Modules — Types utilisateurs

Objectifs

- Comprendre les modules (encapsulation et masquage d'information)
- Savoir définir et manipuler les types utilisateurs (énuméré, enregistrement et tableau)
- Spécifier, implanter et tester des sous-programmes... Toujours !

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 Le module *Dates* se présente en Ada sous la forme de deux fichiers `dates.ads` et `dates.adb`. Le fichier `exemple_dates.adb` est un programme utilisant les dates.
1. Expliquer à quoi correspondent les fichiers `.ads` et `.adb`.
2. Pourquoi n'y a-t-il pas de commentaire devant les sous-programmes `Initialiser`, `Le_Jour`, `Le_Mois` et `La_Date` de `dates.adb` ?
3. Pourquoi la sémantique (le commentaire) du sous-programme `Afficher_Deux_Positions` est donnée dans `dates.adb` ?
4. Lister ce qu'un utilisateur du module *Dates* peut utiliser de ce module.
5. Lire le fichier `exemple_dates_erreurs.adb`. Certaines instructions seront refusées par le compilateur. Les identifier et donner la raison du refus dans un commentaire de fin de ligne.
Contrôler ensuite les réponses en compilant le fichier.
Mettre en commentaire les lignes refusées par le compilateur.
6. Expliquer pourquoi le type `T_Mois` ne peut pas être déclaré privé.
7. Modifier la déclaration du type `T_Date` dans le fichier `dates.ads` pour en faire un type *très privé*. On remplacera **`is private`** par **`is limited private`**.
Quelles sont les incidences de cette modification ?
Compiler le fichier `exemple_dates_erreurs.adb` pour confirmer les réponses.

Exercice 2 : Gestion d'un stock de matériel informatique
Afin de connaître l'état de son stock de matériel informatique, une entreprise décide de réaliser un programme informatique. Un matériel informatique est caractérisé par un numéro de série (que l'on prendra entier mais qui dans une évolution future pourrait être une chaîne de caractères), sa nature (unité centrale, disque, écran, clavier ou imprimante), son année d'achat et son état qui indique s'il est en état de fonctionnement ou pas.

Les opérations que l'entreprise souhaite faire sur ce stock sont les suivantes :

1. Enregistrer un nouveau matériel dans le stock à partir de son numéro de série, sa nature et son année d'achat. On suppose que le nouveau matériel enregistré est en état de fonctionnement.

2. Obtenir le nombre de matériels enregistrés dans le stock.
3. Obtenir le nombre de matériels qui sont hors d'état de fonctionnement.
4. Mettre à jour l'état d'un matériel enregistrer dans le stock à partir de son numéro de série.
5. Supprimer du stock un matériel à partir de son numéro de série.
6. Afficher tous les matériels du stock. Ce sous-programme affiche dans le terminal.
7. Supprimer tous les matériels qui ne sont pas en état de fonctionnement.
8. ...

L'analyste a décidé (et impose donc) de définir un seul module (`stocks_materiel`) et de représenter le stock par un unique tableau.
1. Définir le type `Stock` (et les autres types utiles). On précisera les invariants de type.
2. Nous allons implanter maintenant les différentes opérations demandées par l'entreprise en construisant progressivement le module et un scénario [1] d'exemple. Pour chaque opération, on suivra les étapes suivantes :

1. Spécifier le sous-programme dans la spécification du module.
2. Utiliser le sous-programme dans le scénario (en utilisant `pragma Assert` pour en vérifier l'effet).
3. Coder le sous-programme dans l'implantation du module sans donner son code (on mettra `null;` pour une procédure ou on retournera une valeur quelconque pour une fonction).
4. Compiler et exécuter l'application. Le scénario ne devrait pas fonctionner.
5. Implanter le sous-programme dans la spécification du module.
6. Compiler et exécuter l'application. S'il y a des erreurs, il faut les corriger !
7. Pousser les modifications sur SVN en indiquant dans le message l'opération qui a été implantée.

Dans le squelette qui est donné, les étapes précédentes ont été réalisées pour trois sous-programmes : créer le stock (sous-programme qui doit être appelé sur un stock avant de pouvoir l'utiliser), nombre de matériels dans le stock et enregistrer un matériel. Cependant leur implantation correcte n'a pas été donnée dans l'implantation du module.
Attention : Sauf indication contraire, aucun sous-programme n'est interactif : ils ne doivent pas faire d'entrée/sortie (clavier/écran).

Construire progressivement le module.

Pour aller plus loin...

Exercice 3 : Autre organisation en module
Nous avons fait un seul module pour représenter le stock de matériel informatique. Il aurait été plus judicieux d'en faire davantage.
1. Indiquer les modules qui auraient dû être faits.
2. Restructurer le code pour les faire apparaître.

1. On pourrait aussi définir des programmes de test pour les sous-programmes.