# TD et TP Numéro 12 (suite) 12B

**Objectif :** Après avoir découvert (et testé) le type **`Bounded_String`** avec les fonctionnalités du paquetage générique **`Ada.Strings.Bounded.Generic_Bounded_Length`** il parait formateur de « réécrire » quelques uns des sous-programmes fondamentaux. (On peut aussi imaginer en ajouter d'autres). Traditionnellement quand on souhaite ajouter ou modifier des fonctionnalités **il faut créer un paquetage fils du paquetage en question** permettant ainsi d'accéder à la structure (cachée) implémentant le type en question.
Cependant il n'est pas possible de créer un paquetage fils d'un paquetage lui même inclus dans un autre paquetage (ce qui est le cas ici). Aussi notre simulation se fera à partir d'un paquetage générique nommé **`P_Bounded_G`** (sorte de clone du paquetage officiel) mais un peu plus réduit. ). Remarquons auparavant que le TD sera, cette fois, plutôt court (1 heure) alors que le TP prendra le reste du temps (3 heures).

**Méthode :** (Voir schéma page 4). A partir du paquetage générique **`P_Bounded_G`** définissant l'essentiel de la structure de donnée et une seule méthode, on va découvrir un paquetage fils appelé **`P_Bounded_G.P_Bounded_G_Io`** (dans lequel on définit les E/S sur les chaînes de type **`Bounded_String`**) **mais vous allez créer** le paquetage **`P_Bounded_G.P_New_Bounded_G`** dans lequel vous allez redéfinir et réaliser vous mêmes certaines fonctionnalités (toutes si vous le pouvez !) concernant les nouvelles chaînes de type **`Bounded_String`**.

Le paquetage père **`P_Bounded_G`** étant générique, **`P_Bounded_G.P_New_Bounded_G`** (son fils) doit être également générique (revoir votre cours sur les paquetages ; vu aussi au TP11 sur les E/S des Rationnels). Le corps du paquetage fils a accès à la partie privée du paquetage père. Ceci est important car vous aurez besoin de travailler sur la structure de **`Bounded_String`**.

Le type **`Bounded_String`** est construit (en partie privée de **`P_Bounded_G`**) à partir d'un type « record » de la façon suivante :

```
type  Bounded_String  is  record
      Length : Length_Range := 0 ;
      Data     : String  (1..Max_Length) ;
end record ;
```

On remarque que la structure de donnée est formée de deux entités : la longueur utile **`Length`** qui sera gérée automatiquement par les méthodes (initialisée à zéro !) et ensuite la donnée vraie à savoir la chaîne de caractères dimensionnée au maximum possible de l'instanciation.

On définit aussi les éléments utiles :

```
Max_Length :  constant  Positive :=  Max ;
subtype   Length_Range  is  Natural  range  0..Max_Length ;
```

Max étant le paramètre de généricité du paquetage père.

On verra avec intérêt en le commentant le fichier des spécifications page 5.

Nous vous proposons de réécrire les fonctions et procédures suivantes :

```
function To_String (
      Source : in     Bounded_String )
  return String;

function To_Bounded_String (
      Source : in     String )
  return Bounded_String;

function "&" (Left,
      Right : in     Bounded_String )
  return Bounded_String;

function "&" (Left  : in     Bounded_String;
      Right : in     Character       )
  return Bounded_String;

function "=" (Left,
      Right : in     Bounded_String )
  return Boolean;

function "<=" (Left,
      Right : in     Bounded_String )
  return Boolean;

function Index (
      Source  : in     Bounded_String;
      Pattern : in     String          )
  return Length_Range;

procedure Insert (
      Source   : in out Bounded_String;
      Before   : in     Positive;
      New_Item : in     String          );

procedure Delete (
      Source  : in out Bounded_String;
      From    : in     Positive;
      Through : in     Natural         );

procedure Overwrite (
      Source   : in out Bounded_String;
      Position : in     Positive;
      New_Item : in     String          );

function "*" (Left  : in     Natural;
      Right : in     Character )
  return Bounded_String;
```

Extrait des spécifications du paquetage P_New_Bounded_G fils du paquetage P_Bounded_G

La compréhension est simplifiée si on a participé activement au TD TP 12 A précédent !

**Attention :** Pour simplifier la fonction **`Index`** et les procédures **`Insert`** et **`Overwrite`** ne comportent pas tous les paramètres présents dans les sous-programmes initiaux contenus dans **`Ada.Strings.Bounded.Generic_Bounded_Length`**. Les seules exceptions traitées sont **`Index_Error`** et **`Length_Error`** (voir spécifications de **`P_Bounded_G`** page 5).

Pour les entrées sorties il faut définir (comme pour le TD TP 12A) un paquetage fils de **`P_Bounded_G`**. Les spécifications **étant rigoureusement les mêmes** nous ne nous y arrêtons pas voir (en TP) **`P_Bounded_G.P_Bounded_G_Io`**. Le body méritera un regard intéressant !

La réalisation concrète d'un vrai paquetage se fera comme dans le TP 12A en l'instanciant dans un petit fichier comme ces deux exemples :

```
--  Fichier  p_bounded_80.ads
with  P_Bounded_G;
package P_Bounded_80 is new P_Bounded_G (80);

--  Fichier  p_bounded_io_80.ads
with P_Bounded_G.P_Bounded_G_Io, P_Bounded_80;
use  P_Bounded_80;
package P_Bounded_Io_80 is new P_Bounded_80.P_Bounded_G_Io ;
```

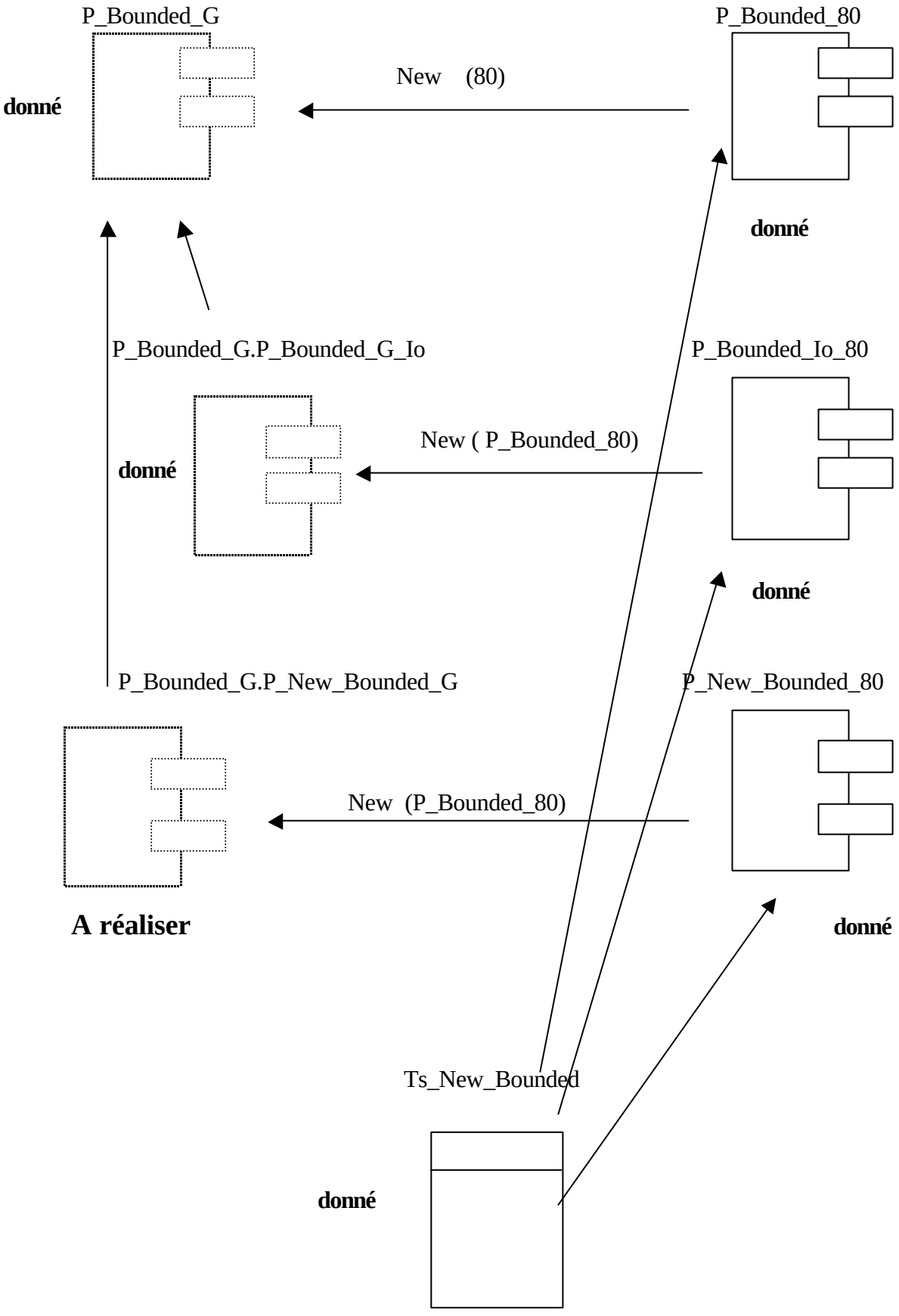
enfin (quand le body sera réalisé et avant de tester)

```
-- fichier  p_new_bounded_80.ads
with P_Bounded_G.P_New_Bounded_G, P_Bounded_80;
use  P_Bounded_80;
package P_New_Bounded_80 is new P_Bounded_80.P_New_Bounded_G;
```

## Détails du TP (3 heures) :

- Créez un répertoire tp12B.
- Copiez les fichiers copiables depuis le répertoire de votre enseignant.
- Le fichier **`p_bounded_g.ads`** (donné en annexe) contient toutes les spécifications du paquetage père. Le fichier **`p_bounded_g.adb`** est très simple il réalise **`Length`** !
- Les fichiers **`p_bounded_g-bounded_g_io.ads`**, **`p_bounded_g-p_bounded_g_io.adb`** définissent et réalisent les E/S nous en avons parlé précédemment.
- Les fichiers **`p_bounded_80.ads`**, **`p_bounded_Io_80.ads`** simples instanciations créent les vrais paquetages **`P_Bounded_80`** et **`P_Bounded_Io_80`**. (cf. ci dessus)
- Le fichier **`p_bounded_g-p_new_bounded_g.ads`** contient les spécifications du paquetage fils. **Il faut réaliser le body**. Partez du fichier d'extension .adb.squ que vous renommez avec l'extension adb.
- Insérez les entêtes partout.
- Compilez avec **`gnat gcc –c p_bounded_g-p_new_bounded_g.adb`**. Vous obtenez les deux fichiers **`p_bounded_g-p_new_bounded.o`** et **`.ali`**.
- Examinez le fichier **`ts_new_bounded.adb`**. Il s'appuie sur une réalisation concrète du paquetage **`P_New_Bounded_80`** réalisé grâce à la compilation (cf. ci dessus) de **`p_new_bounded_80.ads`**. Pour créer l'exécutable : **`gnatmake ts_new_bounded`**. Le test utilise le fichier **`bounded.in`** et crée le fichier **`bounded.out`**.
- Créez un bon fichier **`bounded.in`**.
- Faites tourner et vérifiez le bon fonctionnement de vos fonctions et de vos procédures. Pour faciliter ce contrôle vous comparerez les résultats avec un fichier de référence **`bounded.ora`**. On pourra le créer en lançant la commande **`Crea_Ora`**.

## TP 12 Partie II



```
-- fichier p_bounded_g.ads
-- spécifications d'un clone de Generic_Bounded_Length
-- mettre votre entête
-- ne changez pas ce paquetage !
-- pour le compléter utilisez un paquetage fils
-- P_Bounded_G.P_New_Bounded_G
-- voir le fichier p_bounded_g-p_new_bounded_g.ads
-- complétez en ajoutant les sous programmes que vous
-- vous sentez capables de réaliser
-- réalisez le body associé et surtout TESTEZ !


generic
   Max : Positive;
package P_Bounded_G is

   Max_Length : constant Positive := Max;
   -- redéfinition du paramètre générique

   type Bounded_String is private;

   subtype Length_Range is Natural range 0..Max_Length;
   -- utile pour le body

   function Length (
         Source : in     Bounded_String )
     return Length_Range;
   -- seule méthode proposée !
   -- complétez mais pas ici !

   Length_Error  : exception;
   -- levée quand la chaîne Data sera saturée
   -- identique à Truncation := Error
   Index_Error   : exception;
   -- levée quand un indice est mal situé (à discuter !)

private

   type Bounded_String is
      record
         Length : Length_Range := 0;
         -- une instance est "vide" à la déclaration
         Data : String (1 .. Max_Length);
         -- chaine maximum possible
      end record;

end P_Bounded_G;
```