TD -TP POLYNOME version 1 (13A)

Thème : Conception, réalisation et test d'un T.A.D. (`T_Polynome`). Ce travail se fera en **3 étapes de quatre heures chacune**. Seule la première approche (simple) est détaillée ici. Deux autres documents suivront, l'un (TD13B) pour améliorer le paquetage vu ci-dessous, le dernier (TD17) pour tester un paquetage en utilisant les techniques vues au cours sur la testabilité.

Concepts (encore les mêmes !) **: type abstrait de données** T.A.D. : type exportable (via un paquetage) et **méthodes** associées, masquage d'information, surcharge, généricité. Pas d'exception (c'est pour simplifier). Mais surtout, et c'est nouveau, le type article **mutable**.

Thème : (simulation d'une calculette de polynômes) **:**

Soit un polynôme P à coefficients « **rationnels** » a_i mais d'une variable réelle X :
$P(X) \equiv a_0 + a_1X + a_2X^2 + ... + a_nX^n$. Notation formelle (voir exemples plus bas).

On appellera n le degré du polynôme.

On désire réaliser un type abstrait T.A.D. modélisant ces objets caractérisés pour chacun :

- par leur degré maxi n (de 0 à Max) !
- et par la suite de leurs coefficients a_i (i = 0,1,....,n) de type T_Rationnel (cf. TD11). La variable X étant implicite (réelle) ne sera pas modélisée dans la structure de donnée.

Quatre exemples :

$P(X) = 6/3 + 5/2\ X - 7/4\ X^2 + 4/5\ X^3 \equiv$ degré 3 et coefficients 6/3 , 5/2 , -7/4 , 4/5
$Q(X) = 2/3 - 1/2\ X + 3/4\ X^2 + 4/5\ X^3 \equiv$ degré 3 et coefficients 2/3 , -1/2 , 3/4 , 4/5
$R(X) = 3/2$
$Y(X) = -5\ X^2 + 4/3\ X^5$ ← Degrés ? listes des coefficients ? (de R et de Y)

On souhaite concevoir la plupart des « méthodes » pour travailler avec ces objets (voir plus loin les spécifications essentielles à réaliser), par exemple l'opérateur binaire "-".

Ainsi : $P(X) - Q(X) = 4/3 + 6/2\ X - 10/4\ X^2$. **Le degré a diminué ! et il faudrait réduire !**

On réalisera **aussi** (pour ce premier travail) :

- la fonction "+" (binaire) et la fonction "–" unaire,
- les fonctions binaires "*", "/" et "rem" (plus difficiles !)
- un constructeur nommé Set qui permettra d'initialiser un polynôme avec une liste de coefficients rationnels et une procédure Ecrire pour éditer un polynôme. Pas de Lire !
- la fonction Derivee telle que $Derivee(P) = 5/2 - 7/2\ X + 12/5\ X^2$

Ces concepts connus (BAC !) seront rapidement commentés en TD avec un petit exemple.

Comment modéliser nos polynômes ? Quelle structure de données utiliser ? Comme pour les **`Bounded_String`** que nous avons simulés (TD12B) on écrit qu'un polynôme est un « couple » formé de son degré effectif et de sa liste de coefficients. On pourrait reprendre le même modèle : un **article et ses deux champs**. Il paraît plus astucieux d'utiliser un **article mutant** où le discriminant jouera le rôle du premier champ et servira à dimensionner provisoirement la liste des coefficients (qui évolue avec l'usage) :

Soit (en partie privée évidemment) :

```
type T_Polynome (Degre : T_Degre := 0) is -- type mutant
record
   Coef : T_Vect_Coef (0..Degre) := (others => Nulle);
end record;
```

Le type `T_Degre` est un type Entier de 0 à Max. Cette valeur Max doit être modulable ! Ce sera **l'un des paramètres de généricité** !
Le type `T_Vect_Coef` est évidemment un type tableau de Rationnels ! A définir et pour cela nous avons besoin du paquetage réalisé au TD-TP 11 ! Or, il est **lui même générique**, nous en ferons aussi un paramètre de généricité de notre nouveau paquetage ! Joli, non ?
La liste des coefficients est a priori dimensionné à degré zéro (`Degre : T_Degre := 0`) donc le polynôme constant a la valeur elle même égale à 0 (**`others`** `=> Nulle`) revoyez le TD11 !

Le début du paquetage **pourrait commencer** ainsi :

```
with P_Rationnels_G;
generic
   Max : Positive;   -- degré maxi des polynômes
   with package P_Rationnels is new P_Rationnels_G(<>);
package P_Poly_V1_G is

   use P_Rationnels;

   subtype T_Degre is Natural range 0..Max;

   type T_Polynome (Degre : T_Degre := 0) is private;
```

Il en manque encore !

….. à finir avec les spécifications des méthodes suggérées bas de page précédente.

Remarque : l'édition d'un polynôme (Ecrire) induit l'édition d'une liste de rationnels ! Toute chose réalisée déjà mais … dans le paquetage `P_Rationnels_G.Io` fils de `P_Rationnels_G`. Il faut **y faire référence aussi donc l'ajouter en paramètre de généricité** !! Re joli !

Passons à **l'écriture commentée** en TD des spécifications complètes (page suivante) :

```
with P_Rationnels_G, P_Rationnels_G.Io;
generic
   Max : Positive;   -- degré maxi des polynômes
   with package P_Rationnels is new P_Rationnels_G(<>);
   with package P_Rationnels_io is new P_Rationnels.Io(<>);
package P_Poly_V1_G is

    use P_Rationnels, P_Rationnels_Io;

   subtype T_Degre is Natural range 0..Max;

   type T_Polynome (Degre : T_Degre := 0) is private;
   -- le T.A.D. en question

   -- sous-programmes ou méthodes
   function "+" (Poly_A, Poly_B : in T_Polynome) return T_Polynome;
   -- addition de deux polynômes
   function "-" (Poly_A, Poly_B : in T_Polynome) return T_Polynome;
   -- différence de deux polynômes
   function "-" (Poly_A : in T_Polynome) return T_Polynome;
   -- le moins unaire
   function "*" (Poly_A, Poly_B : in T_Polynome) return T_Polynome;
   -- produit de deux polynômes
   function "/" (Poly_A, Poly_B : in T_Polynome) return T_Polynome;
   -- quotient  de deux polynômes
   function "rem" (Poly_A, Poly_B : in T_Polynome) return T_Polynome;
   -- reste de la division  de deux polynômes
   function Derivee (Poly_A : in T_Polynome) return T_Polynome;
   -- dérivée classique sans commentaire
   procedure Ecrire (Poly : in T_Polynome);
   -- pour éditer dans les tests succincts

   -- constructeur Set :
   type T_Liste_Coef is array (Positive range <>) of T_Rationnel;

   procedure Set (Poly : out T_Polynome; Vect : in T_Liste_Coef);
   -- permet d'initialiser directement un polynôme
   -- avec une liste de coefficients tel que :
   -- Le_Poly := (-1/3,0,3/2,-6); -- pour -1/3 +(3/2)X**2 -6X**3
   -- plus tard (version 2) on ajoutera exceptions et
   -- d'autres méthodes (accesseurs et modifieurs) et plus d'E/S.

private

   type T_Vect_Coef is array (T_Degre range <>) of T_Rationnel;

   type T_Polynome (Degre : T_Degre := 0) is -- type mutant
   record
      Coef : T_Vect_Coef (0..Degre) := (others => Nulle);
   end record;
   -- ainsi une instance de ce type est le polynôme nul

   function Poly_Reduit (Poly: in T_Polynome) return T_Polynome;
   -- utile pour la hiérarchie des paquetages (à venir !)
end P_Poly_V1_G;
```

A commenter en TD !!
Imaginez les algorithmes

Travail à réaliser : voir fichier `p_poly_v1_g.adb.squ` (à renommer et à compléter).

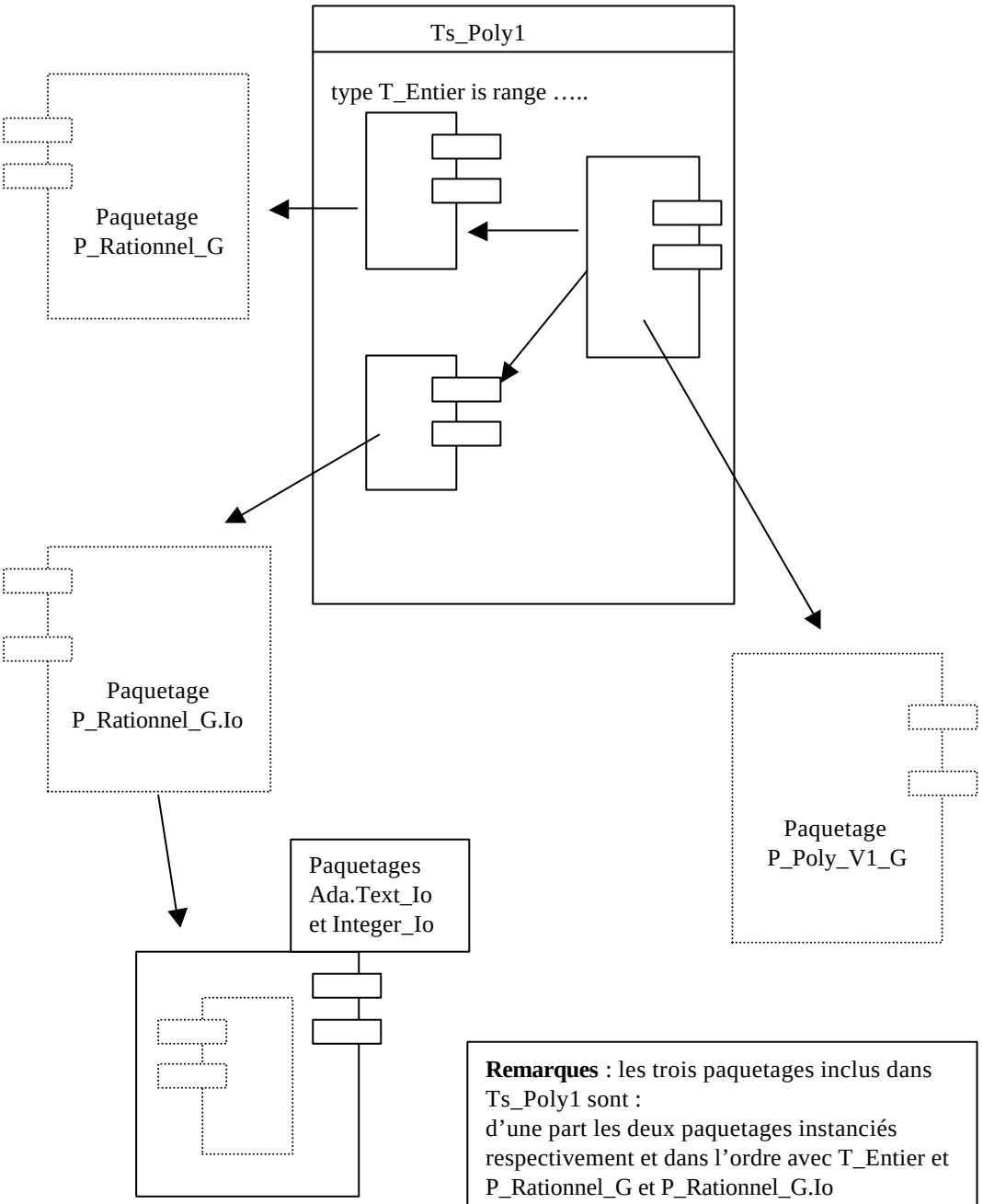
Le travail est assez bien résumé par les spécifications de `p_poly_v1_g.ads`. (page 3) déjà commentées en TD. Le reste du TD (2 heures) sera consacré à mettre en place quelques algorithmes mais **sans les coder en Ada** ! A faire en TP !

Notons que :

- `Max` et les paquetages `P_Rationels_G`, `P_Rationnels_G.Io` sont **paramètres génériques.** Plus précisément ce sont les paquetages appelés `P_Rationnels` et `P_Rationnels_Io` futures instanciations de `P_Rationels_G` et de `P_Rationnels_G.Io` qui sont génériques !
- La structure de données `T_Polynome` « type article mutant » est « cachée » dans un `private` empêchant l'utilisateur du paquetage d'y accéder ; d'où, la nécessité (la séance prochaine) d'offrir des accesseurs et des modifieurs pour parfaire l'utilisation.
- Les « erreurs » (ou exceptions) levées éventuellement sont négligées **aujourd'hui** (à exclure dans les jeux de tests). Le TP est assez conséquent comme cela.
- Notez **en partie privée** la fonction `Poly_Reduit` **utile pour les body des fils** éventuels (notion à revoir) et non offerte aux utilisateurs puisque toutes les opérations (y compris le `Set`) font le travail de réduction.

TP 13A (2 heures)

- Créez un répertoire tp13A.
- Copiez les fichiers fournis comme d'habitude.
- Réalisez le body du paquetage `P_Poly_V1_G`. **Rendez, à la fin, un listing formaté du body** (partez du squelette).
- Voyez et complétez le programme `Ts_Poly1`. Fichier `ts_poly1.adb.squ` (à renommer `ts_poly1.adb`) pour tester les sous-programmes. Voir le schéma (page suivante) pour instancier les différents paquetages génériques. **C'est cette petite partie qui est à comprendre et à réaliser correctement.**
- Exécutez le programme `Ts_Poly1`. Indiquez les S/P sérieusement testés. On **redirigera les Sorties écran** dans un fichier `fic_poly.out`. Refaire avec des `Set` différents (Utilisez des rationnels quelconques puis particuliers). A revoir à la séance n°3 (tests unitaires de la testabilité). Attention, aujourd'hui, **n'utilisez pas de données susceptibles de lever des exceptions** : ce n'est pas le but de ce TP (pour cette semaine).
- Le prochain TP utilisant celui-ci, il convient de repartir sur des bases solides. **A finir absolument pour la semaine suivante !**



Remarques : les trois paquetages inclus dans Ts_Poly1 sont :
d'une part les deux paquetages instanciés respectivement et dans l'ordre avec T_Entier et P_Rationnel_G et P_Rationnel_G.Io
d'autre part le troisième paquetage permettant les polynômes est instancié avec P_Poly_V1_G et les deux précédents instanciés.
Dur !