, on sait que N n'est pas premier. Par contre, si on « épuise » tous les entiers candidats sans en avoir trouvé un seul qui divisait N, alors on sait que N est premier.

On peut faire les calculs à la main, mais c'est vite fastidieux ! Autant programmer une machine pour le faire. Mais une machine ne comprend pas une langue naturelle (ici le Français) : il faut lui décrire la stratégie sous une forme plus « mécanique ».

Exemple de programme appliquant la stratégie (N est connu)

```
candidat <- 2
fini <- faux
TANT QUE (candidat < N et fini=FAUX) FAIRE
      resteDivision <- N / candidat
      SI resteDivision = 0 ALORS
            AFFICHER "N n'est pas premier"
            fini <- VRAI
      SINON
            candidat <- candidat + 1
      FSI
FTQ
SI (candidat = N) ALORS
      AFFICHER "N est premier"
FSI
```

La façon dont l'homme s'adresse à la machine pour lui donner ses instructions est passée de nombreuses étapes depuis l'invention de l'ordinateur. L'idéal est que cette forme soit la

plus intelligible possible pour l'esprit humain. La programmation par objets constitue l'une des étapes en direction de ce but.

1.2 Brefs repères pour l'évolution de l'activité de programmation

Au tout début ($\sim$ 1940), pré-ordinateurs = énormes machines électroniques. Programmer signifie "câbler".

Ensuite ($\sim$ 1944), programmation en binaire : le langage de la machine (que des 0 et des 1)

Puis ($\sim$ 1950), programmation en assembleur : on programme en utilisant les instructions connues du processeur (abstraction du langage machine)

Enfin ($\sim$ depuis 1955), programmation avec un langage impératif (C, Pascal, Fortran, ...) Ces langages sont plus lisibles par un humain, mais restent une abstraction du langage de la machine. L'exemple donné plus haut n'est pas codé dans un langage existant, mais ressemble fortement à un programme écrit en PASCAL ou en C.

La programmation par objets (apparue $\sim$ 1977) a pour point de départ deux constats :

- même avec un langage de haut niveau tel que C ou PASCAL, on continue à ne parler que le langage de la machine. L'être humain, lui, se satisferait plutôt de la description d'un problème mettant en jeu divers OBJETS, dont il connaît ou imagine le fonctionnement.
- il ne faut pas réinventer la roue à chaque fois. Une fois qu'une stratégie de résolution d'un problème a été programmée, on devrait pouvoir la réutiliser sans réécrire le programme à chaque fois qu'on veut résoudre le problème.

Ainsi, la Programmation Objets a pour volonté de décrire des entités facilement manipulables et réutilisables par d'autres programmeurs. Un langage objet offre des stratégies déjà programmées de résolution d'un grand nombre de problèmes.

Exemple. Un programmeur Objet confronté au problème de la primalité d'un nombre, aura envie d'écrire simplement

```
Si (Math.estUnNombrePremier(N)) ALORS
    AFFICHER "N est premier"
SINON
    AFFICHER "N n'est pas premier"
FSI
```

plutôt que de reprogrammer la stratégie vue plus haut.

La programmation objet correspond ainsi à une volonté de décrire le calcul d'une solution à un problème selon les termes même du problème. Chaque terme du problème devient un objet, et chaque objet contient du code qui lui permet d'être manipulé par la machine.

Exemples d'objets selon les problèmes

Jeu d'échec : les pièces, l'échiquier, ...

Recherche d'un chemin entre deux points (problème de graphe) : nœuds du graphe, arêtes, le graphe lui-même,

Problème mathématique : objet matrice, objet nombre complexe, ...

Problème de dessin : objets forme géométrique (cercle, rectangle, ...)

Ce cours détaillera comment :
– décrire de nouveaux objets au moyen d'une classe
– utiliser des objets déjà existants