

Module et Généricité : Exemple de la pile

Objectifs

- Comprendre la notion de module (paquetage en Ada).
- Comprendre la généricité.
- Savoir gérer deux instances d'un même module générique.

Rappel : Comme pour tous les TP, il faut commencer par faire un « svn update » depuis votre dossier « pim/tp » pour récupérer les fichiers fournis pour cette séance.

Exercice 1 : Modules en Ada

L'objectif de cet exercice est de comprendre les modules en Ada. On s'appuiera en particulier sur le module `Piles` (fichiers `piles.ads` et `piles.adb`) et le programme `utiliser_piles.adb`.

1. Pourquoi les commentaires de spécification n'apparaissent que dans la spécification du module `Piles` et pas dans son implantation ?
2. Où sont formalisés les contrats ?
3. Où faudrait-il définir un sous-programme spécifique au module `Piles` ? Et sa spécification ?
4. Qu'est ce que la surcharge. La lecture de la procédure `Illustrer_Surcharge` aidera à répondre à cette question. On suivra en particulier les consignes données en commentaire et on décommentera la ligne indiquée.
5. La procédure `Afficher_Element` n'est pas un paramètre de généricité du module `Piles` mais de sa procédure `Afficher`. Pourquoi ? La lecture de `Illustrer_Plusieurs_Afficher_Pour_Meme_Pile` aidera à répondre à cette question.
6. Si dans un même contexte, par exemple un même sous-programme, on a besoin de deux piles avec des caractéristiques différentes (capacités différentes ou types des éléments différents), comment gérer le fait que les deux instances du module `Piles` fournissent les mêmes noms (`T_Pile`, `Empiler`, `Est_Vide`...) ? Ce cas de figure est présent dans `Illustrer_Plusieurs_Piles`.