Cours n°2 : Types objet en Pascal

Alain Giorgetti

giorgetti@univ-fcomte.fr

http://lifc.univ-fcomte.fr/~giorgett/

Laboratoire d'Informatique

de l'Université de Franche-Comté

Plan de l'exposé

- Rappels de programmation Pascal
- Des enregistrements aux objets
- Syntaxe Pascal
  - Type objet : exemple, syntaxe
  - Types objets et unités Pascal
  - Utiliser un type objet
  - Objets (statiques) : exemples, accès aux champs, ...
  - Méthodes (statiques) : appel, implantation
  - Le mot réservé **`Self`**
  - Spécification d'une méthode

Rappels de programmation Pascal

- Programmation structurée
  - Données organisées en **types**
    - tableaux

```
ARRAY […] OF … ;
```

    - enregistrements

```
TDate = RECORD
            jj : 1..31;
            mm : 1..12;
        END;
```

  - Procédures et fonctions
    - Associées aux types : initialisation, accès, modification, ...
    - Une procédure ou fonction par fonctionnalité
    - Nombre d'instructions réduit : peu d'erreurs
    - Toutes les variables modifiées sont des paramètres

Des enregistrements aux objets

- Objet = enregistrement + procédures + fonctions
- Type objet = type enregistrement + déclaration de procédures + déclaration de fonctions
- ***Type objet*** (ou *classe*), modèle des objets :
  - des ***champs*** de données (comme dans un enregistrement)
  - des ***méthodes*** de traitement : procédures et fonctions intégrées au type objet
- Syntaxe Pascal
  - **`OBJECT`** remplace **`RECORD`**
  - nouveaux mots réservés : **`Self`**, **`inherited`**, ...

Type objet : exemple

```
TDate = OBJECT
    jour : 1..31;
    mois : 1..12;
    PROCEDURE entrer(jj, mm : byte);
     {trouve et stocke la date existante
      la plus proche de jj/mm}
    PROCEDURE lendemain;
     {calcule la date du lendemain}
    FUNCTION finMois : boolean;
     {VRAI si la date est la dernière du mois
      d'une année non bissextile,FAUX sinon}
END;
```

champs

méthodes

Types objet et unités

- Règles
  - une unité par type objet
  - un type objet par unité
  - noms d'unité et de type objet cohérents
- Interface
  - types (objet) liés : **USES**
  - déclaration du type objet
  - liste des champs
  - liste des méthodes
- Implantation
  - méthodes dans le même ordre que dans la partie interface

```
UNIT nomType;
INTERFACE
USES unitéUtile;
TYPE
  TNomType = OBJECT
    champ1 : TChamp1;
    champ2 : TChamp2;
    PROCEDURE meth1(…);
    FUNCTION meth2(…):…;
  END;
IMPLEMENTATION
  …
END.
```

Type objet : exemple plus complet

```
UNIT date;
INTERFACE
  USES mois,jour;
          {unités définissant les types TJour et TMois}
  TYPE
     …
    TDate = OBJECT
       jour : TJour; {numéro du jour entre 1 et 31}
       mois : TMois; {numéro du mois entre 1 et 12}
       …
       PROCEDURE entrer(jj, mm : byte);
        {trouve et stocke la date existante la plus
         proche de jj/mm}
       PROCEDURE lendemain;
        {calcule la date du lendemain}
       FUNCTION finMois : boolean;
        {VRAI si la date est la dernière du mois
         d'une année non bissextile,FAUX sinon}
       …
    END;
     …
IMPLEMENTATION …
```

Type objet : syntaxe générale

```
TNomType = OBJECT
     nomChamp : TChamp;
     PROCEDURE nomProc(nomParam : TParam);
     FUNCTION nomFonc(nomParam : TParam):TRetour;
END; {fin de la déclaration du type objet}
```

- Il peut y avoir aussi
  - d'autres champs, d'autres méthodes
  - des méthodes avec plusieurs paramètres
  - des paramètres passés par donnée ou par variable (**`var`**)
- *Cette syntaxe sera étendue dans les cours suivants, pour traduire l'héritage, les champs privés, les méthodes virtuelles …*

Utiliser un type objet : exemple

```
PROGRAM utilise;
VAR
  hier : TDate;
  ceJour : TDate;
BEGIN
  hier.jour := 4;      {Accès direct aux champs}
  hier.mois := 10;
  hier.entrer(15,14); {Accès contrôlé aux champs}
  write(hier.jour);    {Affichage du jour calculé}
  ceJour := hier;      {copie de la date}
  ceJour.lendemain;    {calcul du jour suivant}
  writeln(ceJour.jour);
  write(hier.jour);    {résultat ?}
END.
```

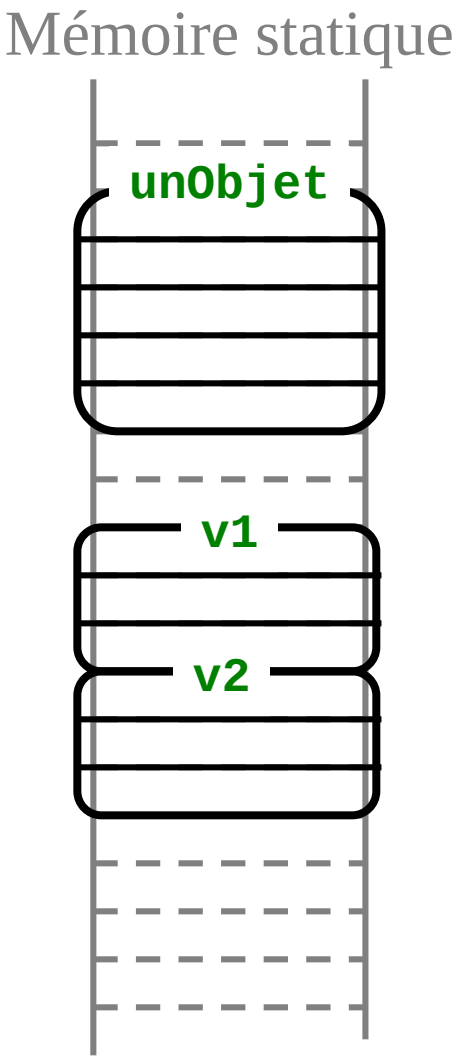
2 objets (statiques) de type **TDate**

Déclaration d'objets statiques

- Variables de type objet

```
VAR
   unObjet : TClasse;
   v1,v2 : TAutreClasse;
```

- Déclaration
  - d'un espace mémoire réservé
  - d'un nom donné à cet espace mémoire
  - taille de l'espace : taille de l'objet
- Une variable de type objet (**`unObjet`**, **`v1`**, **`v2`**) est appelée *objet statique*



Accès aux champs d'un objet

- Affectation d'une valeur à un champ

```
unObjet.nomChamp := expression;
```

  - l'expression est évaluée
  - la valeur de l'expression est donnée au champ

- Utilisation de la valeur d'un champ

```
variable := unObjet.nomChamp;
proc(unObjet.nomChamp);
```

- Exemples

```
hier.mois := 4;
nbHeure := (ceJour.jour - hier.jour)*24;
write(hier.jour);
```

Copie d'objets statiques

- Affectation entre variables objet

```
objetCopie := objetSource;
```

- Condition : types objets identiques ou *compatibles*
- Effets
  - champs copiés un par un
  - ancienne valeur des champs de la copie perdue
  - champs de la source non modifiés
- Exemple

```
ceJour := hier;
```

effectue

```
ceJour.jour := hier.jour;
ceJour.mois := hier.mois;
```

Méthode : exemple

```
TDate = OBJECT
  j : TJour; m : TMois;
   { … }
  PROCEDURE entrer(jj,mm : byte);
   { … }
END;
```

Déclaration de la méthode, dans une unité

```
…
PROCEDURE TDate.entrer(jj,mm : byte);
VAR max : byte;
BEGIN
  IF (jj >= 1) AND (jj <= 28) THEN
    Self.j := jj
  ELSE ...
END;
```

Implantation de la méthode, dans la même unité

Appel de la méthode, dans une autre unité

```
VAR unJour : TDate;
BEGIN
  …
  unJour.entrer(29,12);
  …
END;
```

Appel d'une méthode (procédure)

```
TNomType = OBJECT
  champ : TChamp;
  PROCEDURE methode(param1:T1; VAR param2:T2);
END;
```

- Appel `unObjet.methode(arg1,arg2);`
  - Conditions
    - Objet appelant **`unObjet`** de type **`TNomType`**
    - Types des arguments compatibles avec les types des paramètres, dans le même ordre
  - Effets
    - Transmission des arguments réels (**`arg1`**, **`arg2`**) aux paramètres formels (**`param1`**, **`param2`**)
    - Exécution de la procédure, à partir des valeurs des arguments et des champs (**`champ`**) de l'objet appelant

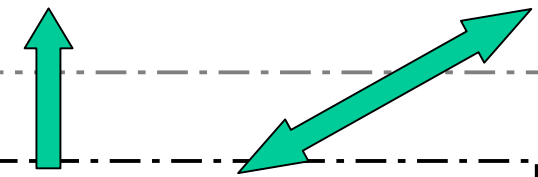
Appel d'une méthode (fonction)

```
TNomType = OBJECT
  champ : TChamp;
  FUNCTION methode(listeParam) : TRetour;
END;
```

- Appel `result := unObjet.methode(listeArg);`
  - Conditions
    - Objet appelant **`unObjet`** de type **`TNomType`**
    - Types des arguments compatibles avec les types des paramètres
  - Effets
    - Transmission des arguments aux paramètres
    - Exécution de la fonction, qui calcule une valeur dans la variable locale **`methode`** de type **`TRetour`**
    - Retourne la valeur calculée dans la variable **`result`**

Transmission des arguments

```
TNomType = OBJECT
  PROCEDURE methode(param1 : T1; VAR param2 : T2);
end;
```



```
unObjet.methode(arg1,arg2);
```

- Passage des arguments aux paramètres
  - *par valeur* : la valeur de **`arg1`** est copiée dans **`param1`**
  - *par adresse* (**`VAR`**) : **`param2`** désigne **`arg2`** lui-même dans le corps de la méthode (même *adresse* en *mémoire*)
- Condition : types compatibles
  - argument **`arg1`** de type **`T1`** ou compatible
  - argument **`arg2`** de type **`T2`** ou compatible

Implantation d'une méthode

```
TNomType = OBJECT
  champ : TChamp;
  PROCEDURE proc(listeParam);
  FUNCTION fonc(listeParam) : TRetour;
END;
```

Implantation avec le nom du type objet

```
PROCEDURE TNomType.proc(listeParam);
BEGIN
    …
END;
FUNCTION TNomType.fonc(listeParam) : TRetour;
BEGIN
    …
   fonc := …;
END;
```

Le mot réservé **Self**

- En général, une méthode utilise ou modifie les champs de son *objet appelant* (**unObjet**)

```
unObjet.methode(…);
```

```
PROCEDURE TNomType.methode(…);
BEGIN
  Self.champ := …;
  Self.autreMethode(…);
END;
```

- Dans le corps de la méthode, le mot réservé **Self** désigne l'objet appelant

```
PROCEDURE TDate.entrer(j,m : byte);
VAR ...
BEGIN
  IF ... THEN
    Self.jour := j
  ELSE ...
END;
```

- L'objet appelant n'est pas un paramètre de la méthode …

Spécification de méthodes : exemples

```
TDate = OBJECT
    jour : TJour;
    mois : TMois;
    {Self=(jour=j,mois=m) est une date existante}
    PROCEDURE lendemain;
    {Self=(jour=nj,mois=nm), la date nj/nm suit
     la date j/m dans un calendrier sans année
     bissextile}

    {Self est une date existante}
    FUNCTION finMois : boolean;
    {retourne VRAI si Self est le dernier jour
     du mois d'une année non bissextile,
     FAUX sinon}
  END;
```

Spécification d'une méthode

- Formalisme

```
{ pré-condition sur l'objet appelant
 (Self) et sur les paramètres }
nomMethode(listeParamètres);
{ post-condition }
```

- Signification
  - le fonctionnement de la méthode n'est garanti que si la pré-condition est vraie
  - La post-condition exprime l'effet de la méthode, sur l'objet appelant et sur les paramètres
- En Pascal : commentaires dans l'interface