## Les propriétés

Si une propriété ressemble à une variable, elle n’en est pas une dans la mesure où l’affectation d’une valeur à une propriété peut déclencher l’exécution d’une méthode. Cette dernière, en plus d’accéder au champ visé, pourra effectuer des traitements particuliers nécessaires au programme.

## La POO en action

### La variable Self

### Parent et propriétaire

On a vu que toute classe a un parent, ne serait-ce que *TObject* et qu’elle hérite de ce parent. La notion de parenté fait par conséquent référence à la structure d’une classe et aux méthodes pour agir sur cette structure.

La notion de *propriétaire* d’un objet ne s’applique que pour les classes de la LCL qui descendent de la classe *TPersistent* qui définit une propriété *Owner*. C’est cette dernière qui sera renseignée pour désigner la classe responsable

Lorsque vous utilisez des composants dont le propriétaire est défini (ce qui est le cas dès que vous vous servez des fiches et que vous y déposez les composants nécessaires à votre application), c’est ce propriétaire qui s’occupe d’allouer et de libérer la mémoire de manière transparente. Vous n’avez par conséquent pas à vous en charger.

Voilà pourquoi les composants présents sur la palette qui sont des classes particulières n’ont pas besoin d’être créés par vos soins. La LCL s’en charge à partir du propriétaire qui est à la base un descendant de *TForm*, c’est-à-dire une autre classe !

En revanche, dès que vous manipulez des objets

var

MonAnimal : TAnimal ;

begin

MonAnimal  := TAnimal.Create ; // on crée la liste de chaînes

try // on protège le code de manipulation pour être sûr de préserver les ressources

// ici le traitement voulu…

MonAnimal.Dormir ;

finally

// en interne, la méthode *Free* appelle le destructeur *Destroy*

MonAnimal .Free ; // les ressources seront toujours libérées

end ;

end ;

### Où est utilisée la POO dans Lazarus ?

### Quand utiliser la POO… et quand ne pas l’utiliser