



Conception en UML, Architecture n-tiers, par l'exemple

Utilisation de php 5, Mysql, Html, css, ...

Inspiré de UML2 par la pratique

M. Blay-Fornarino

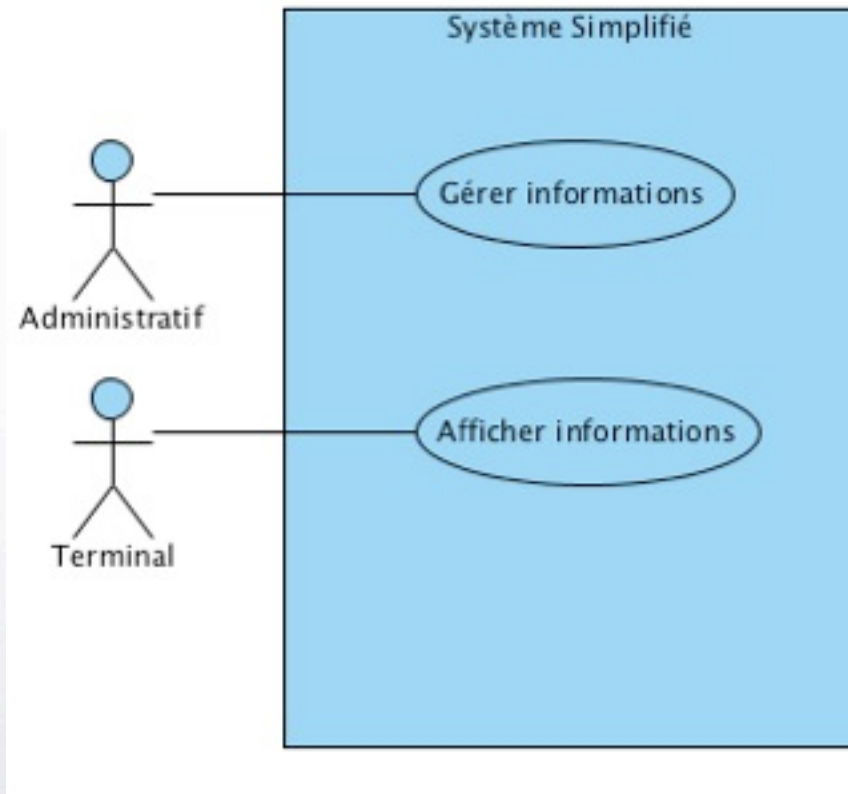
Les codes sont disponibles sur le site web

Bibliographie

- ❧ «Why MVC is not an application architecture» Stefan Pribsch, the PHP.cc ZendCon 2010
- ❧ Developing Web Applications with PHP, RAD for the World Wide Web,

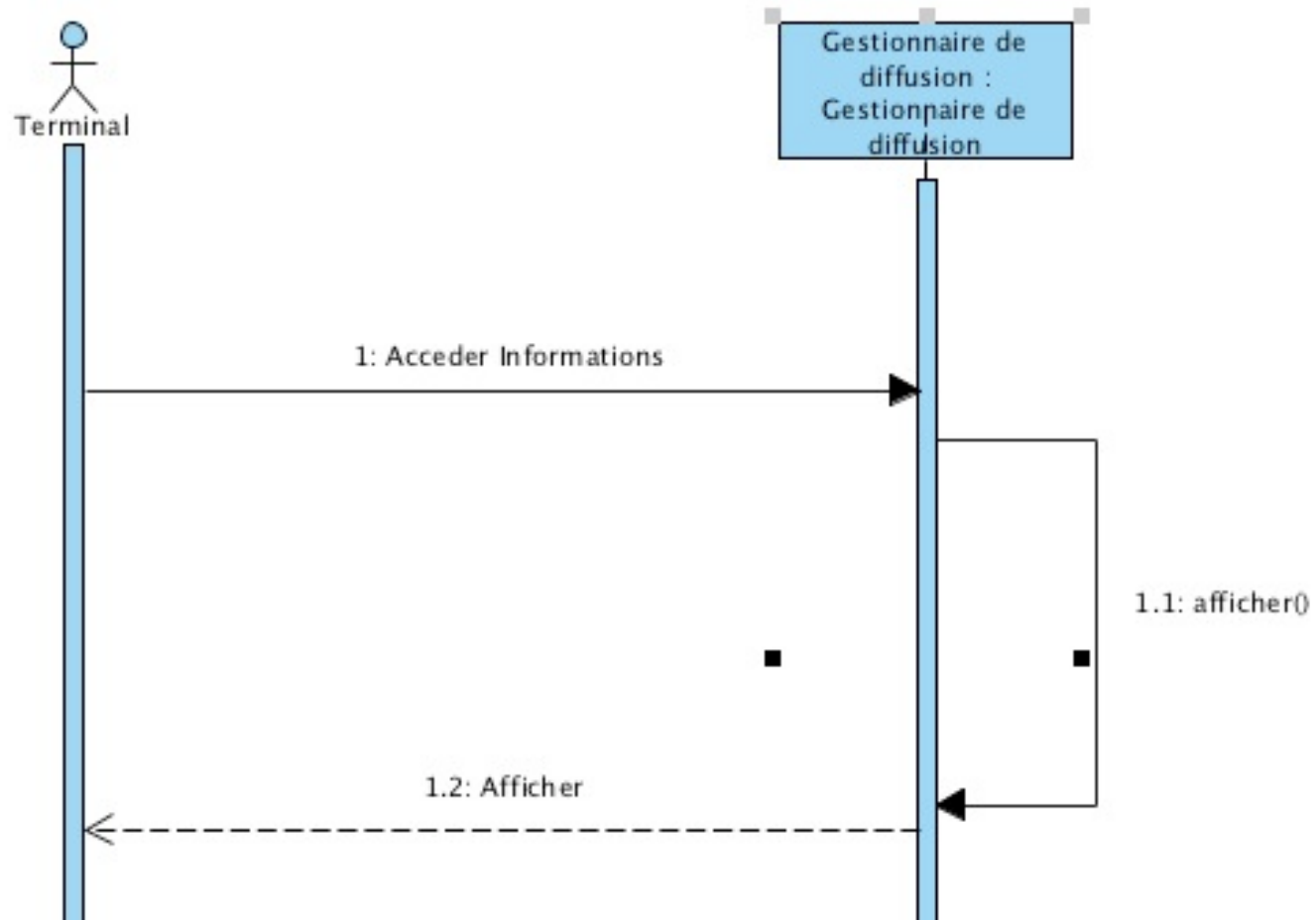


Diagramme de Use-cases



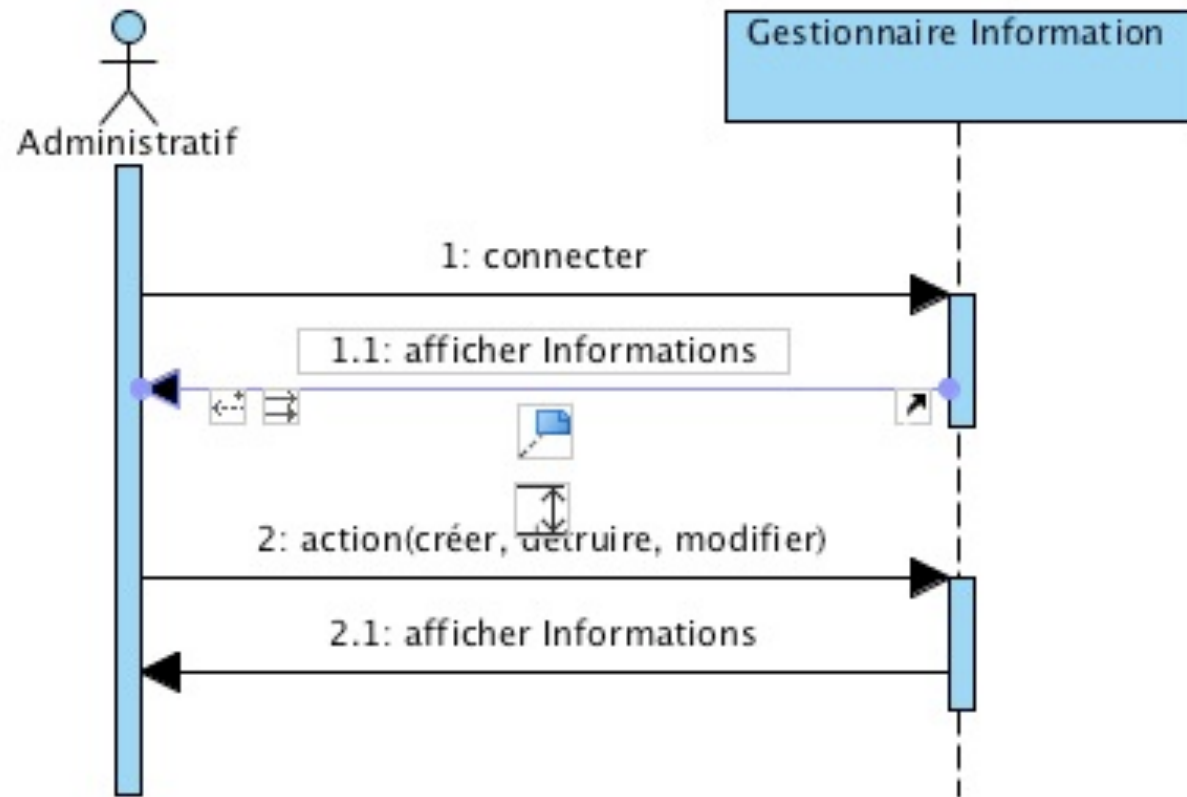


Afficher Informations : niveau Analyse





Gérer Informations : niveau Analyse (0)





Choix d'Architecture

Gestion des Informations

Présentation

Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



Choix d'Architecture

Présentation

Gestion des Informations

Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

[Modifier](#) [Détruire](#)

Titre de l'information

[Créer un nouvelle information](#)

Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



Choix d'Architecture

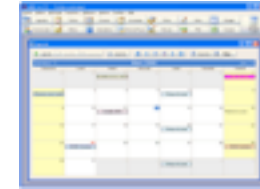
Présentation

Gestion des Informations

Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Titre de l'information



Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



Choix d'Architecture

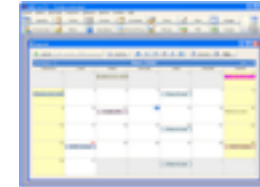
Présentation

Gestion des Informations

Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Titre de l'information



Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



ORACLE



Choix d'Architecture

Gestion des Informations

Liste des informations

- nuit de l'Info [2010-11-12 13:40:18,3]
- Devist [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Titre de l'information

Présentation

Logique applicative

Stockage

clickOn()

IHM

créer Information

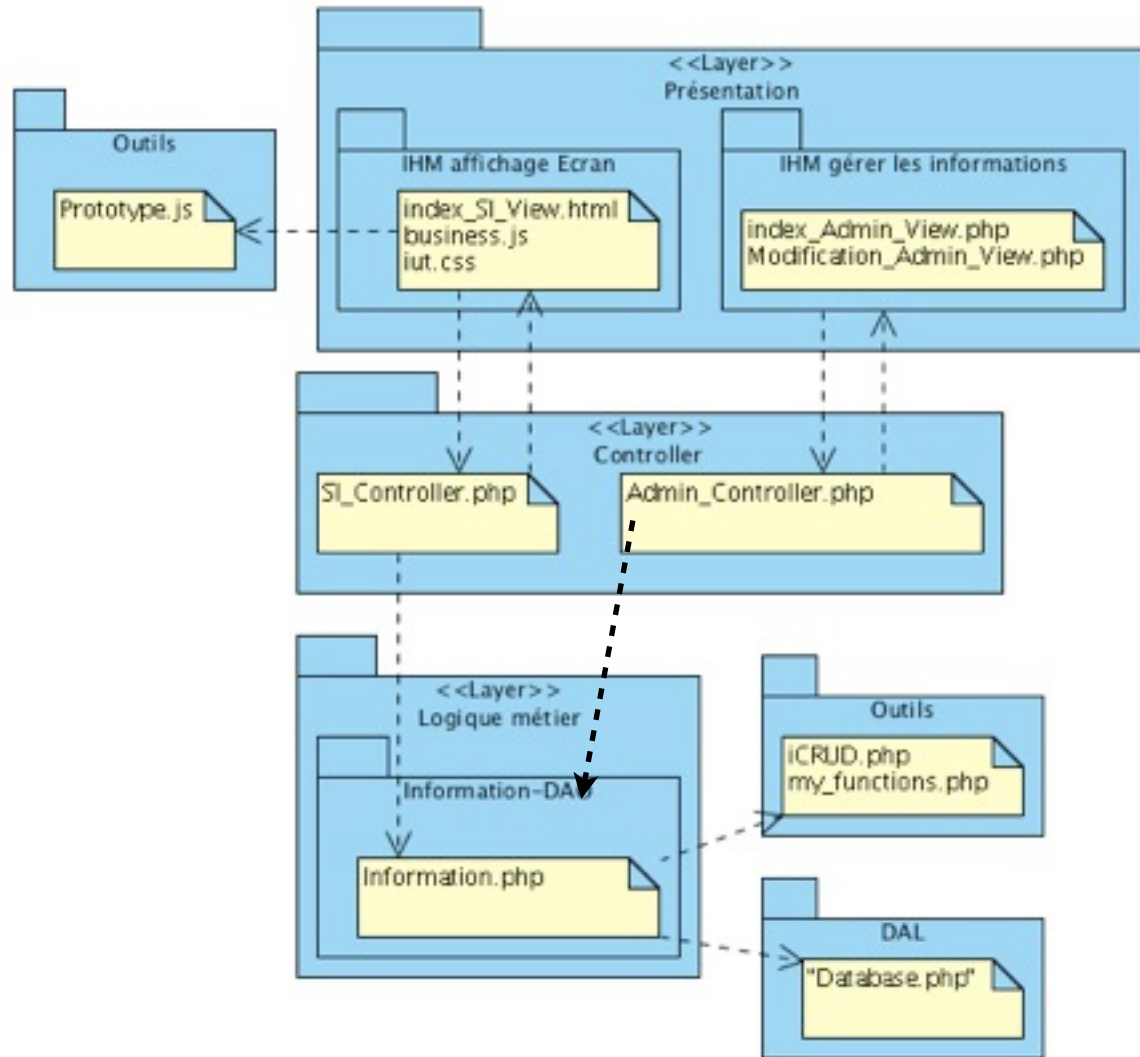
Contrôleur

Gérer les informations : métier

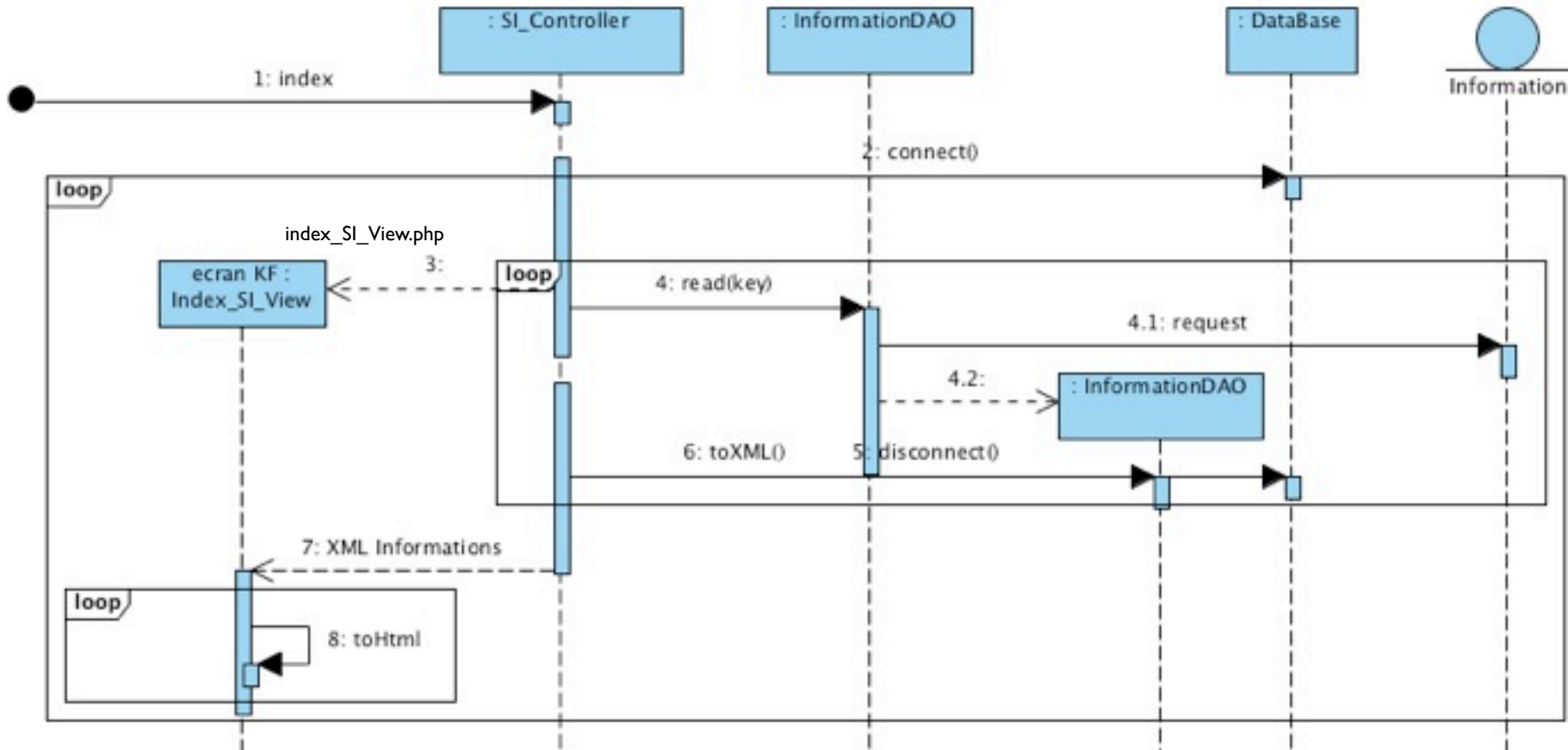
```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



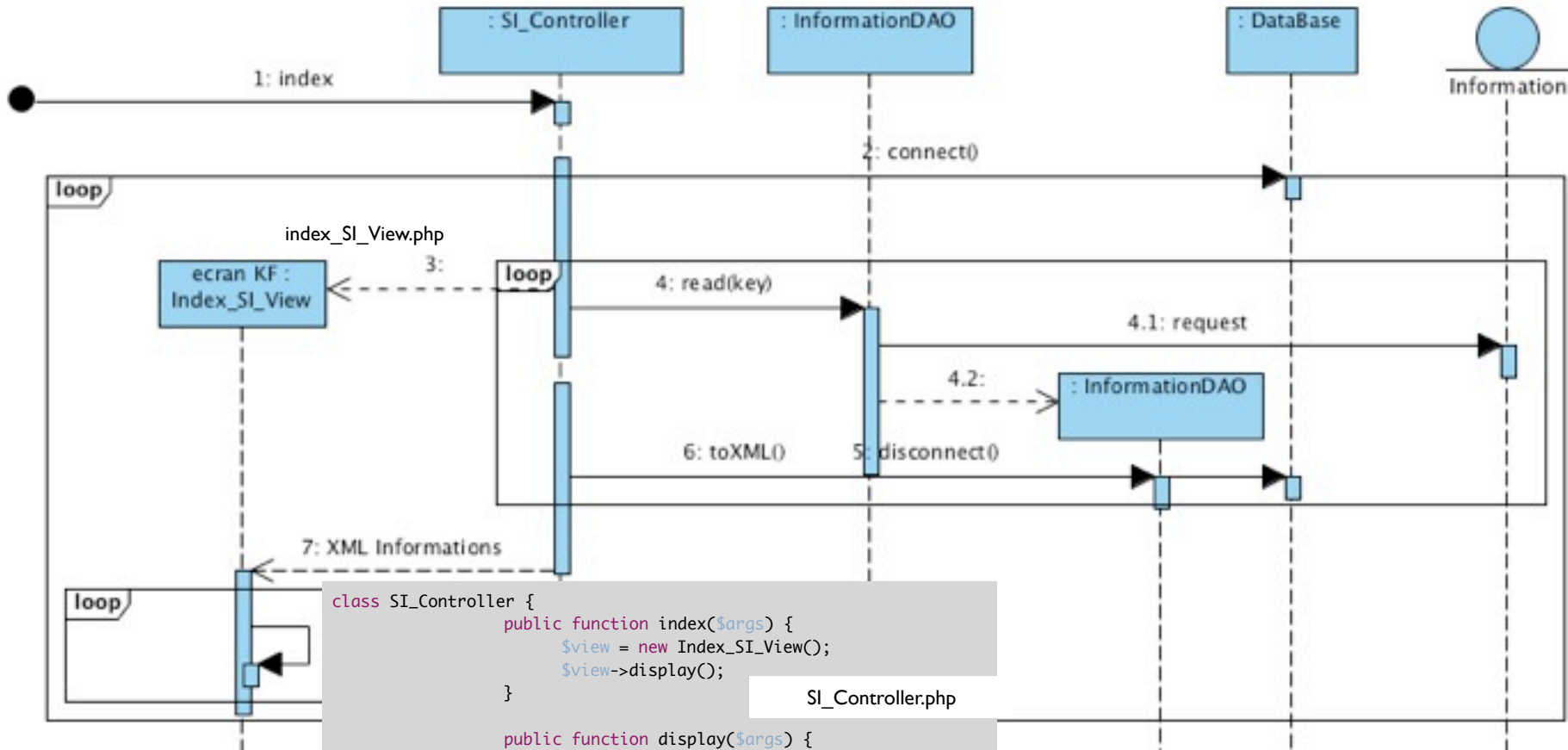

Architecture en couches & Fichiers



Afficher Informations : niveau Conception



Afficher Informations : niveau Conception



```

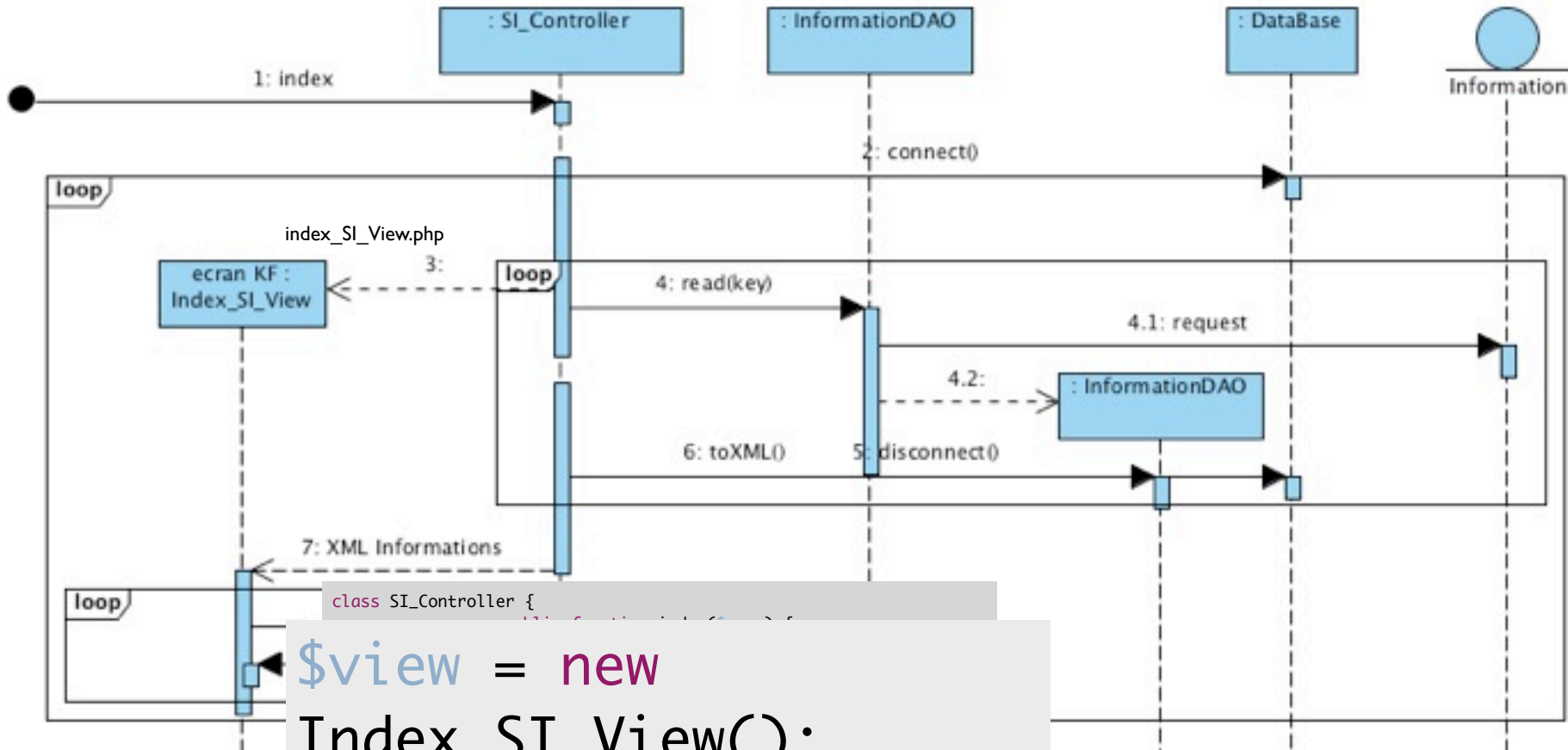
class SI_Controller {
    public function index($args) {
        $view = new Index_SI_View();
        $view->display();
    }

    public function display($args) {
        header('Content-type: text/xml');
        $res='<?xml version="1.0"?><data>';
        $infos = Information::findAll();
        foreach($infos as $tmpInformation) {
            $res = $res.$tmpInformation->toXML();
        }
        $res=$res.'</data>';

        echo $res;
    }
}
    
```

SI_Controller.php

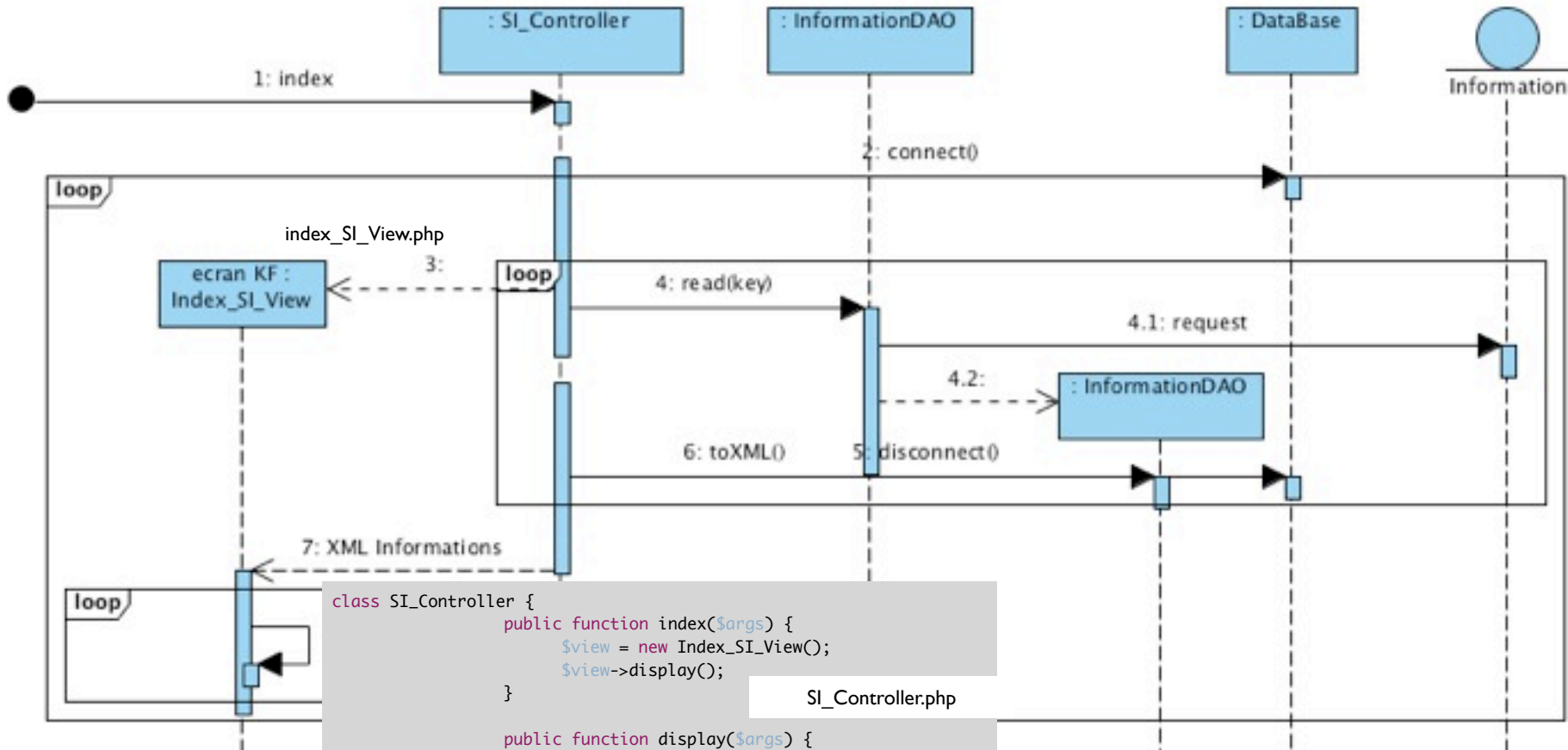
Afficher Informations : niveau Conception



```
class SI_Controller {  
    $view = new  
    Index_SI_View();  
    $view->display
```

```
    $res = $res.$xml2xml($xml);  
    $res=$res.'</data>';  
    echo $res;  
}
```


Afficher Informations : niveau Conception



```

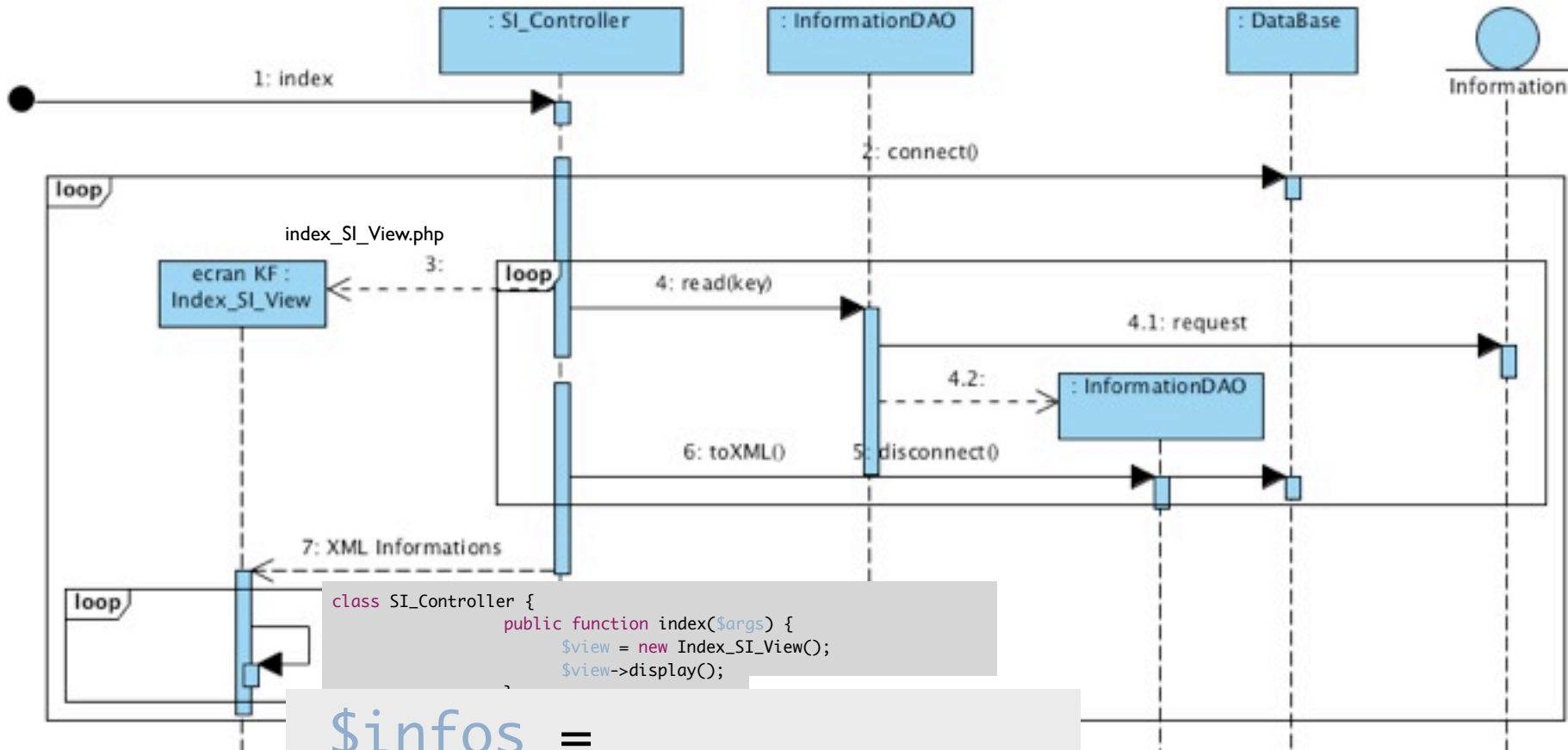
class SI_Controller {
    public function index($args) {
        $view = new Index_SI_View();
        $view->display();
    }

    public function display($args) {
        header('Content-type: text/xml');
        $res='<?xml version="1.0"?><data>';
        $infos = Information::findAll();
        foreach($infos as $tmpInformation) {
            $res = $res.$tmpInformation->toXML() ;
        }
        $res=$res.'</data>';

        echo $res;
    }
}
    
```

SI_Controller.php

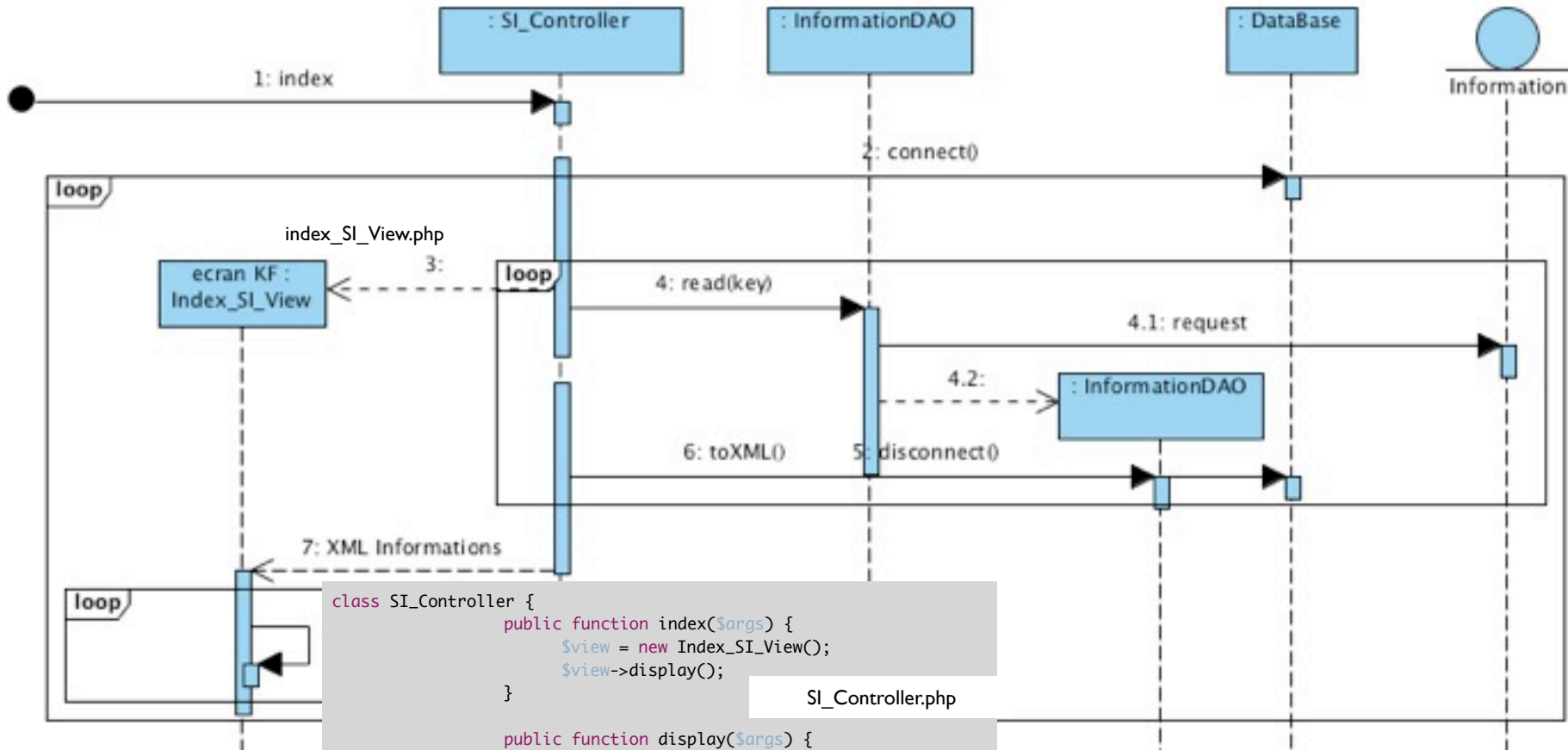
Afficher Informations : niveau Conception



```
$infos =  
Information::findAll();
```

```
$res = $res.$tmpinformation->toXML() ;  
}  
$res=$res."</data>";  
  
echo $res;  
}
```

Afficher Informations : niveau Conception



```

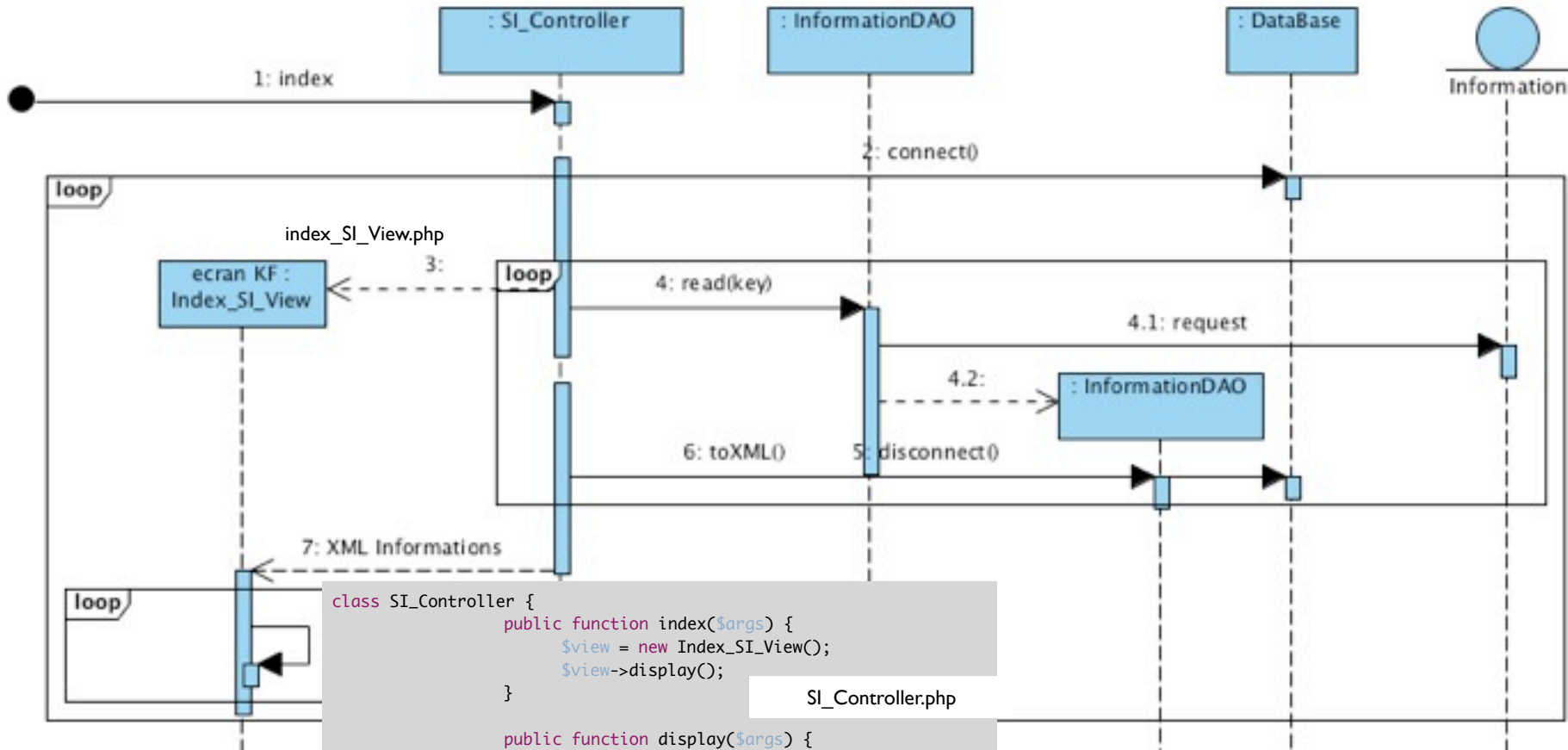
class SI_Controller {
    public function index($args) {
        $view = new Index_SI_View();
        $view->display();
    }

    public function display($args) {
        header('Content-type: text/xml');
        $res='<?xml version="1.0"?><data>';
        $infos = Information::findAll();
        foreach($infos as $tmpInformation) {
            $res = $res.$tmpInformation->toXML() ;
        }
        $res=$res.'</data>';

        echo $res;
    }
}
    
```

SI_Controller.php

Afficher Informations : niveau Conception



```

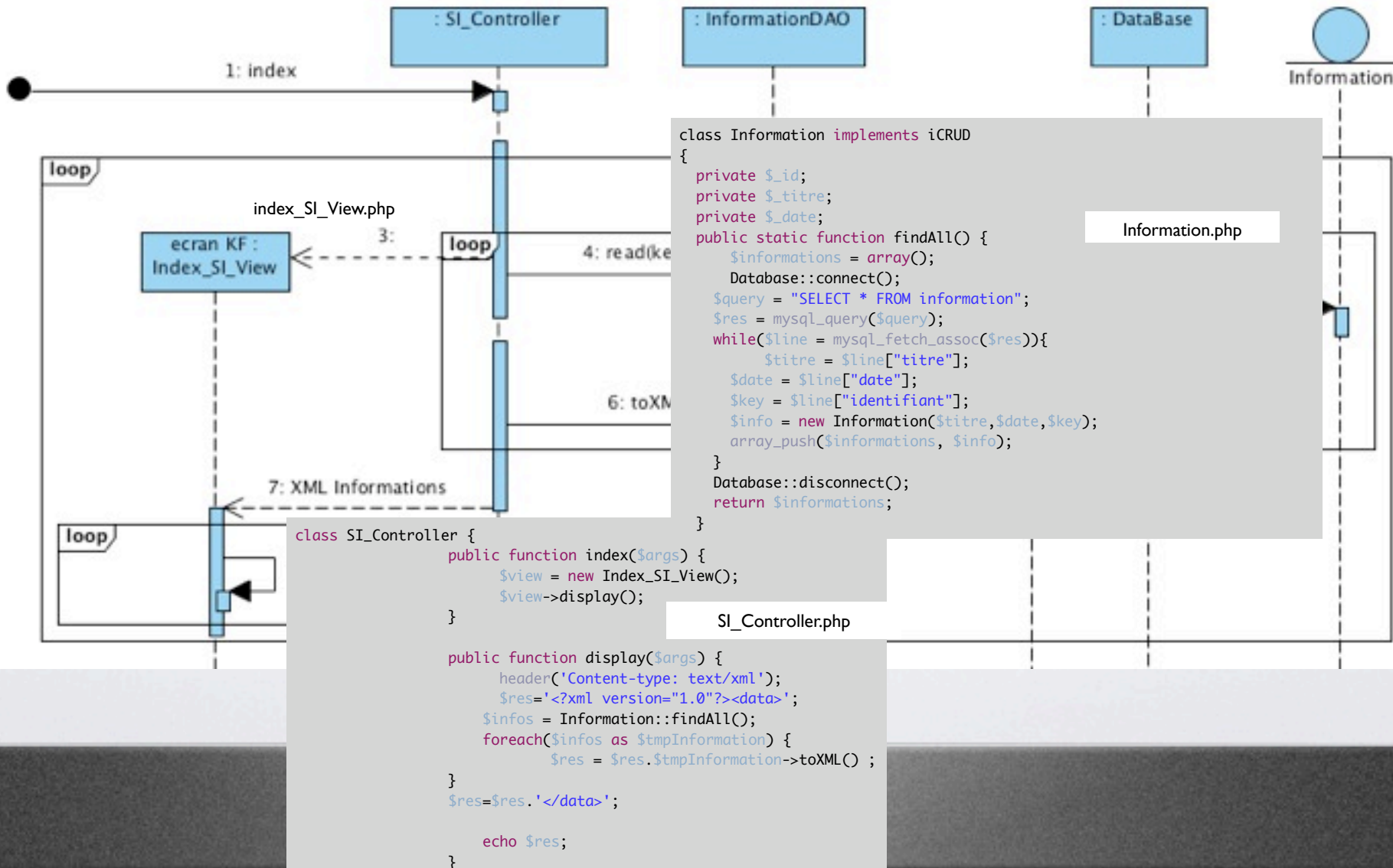
class SI_Controller {
    public function index($args) {
        $view = new Index_SI_View();
        $view->display();
    }

    public function display($args) {
        header('Content-type: text/xml');
        $res='<?xml version="1.0"?><data>';
        $infos = Information::findAll();
        foreach($infos as $tmpInformation) {
            $res = $res.$tmpInformation->toXML();
        }
        $res=$res.'</data>';

        echo $res;
    }
}
    
```

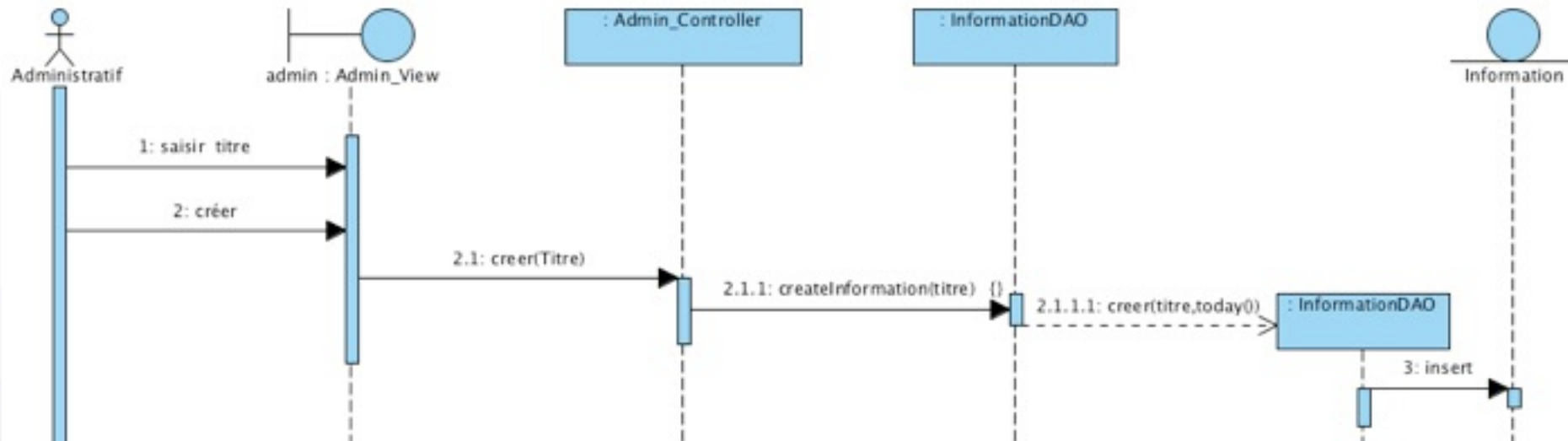
SI_Controller.php

Afficher Informations : niveau Conception



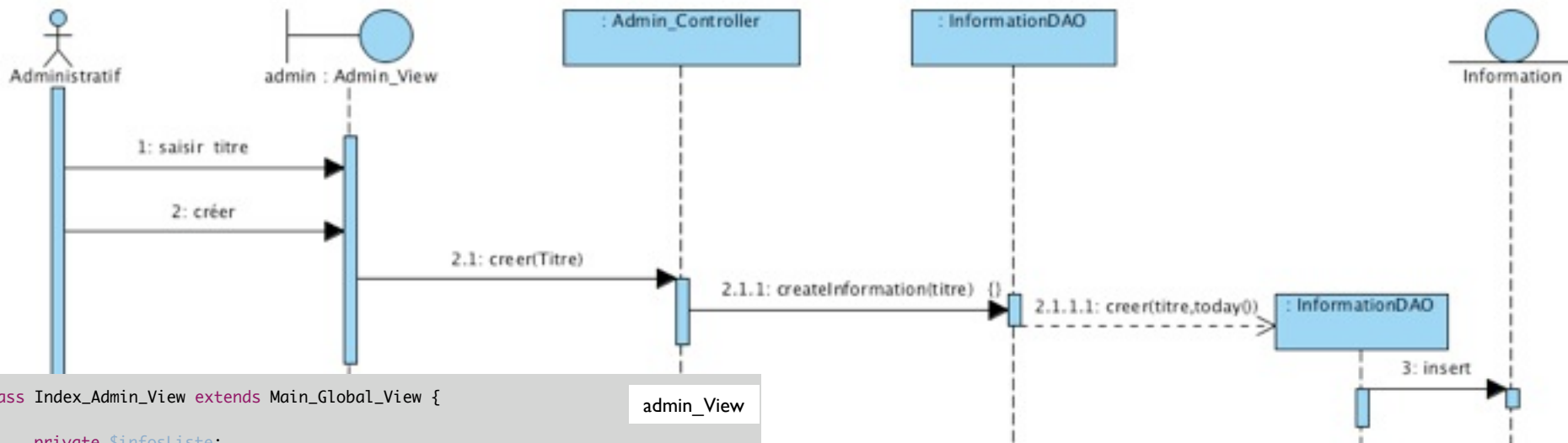


Créer Information : niveau Conception





Créer Information : niveau Conception



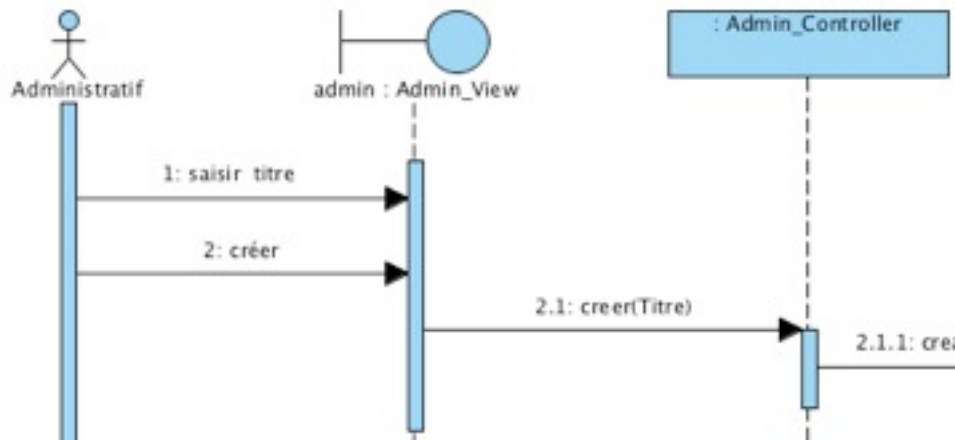
```

class Index_Admin_View extends Main_Global_View {
    private $infosListe;

    public function Index_Admin_View($args) {
        $this->infosListe = $args['infosListe'];
    }...
<h1>Gestion des Informations</h1>
<h2> Liste des informations </h2>
<form id="infoModifForm" name="infoModifForm" method="post" action="."/ >
<p>
  
```




Créer Information : niveau Conception



```

class Admin_Controller {
    public function index($args) {
        $args['infosListe'] = Information::findAll();
        $view = new Index_Admin_View($args);
        $view->display();
    }
}
Admin_Controller.php

public function confirmer_modifier($args) {
    $key = $_POST["key"];
    $newTitre = $_POST["NouveauTitre"];
    $info = Information::read($key);
    $info->setTitre($newTitre);
    $info->update();

    $args['infosListe'] = Information::findAll();

    $view = new Index_Admin_View($args);
    $view->display();
}

public function create($args) {
    $titre = $_POST["Titre"];
    $info = new Information($titre, $this->today());
    $info->create();

    $args['infosListe'] = Information::findAll();

    $view = new Index_Admin_View($args);
    $view->display();
}
    
```

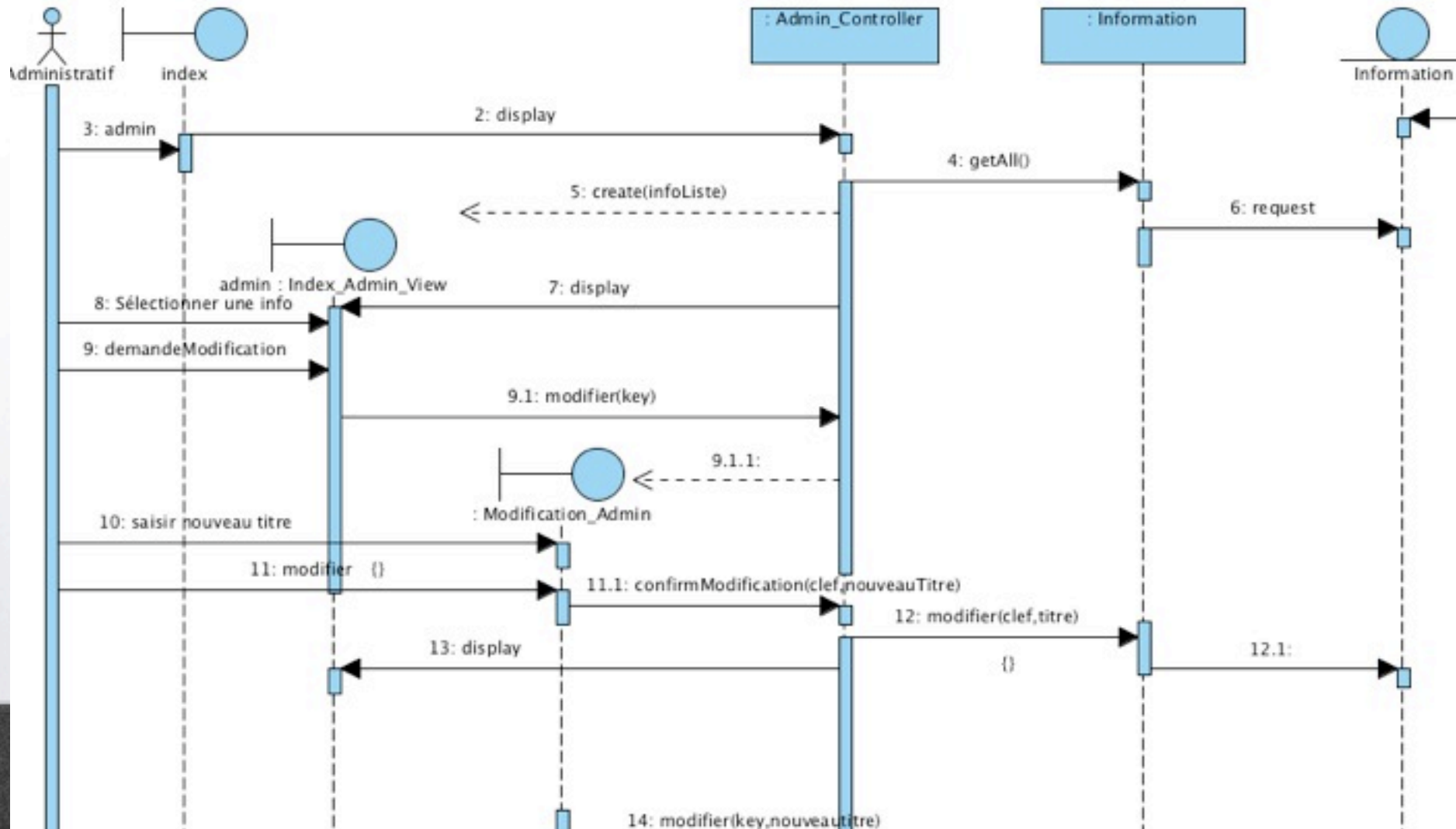
```


class Index_Admin_View extends Main_Global_View {
    private $infosListe;

    public function Index_Admin_View($args) {
        $this->infosListe = $args['infosListe'];
    }...
}
admin_View

<h1>Gestion des Informations</h1>
<h2> Liste des informations </h2>
<form id="infoModifForm" name="infoModifForm" method="post" action="."/ >
<p>
    
```

modifier Information : niveau Conception

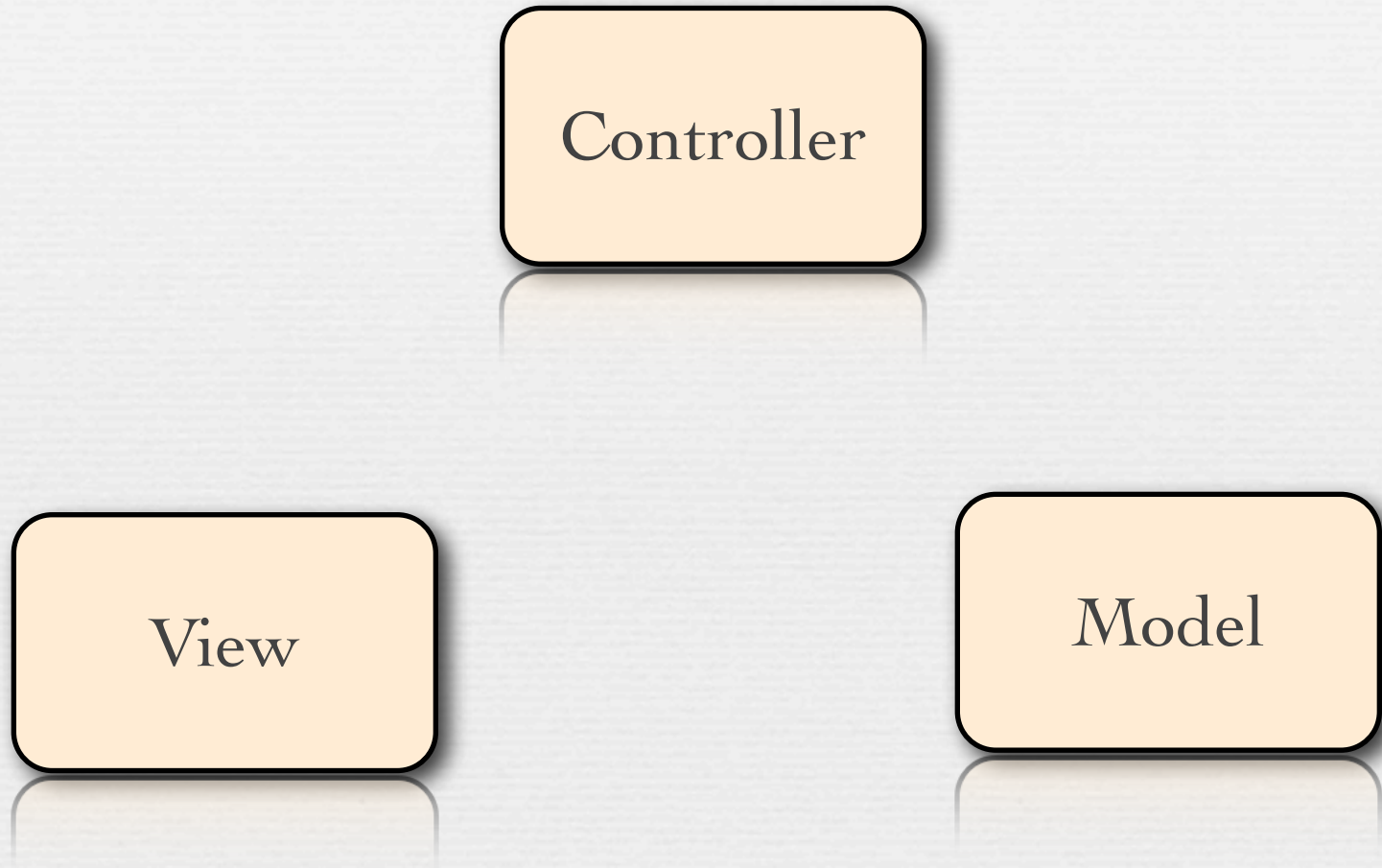


A photograph of a wooden fence with decorative posts, set in a yard with fallen autumn leaves. The fence is made of vertical wooden planks and has several posts with decorative caps. The ground is covered with green grass and many brown, fallen leaves. In the background, there are bare trees and a utility pole.

SEPARATIONS :

Données, Interactions et Visualisation, Contrôles

Modèle-Vue-Contrôleur (MVC)



Modèle-Vue-Contrôleur (MVC)

Controller

View

Model

La vue: présentée
à l'utilisateur

Modèle-Vue-Contrôleur (MVC)

Controller

View

Model

La vue: présentée
à l'utilisateur

Le modèle: les données
indépendantes

Modèle-Vue-Contrôleur (MVC)

Controller

contrôleur:
chef
d'orchestre

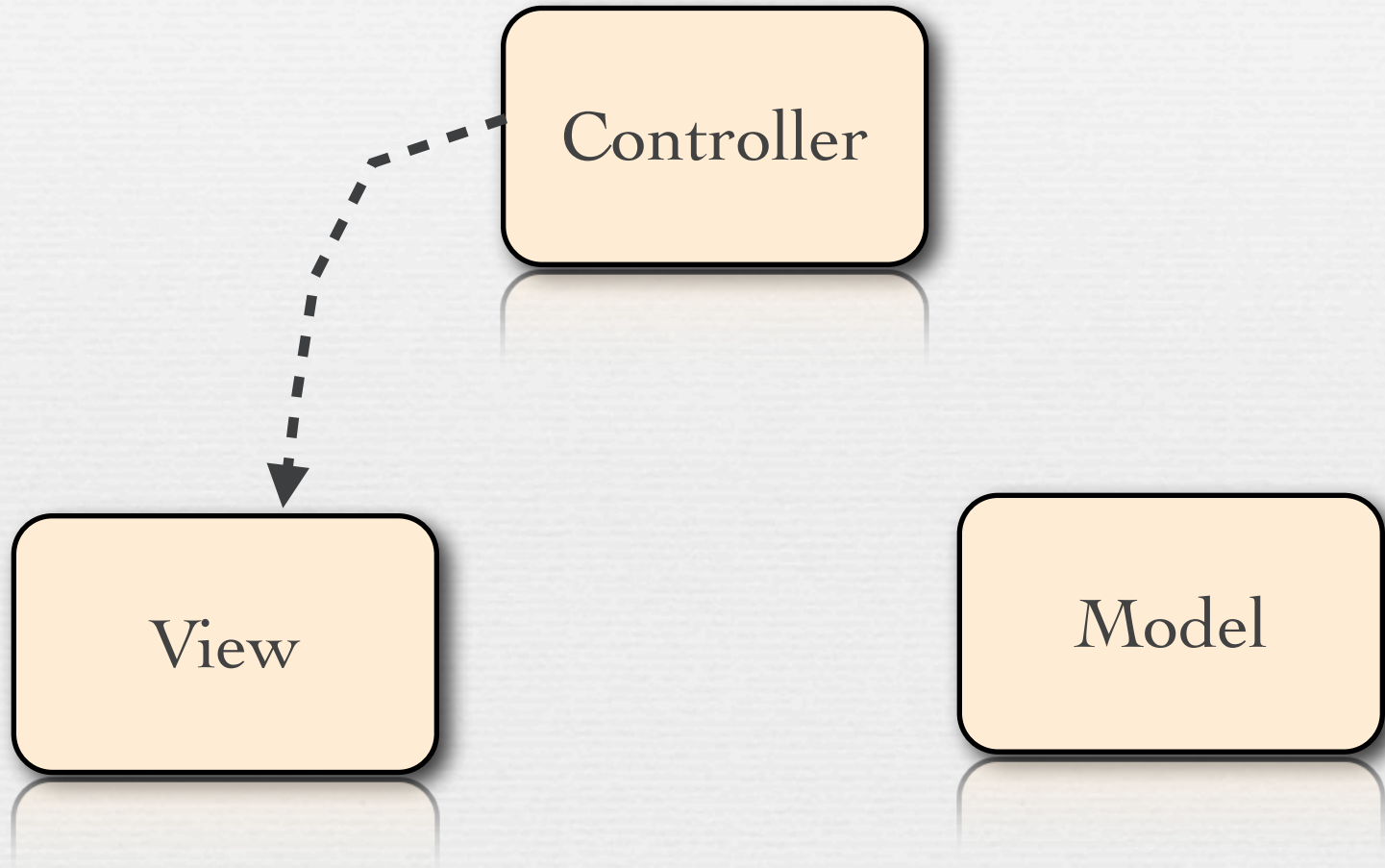
View

La vue: présentée
à l'utilisateur

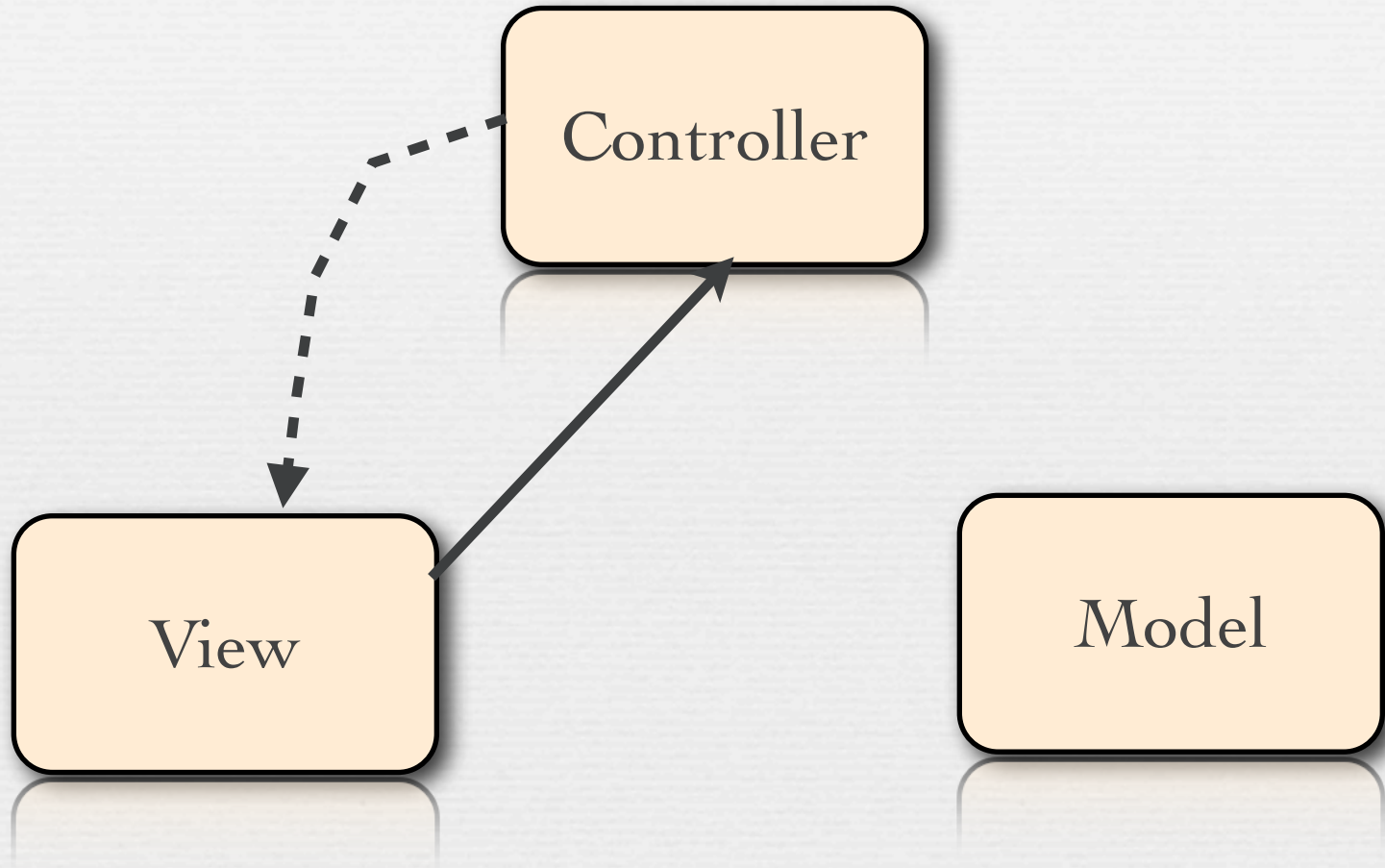
Model

Le modèle: les données
indépendantes

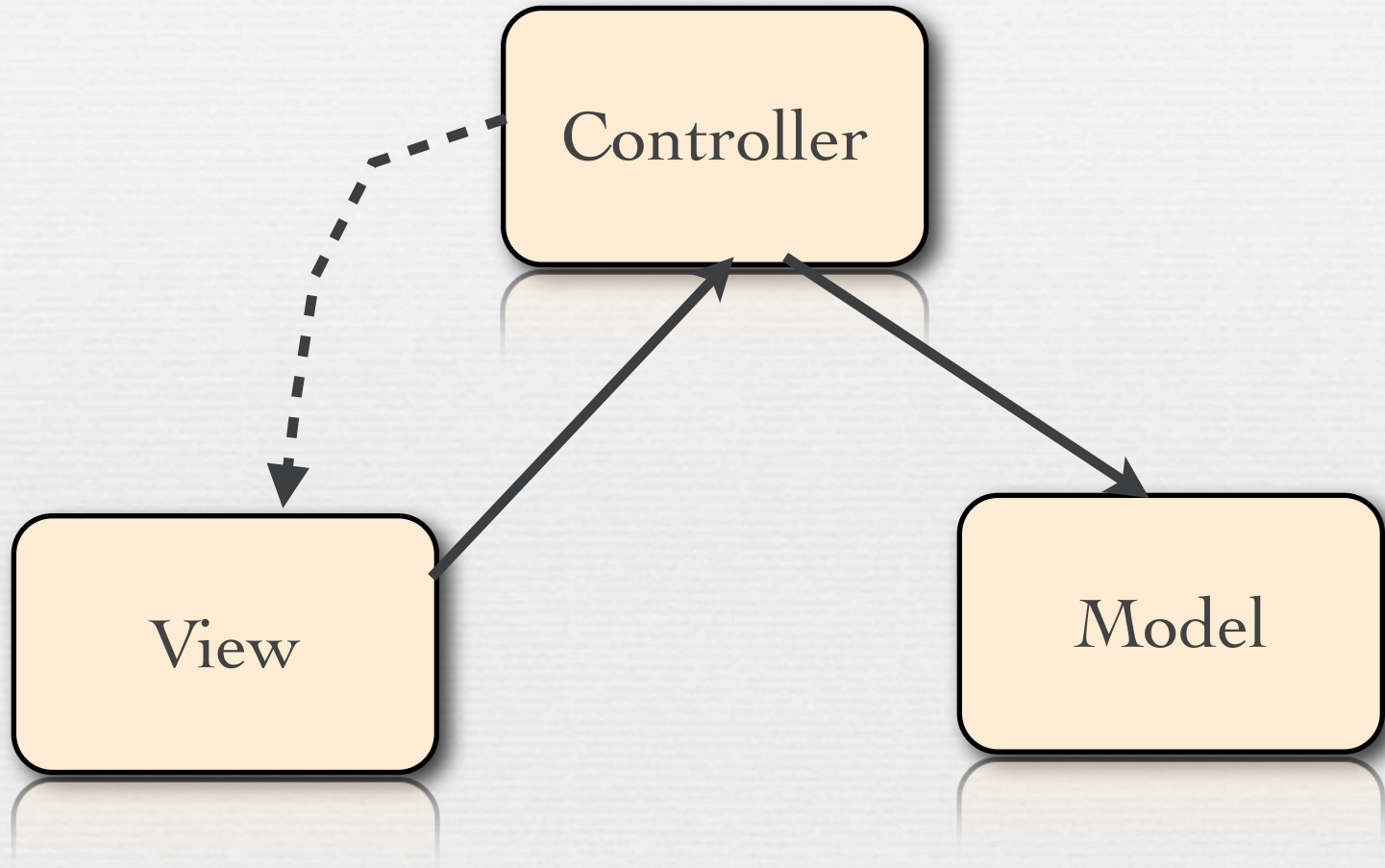
Controlleur observe la vue



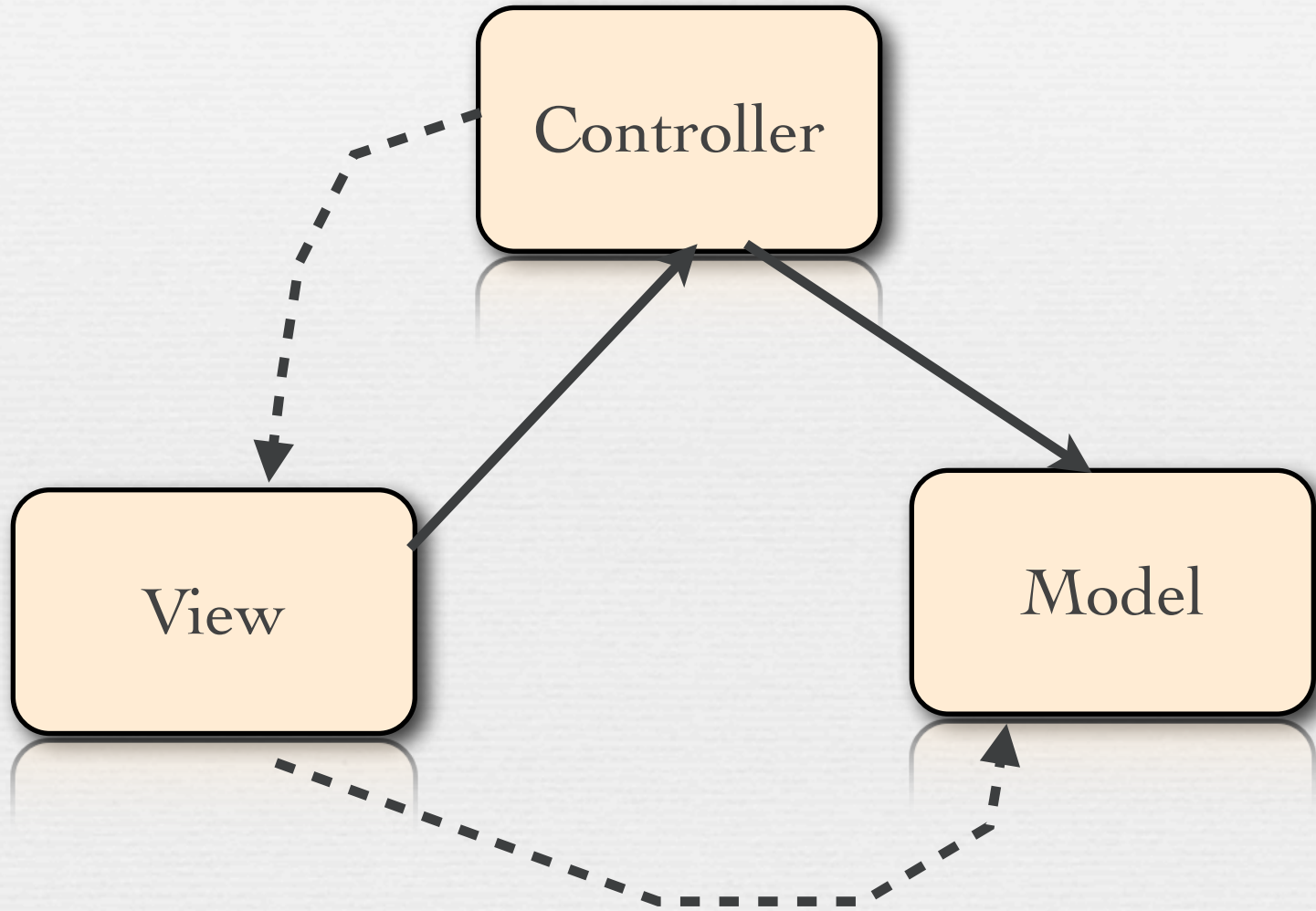
Contrôleur récupère les données de la vue



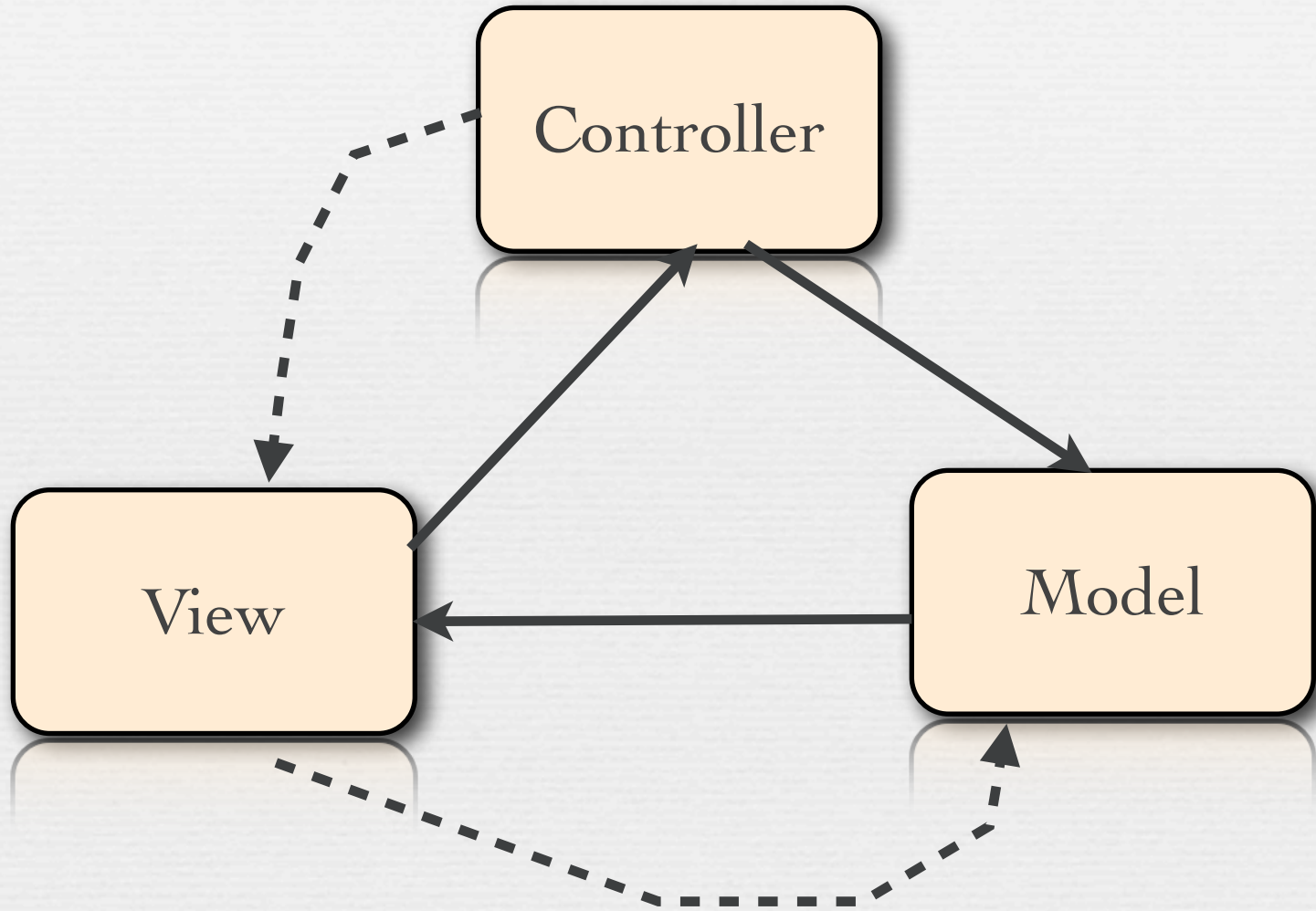
Contrôleur modifie le modèle



Vue observe le modèle



Vue récupère les données du modèle



The background of the slide is a blue-tinted photograph showing several hands in the process of assembling puzzle pieces. The hands are positioned around the edges of the puzzle, with some fingers gripping the pieces. The puzzle pieces are white and have a classic interlocking shape. The overall scene is brightly lit, creating a soft, ethereal atmosphere.

SEPARATIONS :
De la définition du
système à sa mise en
oeuvre

Domaine

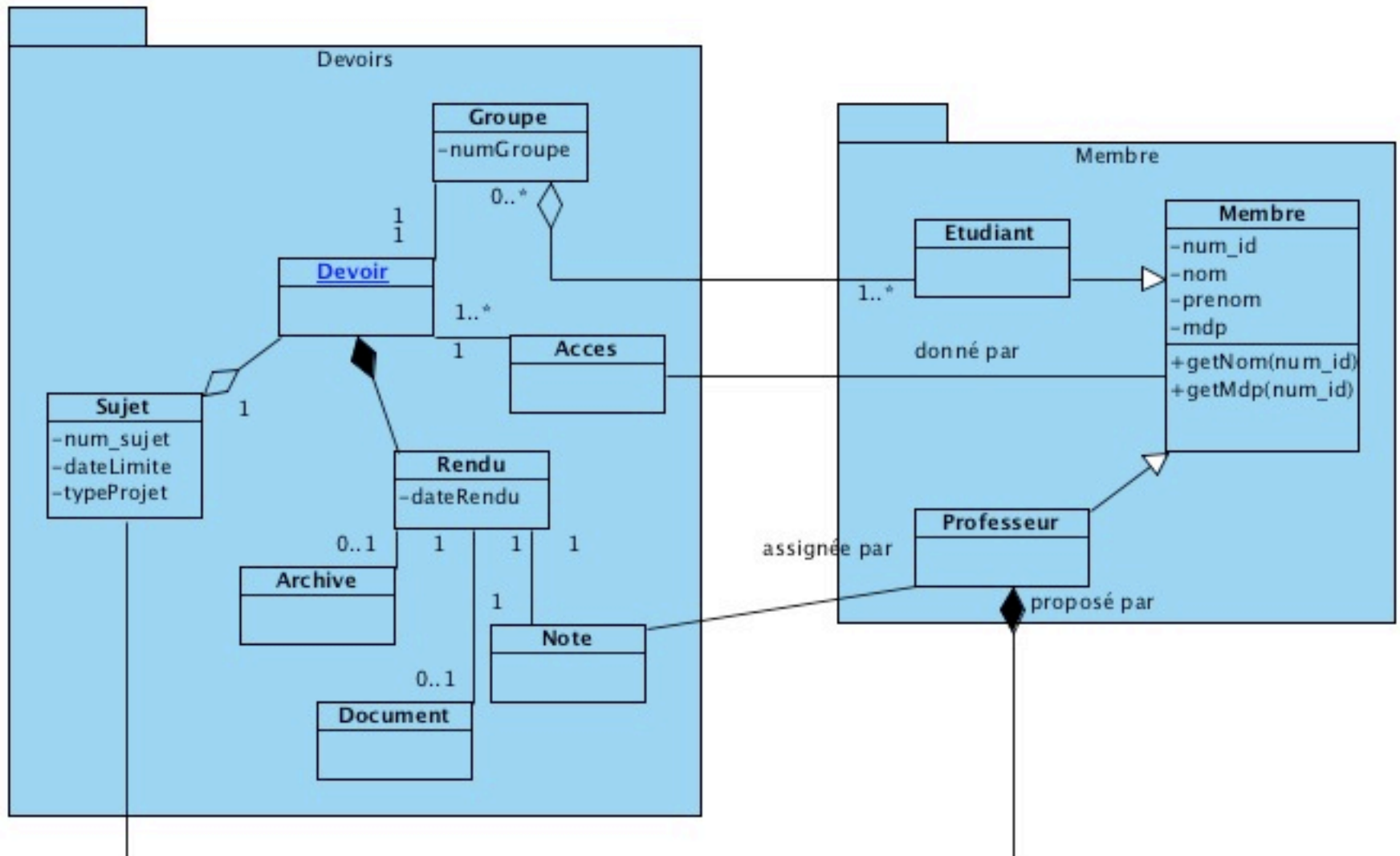
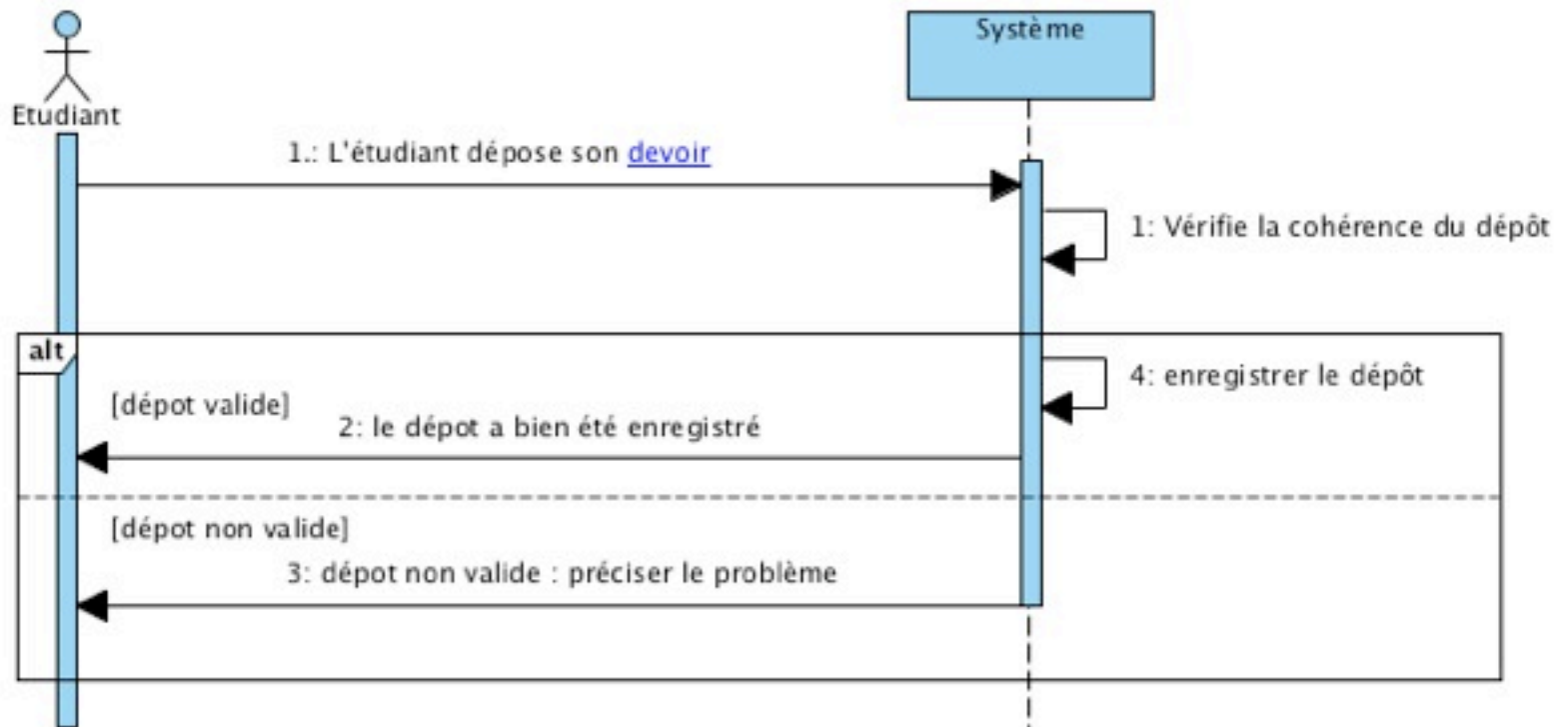


Diagramme de séquence système

sd Déposer un devoir - Flow of Events



Vers la mise en oeuvre

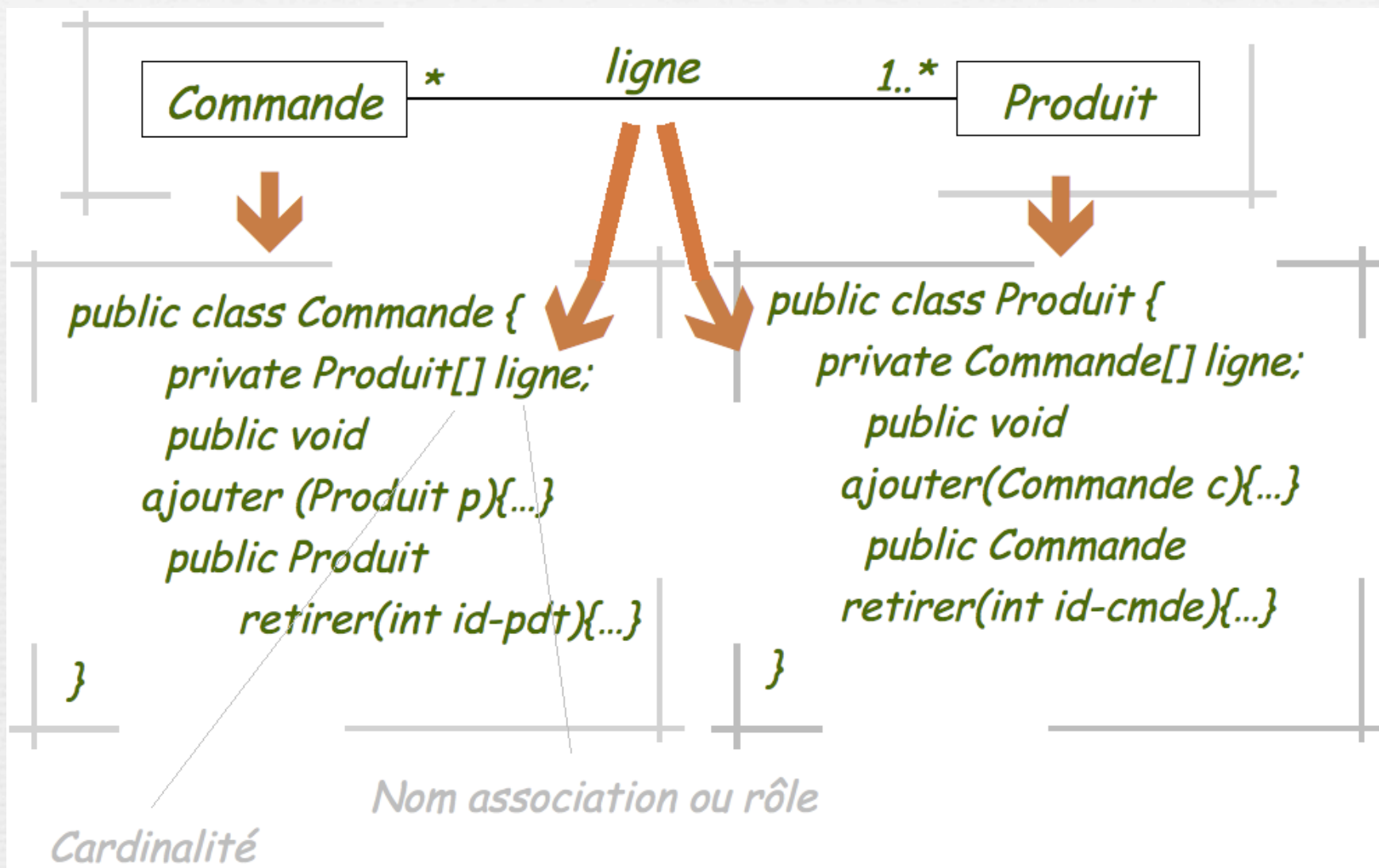
- Reprise du scénario : décomposition
- Retour sur le diagramme de classes

Vers la mise en oeuvre des classes

- Visibilité
- Abstraction
- Attributs et Opérations* de Classes
- Généralisation
- Packages
- Transformations des associations
- Anti-Patterns

Opération : terme générique désignant le plus souvent des méthodes

Association...



Association...



```

public class Commande {
    private Produit[] ligne;
    public void
    ajouter (Produit p){...}
    public Produit
    retirer(int id-pdt){...}
}
    
```

```


public class Produit {
    private Commande[] ligne;
    public void
    ajouter (Commande c){...}
    public Commande
    retirer(int id-cmde){...}
}
    

```

Association:

De la conception à l'implémentation

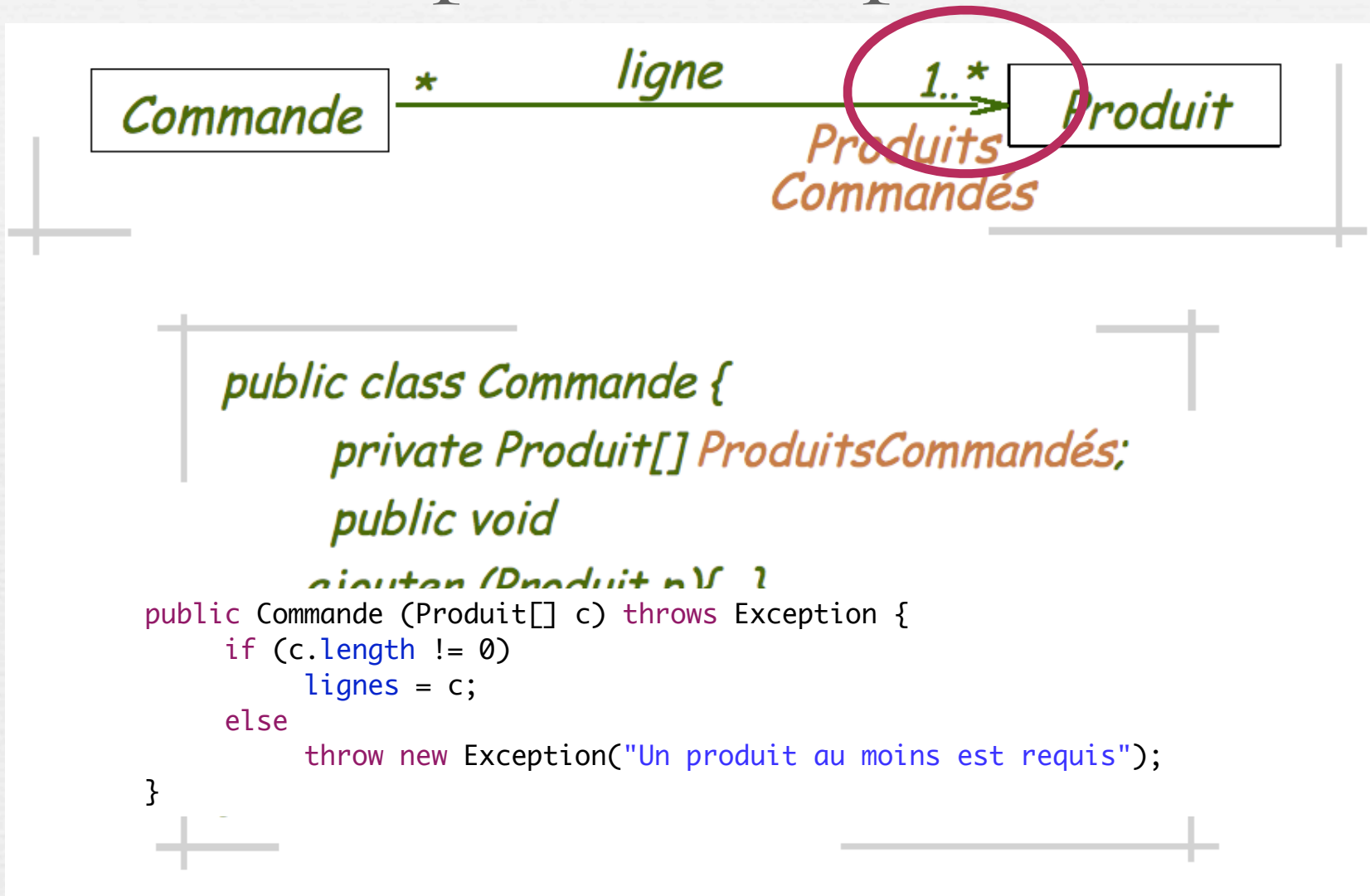


```

public class Commande {
    private Produit[] ProduitsCommandés;
    public void
    ajouter (Produit p){...}
    public Produit
    retirer(int id-pdt){...}
}
    
```

Association:

De la conception à l'implémentation



Association:

De la conception à l'implémentation



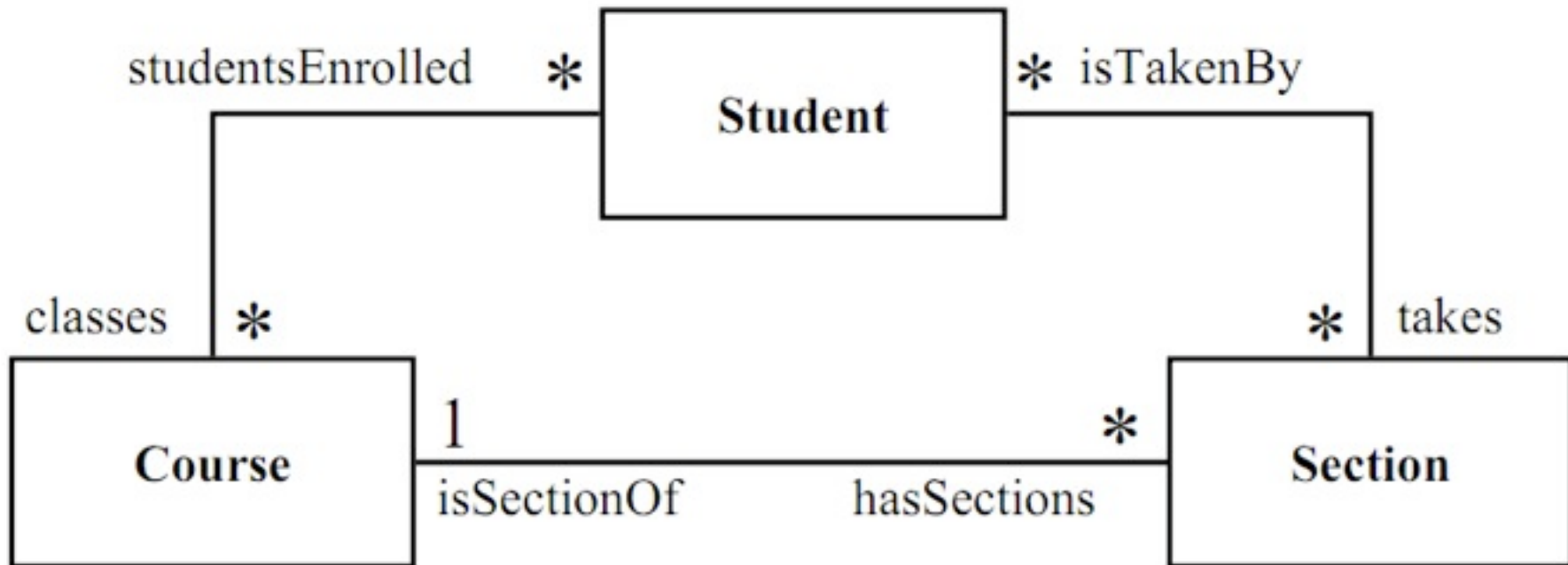
```

public Commande (Produit[] c) throws Exception {
    if (c.length != 0)
        lignes = c;
    else
        throw new Exception("Un produit au moins est
requis");
}
    
```

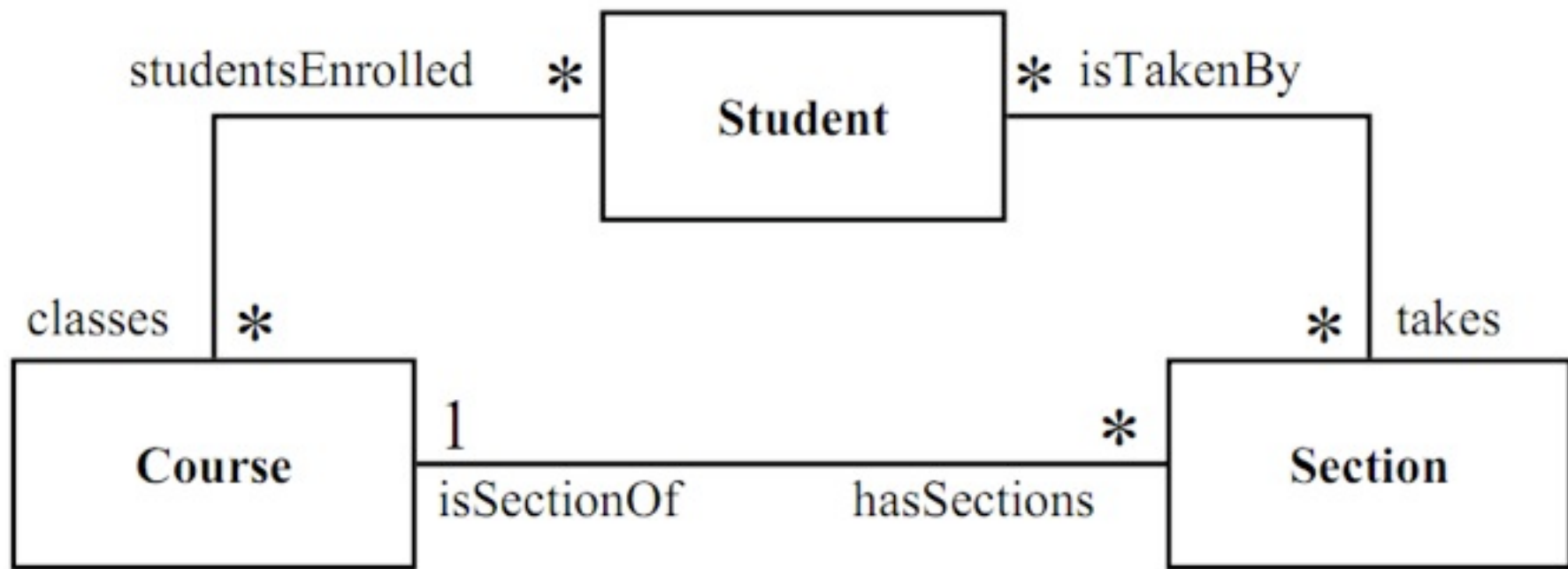
```

public boolean oterProduit(Course c) {
    if (lignes.length==1)
        return false;
    ...
}
    
```

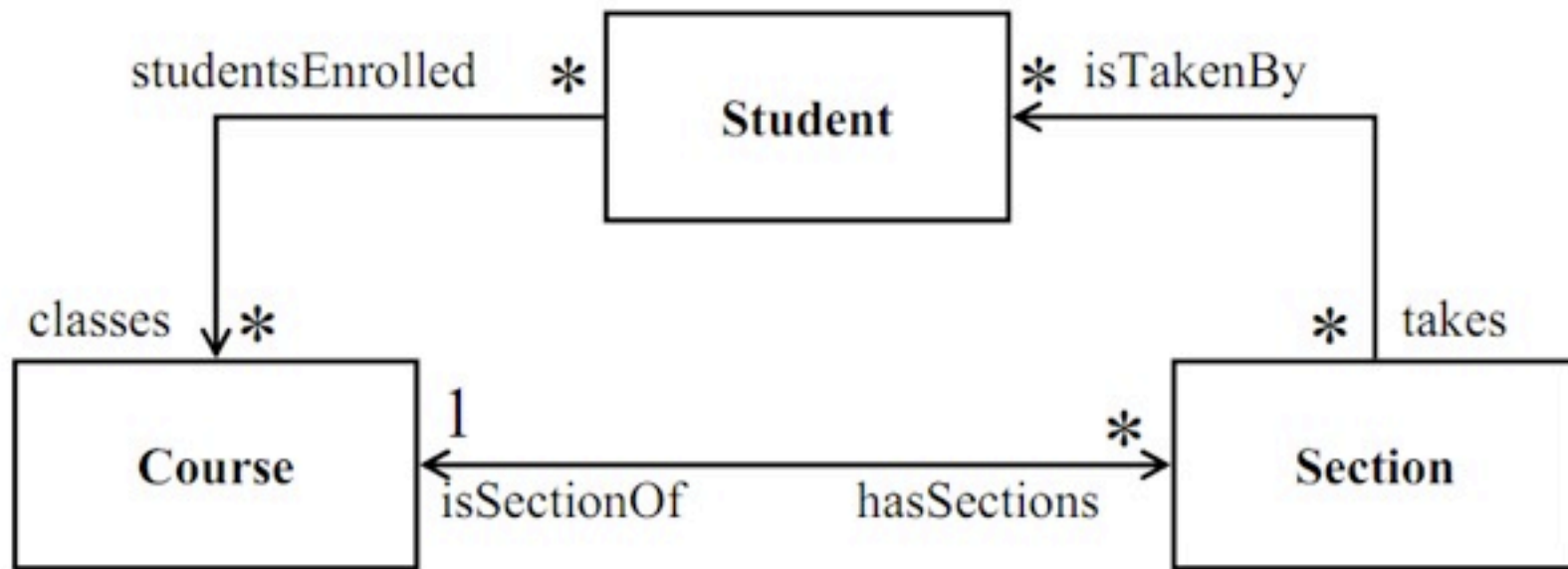
Gestion des associations



Associations & Navigations



Exemple de Raffinement



Ce n'est qu'un exemple, d'autres raffinements sont possibles...

Principes d'implémentation

Principes d'implémentation



Extrémité d'association 1

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *
 - Rôle (pluriel) en collection

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *
 - Rôle (pluriel) en collection
 - Type de l'extrémité en élément de collection

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *
 - Rôle (pluriel) en collection
 - Type de l'extrémité en élément de collection
 - Collection `getRoles()`

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *
 - Rôle (pluriel) en collection
 - Type de l'extrémité en élément de collection
 - Collection `getRoles()`
 - `// Collection<TypeExtrémité>//`

Principes d'implémentation (Suite)

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1
 - `void setRole(Type t)`

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1
 - void setRole(Type t)
- *Fixer* une extrémité d'association *

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association l
 - `void setRole(Type t)`
- *Fixer* une extrémité d'association *
 - `void setRoles(Collection c)`

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1
 - `void setRole(Type t)`
- *Fixer* une extrémité d'association *
 - `void setRoles(Collection c)`
 - `void addRole(TypeElement t)`

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1
 - `void setRole(Type t)`
- *Fixer* une extrémité d'association *
 - `void setRoles(Collection c)`
 - `void addRole(TypeElement t)`
- *Fixer* une association navigable dans les 2 sens :

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association l
 - void setRole(Type t)
- *Fixer* une extrémité d'association *
 - void setRoles(Collection c)
 - void addRole(TypeElement t)
- *Fixer* une association navigable dans les 2 sens :
 - Définir les responsabilités : un des objets est responsable de la connexion/déconnexion (cf. exemple)

Implémentation

```

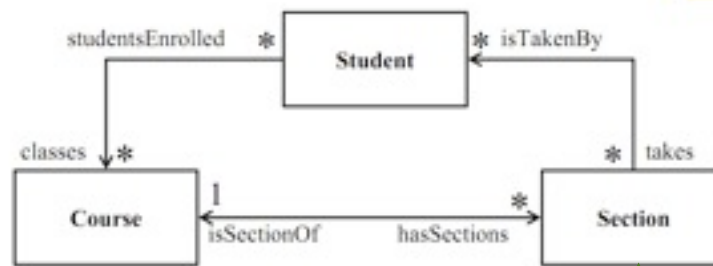
public class Section {
    private String name;
    private Course isSectionOf;
    private Collection<Student> isTakenBy = new ArrayL
    public String toString() {
    public Section(String name, Course isSectionOf) {
        this.name = name;
        //ATTENTION ...
        setIsSectionOf(isSectionOf);
    }

    public void setIsSectionOf(Course c) {
        isSectionOf = c;
        c.addHasSections(this);
    }

    public Course getIsSectionOf() { return isSectionOf;}

    public Collection<Student> getIsTakenBy() {
        return isTakenBy;
    }

    public void addStudent(Student s) {
        isTakenBy.add(s);
        if (!(s.getClasses().contains(isSectionOf)) )
            s.addClass(isSectionOf);
    }
}
    
```



Prise de responsabilités

Implémentation

```

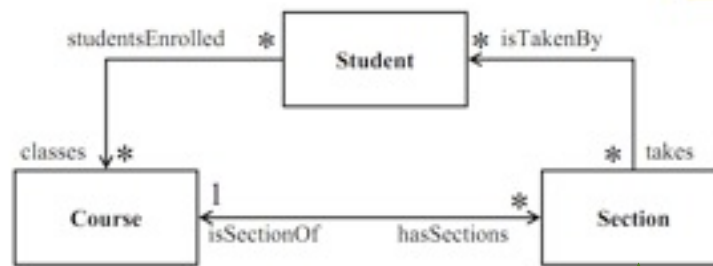
public class Section {
    private String name;
    private Course isSectionOf;
    private Collection<Student> isTakenBy = new ArrayL
    public String toString() {
    public Section(String name, Course isSectionOf) {
        this.name = name;
        //ATTENTION ...
        setIsSectionOf(isSectionOf);
    }

    public void setIsSectionOf(Course c) {
        isSectionOf = c;
        c.addHasSections(this);
    }

    public Course getIsSectionOf() { return isSectionOf;}

    public Collection<Student> getIsTakenBy() {
        return isTakenBy;
    }

    public void addStudent(Student s) {
        isTakenBy.add(s);
        if (!(s.getClasses().contains(isSectionOf)) )
            s.addClass(isSectionOf);
    }
}
    
```



Prise de responsabilités

Implémentation

```

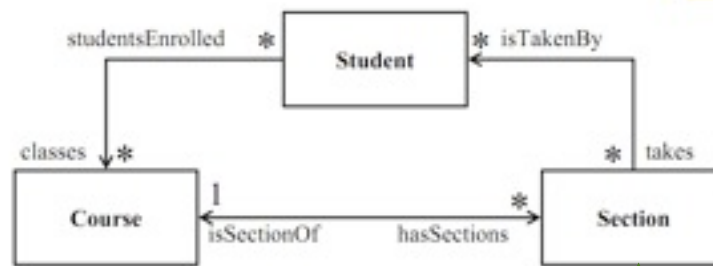
public class Section {
    private String name;
    private Course isSectionOf;
    private Collection<Student> isTakenBy = new ArrayL
    public String toString() {
    public Section(String name, Course isSectionOf) {
        this.name = name;
        //ATTENTION ...
        setIsSectionOf(isSectionOf);
    }

    public void setIsSectionOf(Course c) {
        isSectionOf = c;
        c.addHasSections(this);
    }

    public Course getIsSectionOf() { return isSectionOf;}

    public Collection<Student> getIsTakenBy() {
        return isTakenBy;
    }

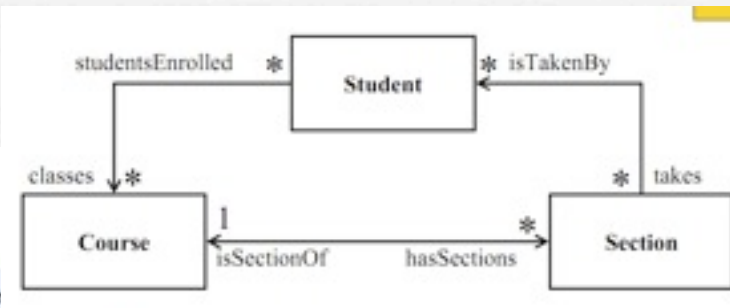
    public void addStudent(Student s) {
        isTakenBy.add(s);
        if (!s.getClasses().contains(isSectionOf))
            s.addClass(isSectionOf);
    }
    
```



Prise de responsabilités



Implémentation



```

public class Student {
    private String name;
    private Collection<Course> classes;

```

```

    public Student(String name) {..}
    public String toString() {..}
    public Collection<Course> getClasses() {
        return classes;
    }
    protected void addClass(Course c){
        classes.add(c);
    }

```

```

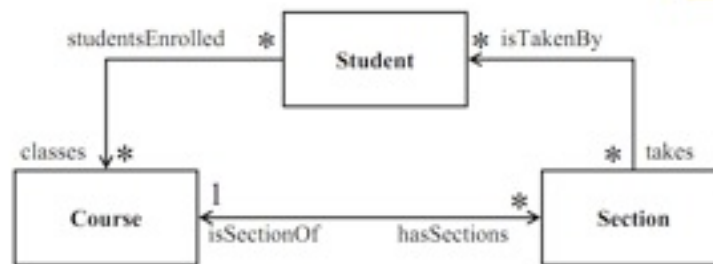
public class Course {
    private String name;
    private Collection<Section> hasSections = new ArrayList<Section>();

    public Course(String name) {..}
    public String toString() {..}
    public Collection<Section> getHasSections() {
        return hasSections;}
    protected void addHasSections(Section s){
        hasSections.add(s);
    }
}

```

Définition des responsabilités
 Ne jamais appeler *addHasSections* ou *addClass* directement !

Implémentation



```

public class Student {
    private String name;
    private Collection<Course> classes;

```

```

    public Student(String name) {..}
    public String toString() {..}
    public Collection<Course> getClasses() {
        return classes;
    }

```

```

    protected void addClass(Course c){
        classes.add(c);
    }

```

```

public class Course {
    private String name;
    private Collection<Section> hasSections = new ArrayList<Section>();

```

```

    public Course(String name) {..}
    public String toString() {..}
    public Collection<Section> getHasSections() {
        return hasSections;
    }

```

```

    protected void addHasSections(Section s){
        hasSections.add(s);
    }
}

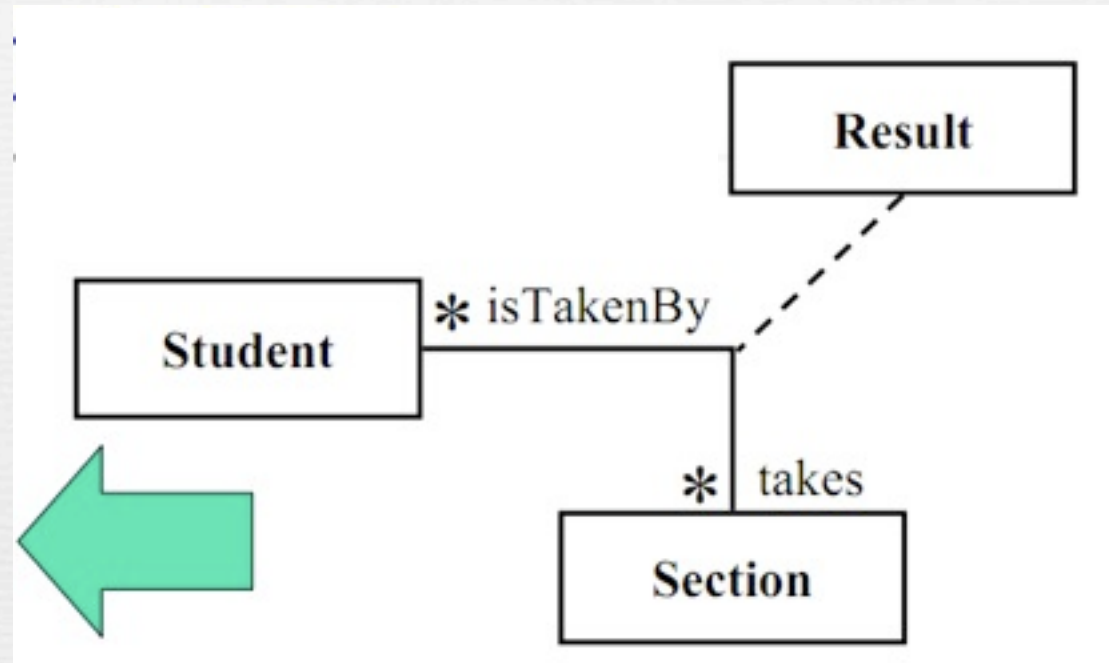
```

Définition des responsabilités
 Ne jamais appeler *addHasSections* ou *addClass* directement !

Implémentation

```

class Student {
    Result getResult(Section s)
}
class Section {
    Result getResult(Student s)
}
class Result {
    Student getStudent()
    Section getSection()
}
    
```



En résumé : Traduction des associations en attributs

En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)

En résumé : Traduction des associations en attributs

- ① Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- ① Association unidirectionnelle = pas d'attribut du côté de la flèche

En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- Association unidirectionnelle = pas d'attribut du côté de la flèche
- Nom de l'attribut = nom du rôle ou forme nominale du nom de l'association

En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- Association unidirectionnelle = pas d'attribut du côté de la flèche
- Nom de l'attribut = nom du rôle ou forme nominale du nom de l'association
- Attribut du type référence sur un objet de la classe à l'autre extrémité de l'association
 - Référence notée « @ »

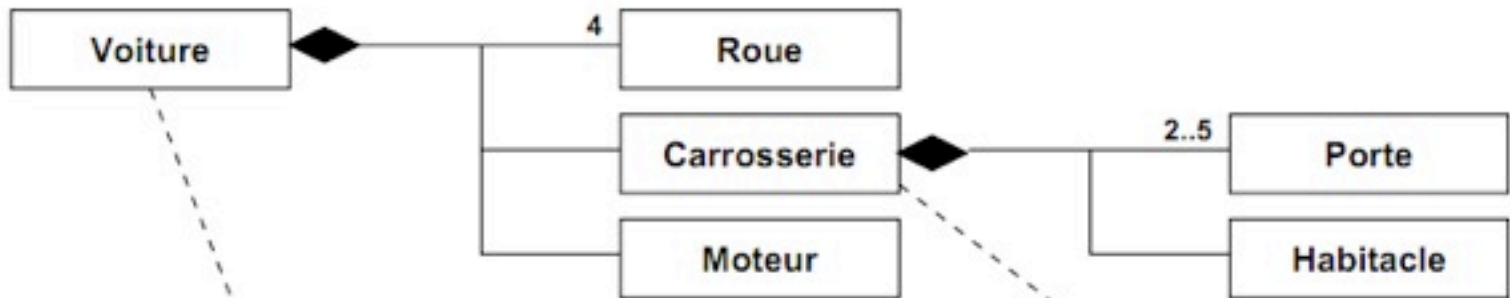
En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- Association unidirectionnelle = pas d'attribut du côté de la flèche
- Nom de l'attribut = nom du rôle ou forme nominale du nom de l'association
- Attribut du type référence sur un objet de la classe à l'autre extrémité de l'association
 - Référence notée « @ »
- Traduction des multiplicités
 - 1 \Rightarrow @Classe
 - * \Rightarrow Collection @Classe
 - 0..N \Rightarrow Tableau[N] Classe

En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- Association unidirectionnelle = pas d'attribut du côté de la flèche
- Nom de l'attribut = nom du rôle ou forme nominale du nom de l'association
- Attribut du type référence sur un objet de la classe à l'autre extrémité de l'association
 - Référence notée « @ »
- Traduction des multiplicités
 - 1 \Rightarrow @Classe
 - * \Rightarrow Collection @Classe
 - 0..N \Rightarrow Tableau[N] Classe
- Multiplicité avec tri = Collection ordonnée @Classe

Compositions



Contrôle du cycle de vie des éléments :

```

constructeur() {
    création objets Roue
    création objet Carrosserie
    création objet Moteur
}

destrcteur() {
    destruction objets Roue
    destruction objet Carrosserie
    destruction objet Moteur
}
    
```

Contrôle du cycle de vie des éléments :

```

constructeur() {
    création objets Porte
    création objet Habitable
}

desrtucteur() {
    destruction objets Porte
    destruction objet Habitable
}
    
```