UML

Unified Modeling Language

UML C'est quoi ?

Et bien ..

UML, c'est l'acronyme anglais pour « Unified Modeling Language ».

On le traduit par « Langage de modélisation unifié ».

La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter.

UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc..

Réaliser ces diagrammes revient donc à **modéliser les besoins** du logiciel à développer.

UML définit 9 types de diagrammes dans deux catégories de vues, les vues statiques et les vues dynamiques.

Vues statiques:

- Les diagrammes de cas d'utilisation décrivent le comportement et les fonctions d'un système du point de vue de l'utilisateur
- Les diagrammes de classes décrivent la structure statique, les types et les relations des ensembles d'objets
- Les diagrammes d'objets décrivent les objets d 'un système et leurs relations
- Les diagrammes de composants décrivent les composants physiques et l'architecture interne d'un logiciel
- Les diagrammes de déploiement décrivent la répartition des programmes exécutables sur les différents matériels

Vues dynamiques :

- Les diagrammes de collaboration décrivent les messages entre objets (liens et interactions)
- Les diagrammes d'états-transitions décrivent les différents états d'un objet
- Les diagrammes d'activités décrivent les comportements d'une opération (en termes d'actions)
- Les diagrammes de séquence décrivent de manière temporelle les interactions entre objets et acteur

Pour ce cours, nous ne couvrirons que les diagrammes de classes

Le langage UML a été créé sous forme de modèle standardisé pour décrire une approche de la programmation orientée objet.

Comme les classes sont les composantes des objets, les diagrammes de classes sont les composantes de l'UML.

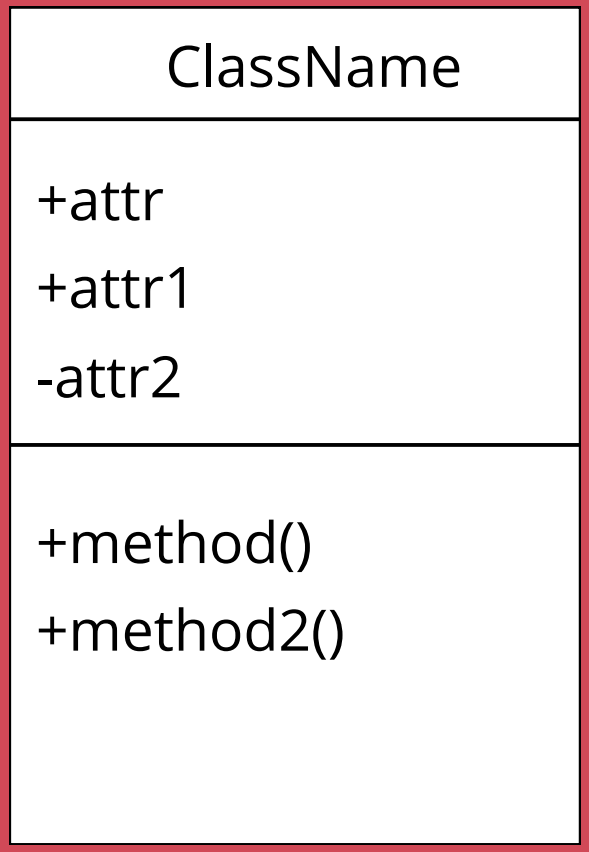
Les divers éléments d'un diagramme de classes peuvent représenter les classes qui seront effectivement programmées, les principaux objets ou les interactions entre classes et objets.

La forme de la classe à proprement parler se compose d'un rectangle à trois lignes.

- La ligne supérieure contient le nom de la classe
- celle du milieu affiche les attributs de la classe
- la ligne inférieure exprime les méthodes ou les opérations que la classe est susceptible d'utiliser.

Les classes et sous-classes sont regroupées pour illustrer la relation statique entre chaque objet.

Une classe en UML



Modificateurs d'accès

Toutes les classes ont des niveaux d'accès différents, en fonction du modificateur d'accès (indicateur de visibilité).

Voici les niveaux d'accès existants et les symboles qui leur sont associés :

- -Public (+)
- -Privé (-)
- -Protégé (#)
- -Paquetage (~)
- -Dérivé (/)
- -Statique (souligné)

Les différents types d’associations entre deux classes :

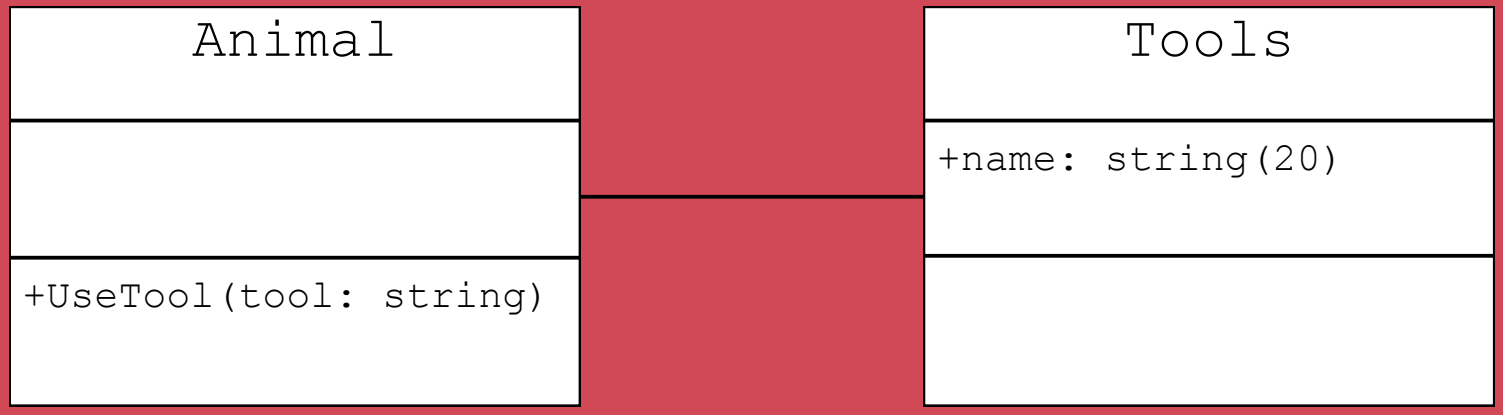
L’association simple:

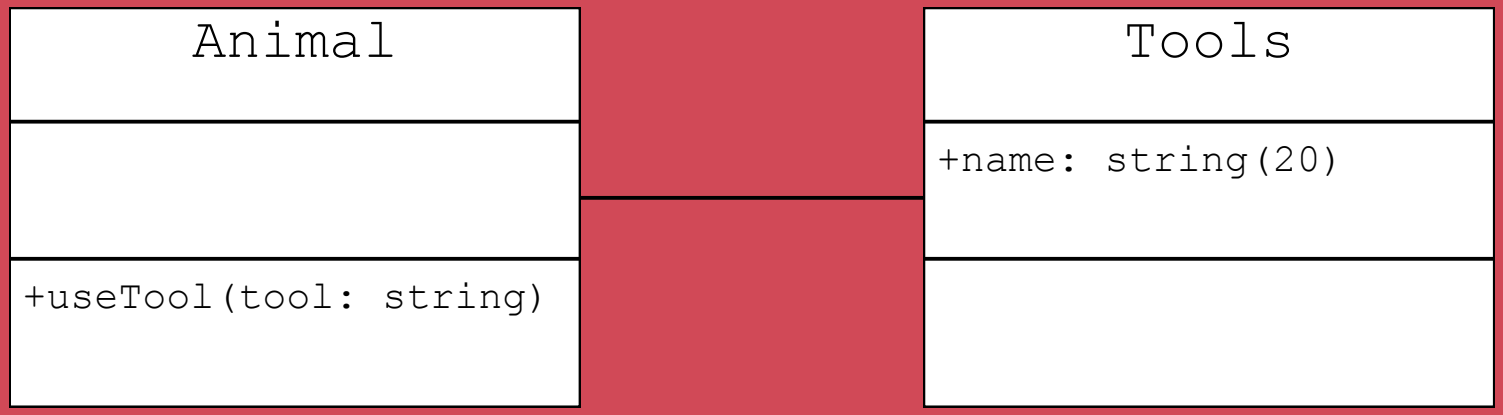
Exemple: Un Animal utilise/crée un Outil.

L’outil a besoin de l’animal pour être créer.

L’animal a besoin d’utiliser l’outil.

il y a une relation de co-dépendance.



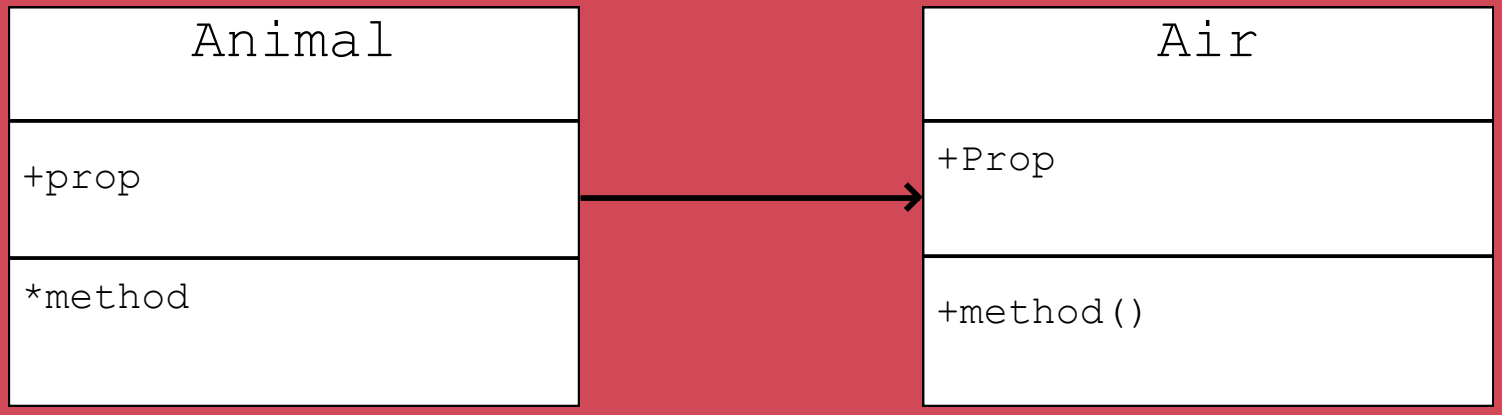


Représente

```php
class Animal {
     public function useTool(string $tool) {
       echo "J'utilise un " . $tool;
     }
 }

class Tools  {
   public $name;

   constructor($name) {
      $this->name = $name;
   }
}

$caillou = new Tool("caillou");
$singe = new Animal();

$singe->useTool($caillou->name);
```

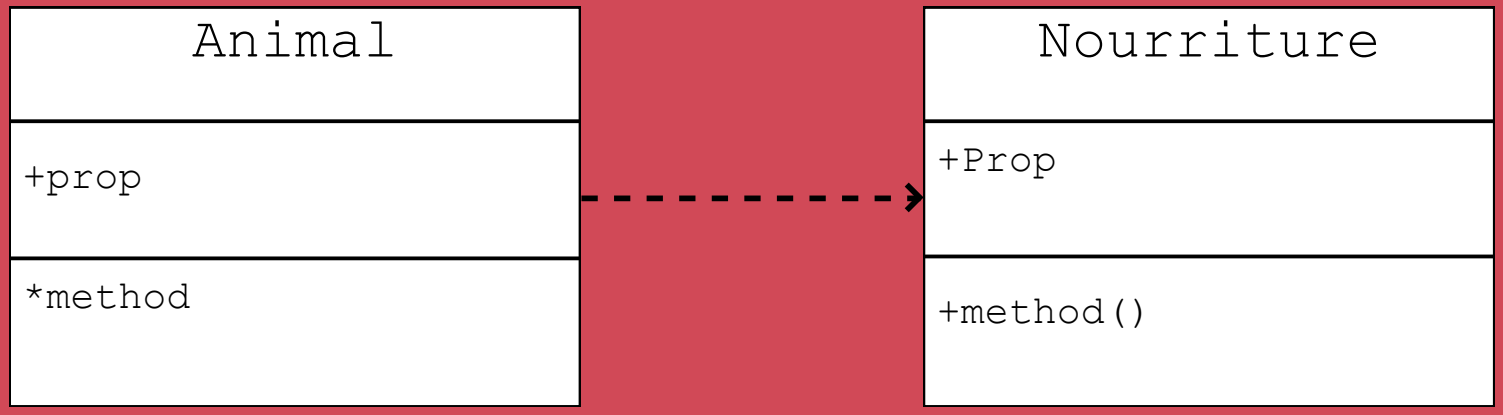
L’association directe:

un Animal respire de l’Air. L’animal utilise l’air sans interruption. L’animal ne vit pas sans air, l’air peut vivre sans animal.



L’association temporaire:

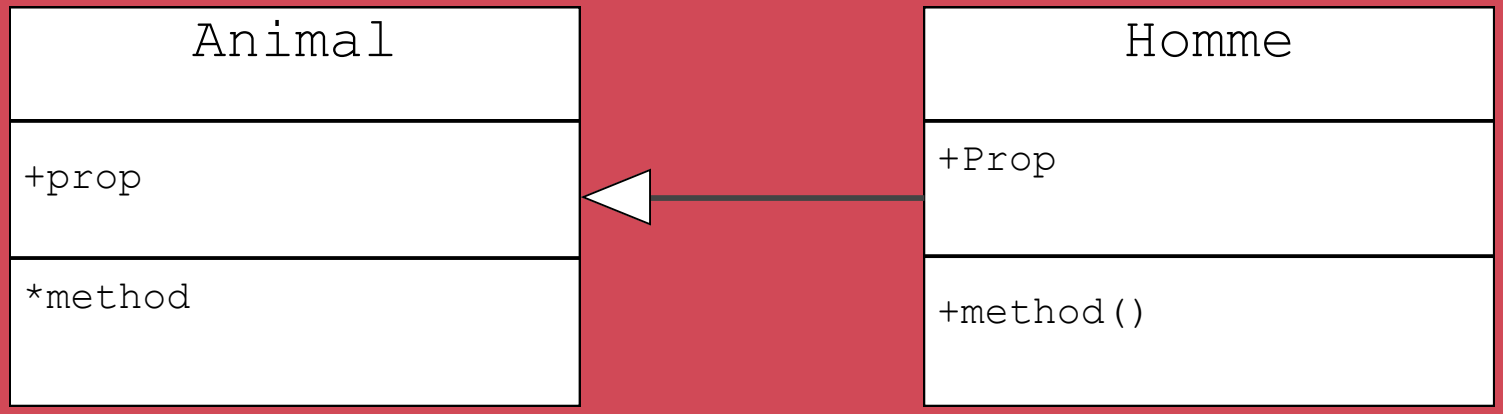
un Animal mange de la nourriture. L’animal utilise ponctuellement la nourriture. L’animal ne vit pas sans nourriture, la nourriture vit sans l’animal.



L'association d’héritage:

L’Homme est un Animal.

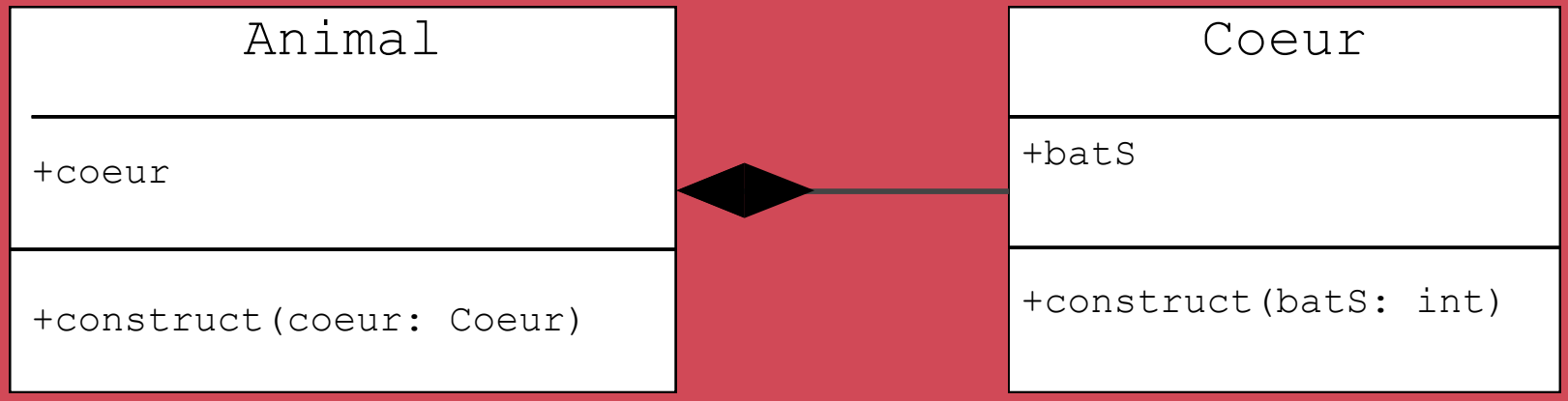
Tout les animaux ne sont pas des hommes. Tout les hommes ont les mêmes propriétés et méthodes que les animaux (squelette, respiration(),...). Tout les animaux n’ont pas les mêmes propriétés que l’homme (pouceOppose, marcherDebout()).

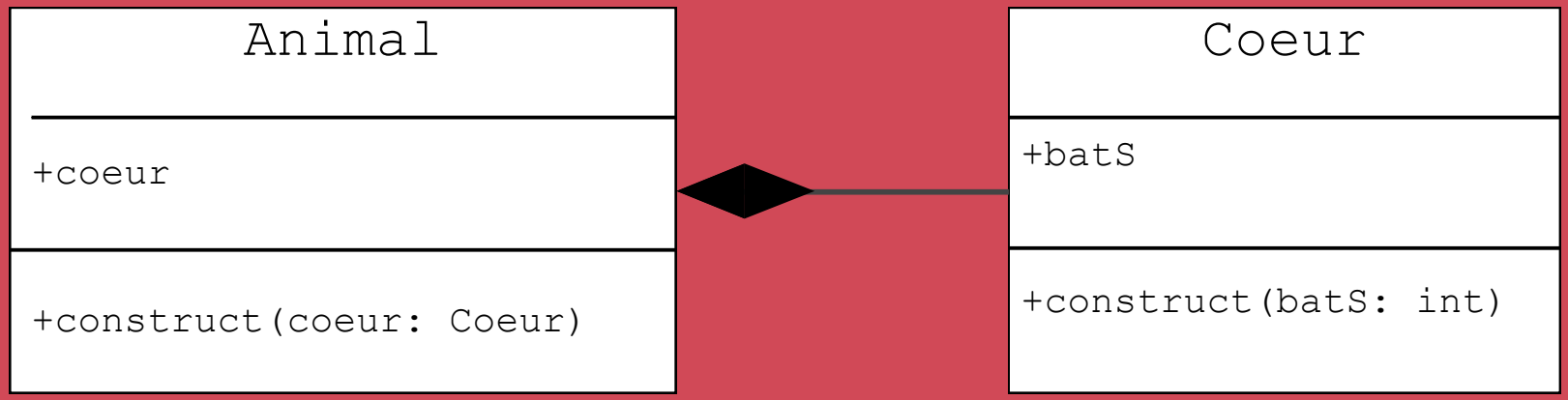


L’association de composition:

un Animal a un Coeur.

L’un fait partie de l’autre et l’un ne vit pas sans l’autre.





Représente

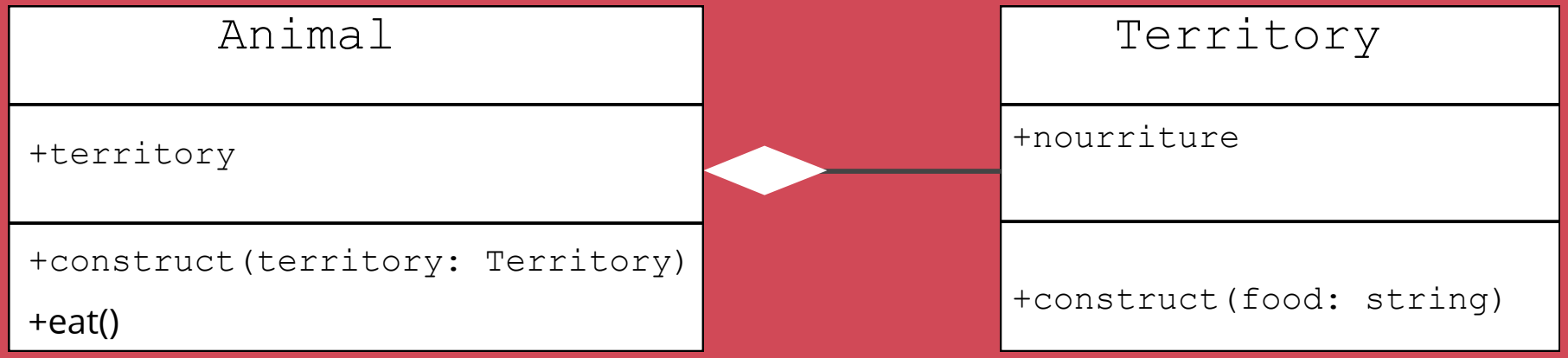
```php
class Animal {
    public $coeur;

    public function __construct (Coeur $coeur) {
      $this->coeur = $coeur;
    }
}

class Coeur {
    public $batS;

    public function __construct (int $batS) {
         $this->batS = $batS;
    }
}

$coeur = new Coeur(90);
$humain = new Animal($coeur);

echo $humain->coeur->batS;
```

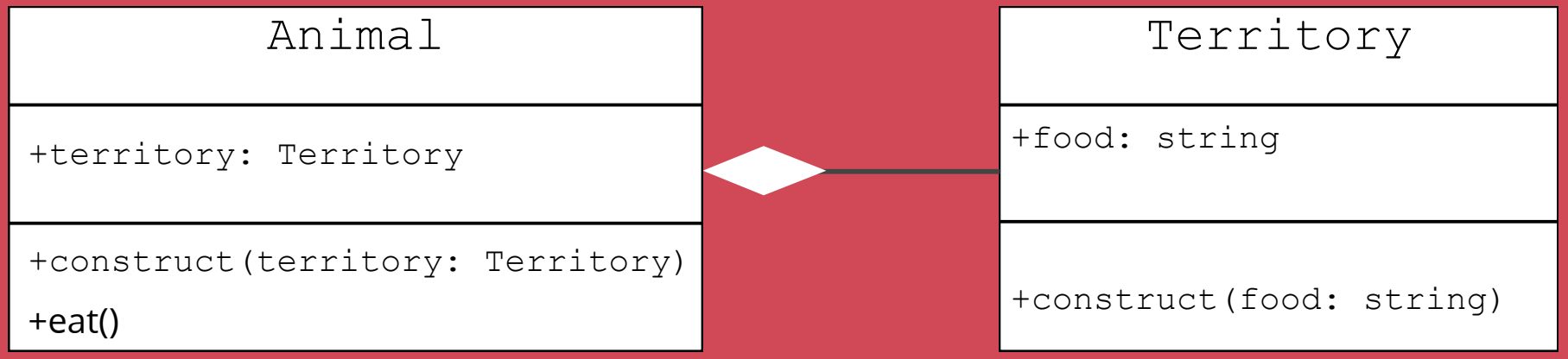
L’association d'agrégation:

un Animal a un Territoire.

Les deux sont indépendants l’un de l’autre. L’animal peut survivre hors de son territoire et inversement.

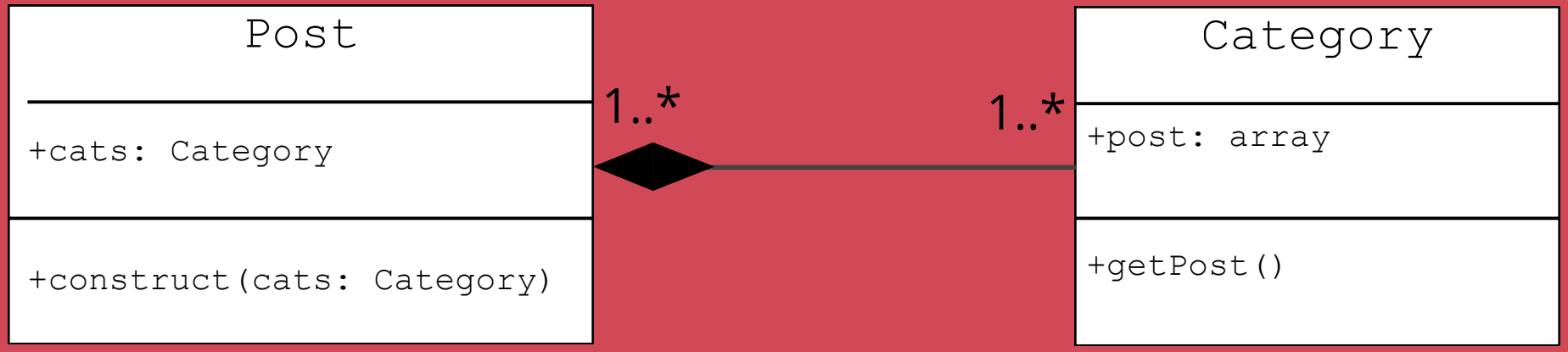
Le Territoire a des propriétés et des méthodes qui ne sont pas liés à l’animal et inversement.






```php
class Animal {
   public $territory;

   public function __construct(Territory $territory) {
     $this->territory = $territory;
   }

   public function eat() {
       echo 'je mange des ' . $this->territory->food
   }

}

class Territory {
   public $food;

   public function __construct (String $food) {
     $this->food = $food
   }
}

$verger = new Territory('pommes');
$me = new Animal($verger);

$me->eat();
```

Pour chaque type d’association on peut définir une quantité de l’un vers l’autre.



Multiplicité	**Signification**
1	Exactement 1
5	Exactement 5
*	Plusieurs incluant la possibilité d’aucun
0..*	Idem
0..1	Au plus un, ce qui signifie également optionnellement
1..*	Au moins un
1..5	Entre un et cinq

Un autre exemple