Hello, Java

A l'ère des langages obscurs et inaccessibles pour le commun des programmeurs, le langage Basic remplissait son office : permettre l'écriture du code source des exemples de programmation. L'intérêt pédagogique était certain. Certes répandu, il était hélas peu portable d'une machine à l'autre. Puis s'est imposé le Pascal, langage didactique par excellence puisque conçu dans ce but. Mais hélas il a souffert d'un manque de standardisation. Alors qu'avons nous aujourd'hui à notre disposition comme langage d'illustration, d'exemple - pas celui qui va donner les meilleurs performances dans tel ou tel domaine sur telle ou telle machine. Pourquoi donc ne pas utiliser le langage Java pour écrire les codes sources des exemples de programmation ?

Bien que Java soit un nouveau concept de langage, on ne peut s'empêcher de le comparer au langage C.

1) Une première constatation concerne la disparition du macro-processeur. C'est une bonne nouvelle. Son utilisation, bien des fois incontournable, obscurcit dramatiquement la compréhension des sources C quand il est utilisé à outrance. Dans le même genre de bonne nouvelle, il n'y a plus besoin des fichiers interfaces ".h" qui fleurissaient à profusion. Le nombre indéfini d'arguments des fonctions a aussi disparut. Les pointeurs se font discrets dans le sens où leur utilisation est implicite. On n'a plus le droit de faire de l'arithmétique absconse avec.

2) La deuxième constatation concerne les structures de contrôle (if ... else, switch, while, for). Elles gardent leur similitude avec le C. Dommage, comme le Pascal, le C, Java ne possède donc pas de mot clé de terminaison des structures de contrôle, comme on peut en trouver en Basic ou en Ada (end if, end case, end loop). Celles ci sécurisent les portions de code sur un bloc d'instructions ou une seule instruction. Dans le cas contraire, il est facile d'indenter rapidement trois lignes après un if en oubliant les accolades. Seule la première ligne sera conditionnée par le if,

malheureusement pas les deux autres comme souhaité. Ce genre d'erreur n'est pas facilement repérable d'un coup d'oeil sur le source.
Autre erreur fréquente et difficile à repérer est la confusion des opérateurs = et ==. A noter que Java garde la notation sybilline des opérateurs et, ou : &, |. La conservation de la distinction majuscule / minuscule contribue toujours à une certaine confusion. D'autant plus que le langage encourage à avoir des noms identiques différenciés uniquement par la casse des lettres.

3) La troisième constatation concerne les nouveautés de Java. Sans entrer dans le détail on peut citer les exceptions, les threads, le ramasse miettes, les liens dynamiques, la prise en compte des modifications sans recompilation, l'association avec un système d'exécution standard, des bibliothèques standards fournies qui masque bien le système sous jacent. Ce sont autant de facilités qui offre du confort dans la réalisation de petites portions de code sans mettre en oeuvre une grosse mécanique.

Néanmoins, l'exigence dans la question posée est de prendre un langage suffisamment clair et simple dans sa construction pour ne pas décourager le néophyte et suffisamment répandu pour être mis en oeuvre largement. Pour moi, l'Ada, héritier du Pascal, néanmoins moderne, ferait bien l'office s'il était un peu plus répandu. D'autre part, Ada est dit portable au niveau du source, Java est portable aussi au niveau du binaire. Il faut reconnaitre cet atout à Java qui supplante ainsi le C langage bien connu pour sa mauvaise portabilité.

Il resterait à dire quelques mots de C++, mais ne le connaissant pas suffisamment, je m'abstiendrais. Envoyer vos remarques, je les publierais à la suite.

Pascal Pignard, août 2001.