UNIVERSITE SAAD DAHLAB DE BLIDA
DÉPARTEMENT D' INFORMATIQUE

# Cours de Bases de données 2

## Le Relationnel Objet : SQL3 Modèle et Langage (Partie 3 « Les méthodes » )

# Plan du Cours

- ✓Définition d’une méthode
- ✓Utilisation d’une méthode
- ✓Surcharge et redéfinition des méthodes

# Les méthodes

Une méthode (**Une fonctions ou Une procédure**) est la modélisation d'une action applicable sur un objet, caractérisée par un en-tête appelé **signature** définissant **son nom, ses paramètres d'appel et de retour**, et qui permet de modifier l'état de l'objet ou de renvoyer un résultat.

En **relationnel objet**, les types peuvent admettre soit des fonctions soit des procédures.

**On déclare les méthodes :**

- Soit au début lors de la déclaration de l'objet
- Soit plus tard avec commande ALTERTYPE

# Les méthodes Membres

# Les méthodes : Déclaration de type avec méthodes

```
CREATE TYPE nom_type AS OBJECT (
 nom_attribut1 type_attribut1
 ...
 MEMBER FUNCTION nom_fonction1 (parametre1   type_parametre1, ...) RETURN
type_fonction1
 ...
);
/
CREATE TYPE BODY nom_type
 IS
 MEMBER FUNCTION nom_fonction1 (...) RETURN type_fonction1
  IS
  BEGIN
  ...
  END ;
 MEMBER FUNCTION nom_fonction2 ...
  ...
 END ;
END ;
/
```

**il y a deux sortes de méthodes : MEMBER et STATIC**

# Les méthodes Membres

Les méthodes membres (Méthodes d'instances) ont un paramètre intégré nommé SELF qui désigne l'instance d'objet appelant actuellement la méthode. Le mot clé **MEMBER** précède chaque méthode d'instance.

```
CREATE OR REPLACE TYPE Trectangle  AS OBJECT (
 Longueur Float,
 Largeur Float,
  MEMBER FUNCTION surface RETURN Float,
 MEMBER FUNCTION périmètre RETURN Float
);
```

# Les méthodes Membres

```
CREATE OR REPLACE TYPE BODY Trectangle
AS MEMBER FUNCTION surface RETURN Float IS
 BEGIN
RETURN  Longueur * Largeur;
-- RETURN SELF. Longueur * SELF. Longueur; -- équivalent à la ligne précédente
END;
 MEMBER FUNCTION périmètre  RETURN  Float IS
 BEGIN
RETURN 2 * (Longueur+ Largeur);
END;
END;
```

# Les méthodes (Invocation d'une fonction)

CREATE TABLE ***RECTANGLES*** of **TRECTANGLE**;

- **INSERT INTO RECTANGLES VALUES(10.2, 3.3);**
- **INSERT INTO RECTANGLES VALUES(16.2, 6);**

SELECT **r**.périmètre() , **r**.surface() FROM RECTANGLES **r** WHERE **r**.largeur = 6;

| R.PÉRIMÈTRE() | R.SURFACE() |
|---|---|
| 44,4 | 97,2 |

# Les méthodes membres (Invocation)

```
DECLARE
  rect TRECTANGLE;
BEGIN -- Bloc PL/SQL pour sélectionner un RECTANGLE et effectuer des opérations

 SELECT VALUE(r) INTO rect FROM RECTANGLES r WHERE r.largeur  = 6;

DBMS_OUTPUT.PUT_LINE('la surafec =' || rect.surface() || ' Le périmètre =' ||rect.périmètre());
END;
```

```
Procédure PL/SQL terminée.

la surafec =97,2 Le périmètre =44,4
```

# Les méthodes (Ajout d'une méthode )

Ajouter une méthode pour un type Objet se fait en deux étapes:

- La première la définition de la signature de la méthode
- La deuxième phase est la déclaration de corps de la méthode « body »

**Exemple 1:Ajouter une méthode afficher pour le type Trectangle**

**Signature: alter type Trectangle add member PROCEDURE afficher cascade;**

# Les méthodes (Ajout d'une méthode )

```
Create or replace type body Trectangle as
MEMBER FUNCTION surface RETURN Float IS
 BEGIN
---------
END;
MEMBER FUNCTION périmètre  RETURN  Float IS
 BEGIN
RETURN 2 * (Longueur+ Largeur);
END;
Member  PROCEDURE afficher  is
Begin
 DBMS_OUTPUT.PUT_LINE('la surafec =' || self.surface() || ' Le périmètre =' ||self.périmètre());
End ;
End;
```

# Les méthodes (Invocation d'une procédure )

```
DECLARE
  rect TRECTANGLE;
BEGIN
 SELECT VALUE(r) INTO rect FROM RECTANGLES r WHERE r.largeur  = 6;
 rect.afficher ();
END;
```

```
Procédure PL/SQL terminée.

la surafec =97,2 Le périmètre =44,4
```

# Les méthodes (Ajout d'une méthode )

**Exemple 2: Ajouter une méthode qui permet de modifier la largeur et une autre méthode pour récupérer la  valeur de la largeur**

**Signature méthode 1 : alter type Trectangle  add member PROCEDURE setLargeur (largeur float) cascade;**

**Signature méthode 2 : alter type Trectangle  add member  function getLargeur  return float cascade;**

# Les méthodes (Ajout d'une méthode )

Create or replace type body Trectangle as

MEMBER FUNCTION surface RETURN Float IS……END;

MEMBER FUNCTION périmètre  RETURN  Float IS…..END;

MEMBER PROCEDURE afficher  is……End ;

```
MEMBER PROCEDURE SETLARGEUR ( LARGEUR float) is
Begin
  self.LARGEUR:=LARGEUR;
End ;
MEMBER FUNCTION GETLARGEUR  RETURN  Float IS
 BEGIN
RETURN   Largeur;
END;
```

# Les méthodes Statiques

# Les méthodes Statiques

La méthodes statique (Méthode de classe ) n‘a pas l'objet courant comme paramètre implicite. Elle n’a pas besoin d’un objet pour être invoquée. Le mot clé **STATIC** précède chaque méthode de classe.

**Exemple: Ajouter une méthode dans le type Trectangle qui permet de tester est ce que deux rectangles ont la même surface ou pas.**

**Etape 1**: **Alter type** Trectangle add **static** function **memeSurface(tri1 Trectangle , tri2 Trectangle) return boolean cascade**;

# Les méthodes statiques

Create or replace type body Trectangle as

MEMBER FUNCTION surface RETURN Float IS……END;

MEMBER FUNCTION périmètre  RETURN  Float IS…..END;

MEMBER PROCEDURE afficher  is……..End ;

MEMBER PROCEDURE **SETLARGEUR** ( LARGEUR float) is……End ;

MEMBER FUNCTION **GETLARGEUR**  RETURN  Float IS……END;

```
STATIC FUNCTION  memeSurface(tri1 Trectangle , tri2 Trectangle) return boolean is
BEGIN
IF tri1.surface()=tri2.surface() THEN  return true;
ELSE  return false ;
END;

END;
```

# Les méthodes (Invocation d'une méthode statique)

```
DECLARE
  rect1 TRECTANGLE;
  rect2 TRECTANGLE;
  n boolean;
BEGIN
  SELECT VALUE(r) INTO rect1 FROM RECTANGLES r WHERE r.largeur  = 6;
  SELECT VALUE(r) INTO rect2 FROM RECTANGLES r WHERE r.largeur  = 3.3;
  n:=TRECTANGLE.memeSurface(rect1,rect2);
  IF n=true  THEN
  DBMS_OUTPUT.PUT_LINE('true');
  ELSE   DBMS_OUTPUT.PUT_LINE('false');
  end if;
END;
```

```
Procédure PL/SQL terminée.

false
```

# Surcharge et Redéfinition

# Surcharge et Redéfinition

**Surcharge (overloading):** Possibilité de définir plusieurs codes pour une même opération d'une classe, le code sera sélectionné en fonction du type de paramètre lors de l'appel.

**Exemple typique sur le surcharge :** l'opération **+**

+(entier, entier) ➔ entier
+(réel, réel) ➔ réel
+(chaine, chaine) ➔ chaine

**Redéfinition(overriding) :** Spécification d'une opération existante dans une super-classe au niveau d'une sous-classe, avec une implémentation différente.

# Surcharge et Redéfinition

N.B: Il est possible de ne pas définir le code de l'opération au niveau de la classe et de le définir au niveau de la sous-classe.