

# TD Introduction aux Systèmes Interactifs

<https://www.lri.fr/~prouzeau/ISI/Sujets/index.html>

Arnaud Prouzeau  
[prouzeau@lri.fr](mailto:prouzeau@lri.fr)

# **Design Pattern**

## **Modèle-Vue-Contrôleur (MVC)**

Exercice 3-4

# MVC

---

Basé sur le principe « Diviser pour régner » :

## **Modèle**

Représente le coeur de l'application, les données et les méthodes associées.

## **Vue**

Définit une représentation visuelle du modèle (par exemple un Slider pour représenter un nombre).

## **Controller**

Fait le lien entre les vues et le modèle, gère les actions utilisateurs. Dès qu'il capte un événement, il modifie le modèle en conséquence et demande le rafraichissement des vues.

# Modèle

---

## Different types

### Valeurs

String, numbers

### Séquences

Lists (AbstractListModel)

# Views

---

Fournit une représentation du modèle

La représentation est souvent discrète.

Exemples de Java Swing :

Texte

JTextField, JLabel

Nombre

JSlider, JTextField, JLabel

Séquence

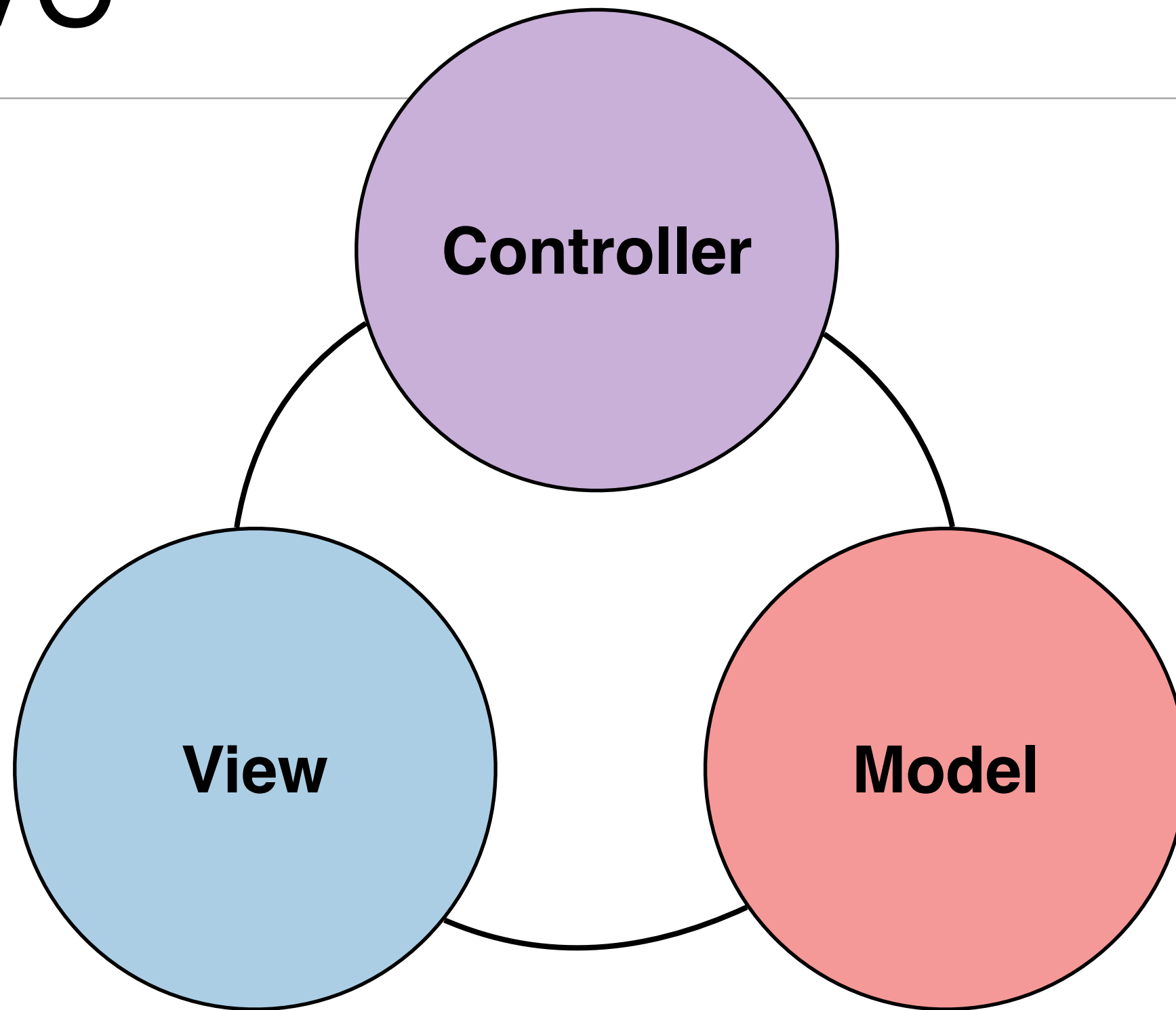
JLists, JComboBox

Méthode

JButton, JMenuItem

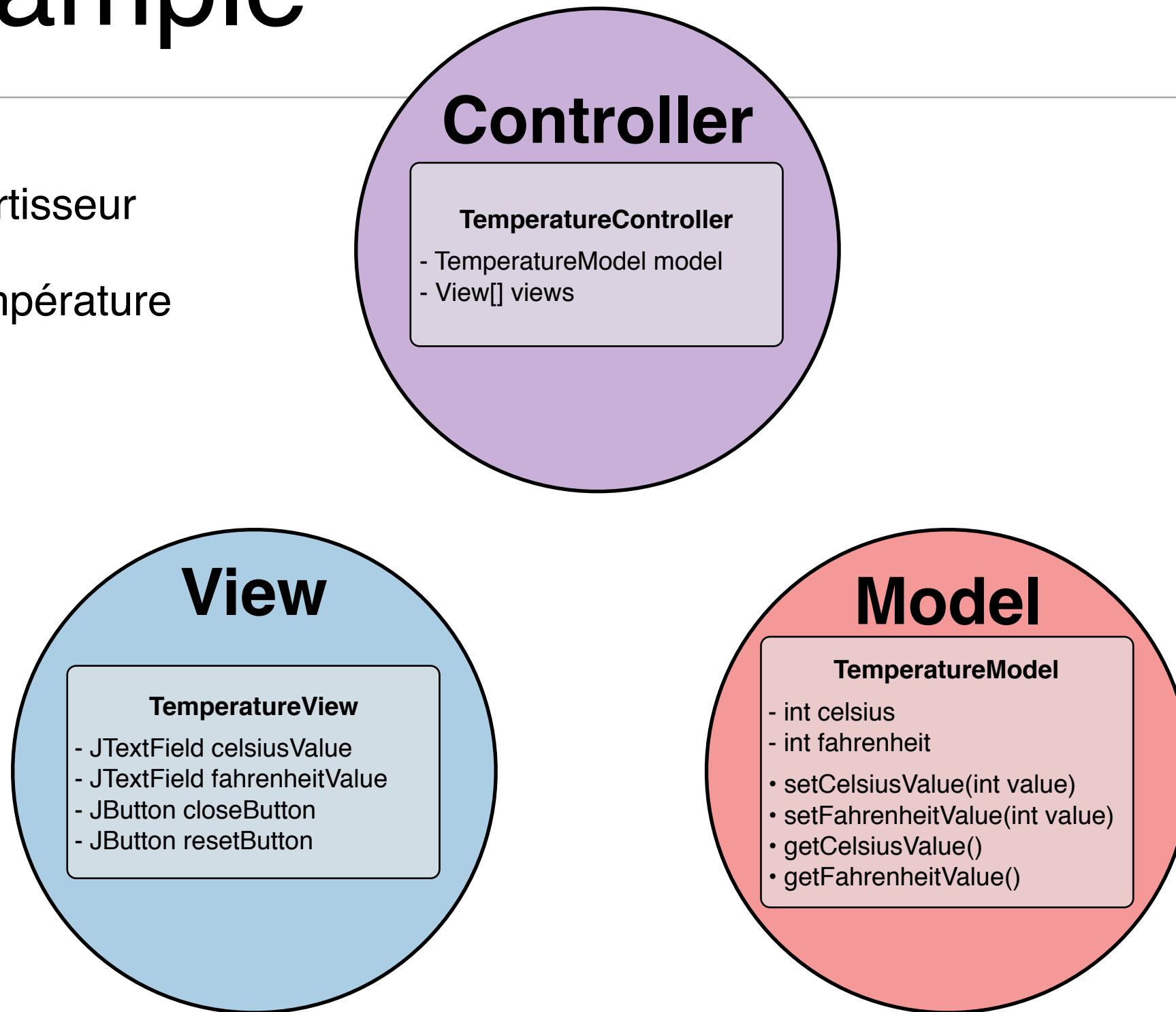
# MVC

---



# Example

Convertisseur  
de Température



# Example

Convertisseur  
de Température

## Controller

### TemperatureController

- TemperatureModel model
- View[] views
- setCelsiusValue(int value, Object caller)
- setFahrenheitValue(int value, Object caller)
- addView(View view)
- NotifyAllViews()

notify

request update

modify

## View

Temperature converter

Celsius  
37

Fahrenheit  
98.6

Reset Close

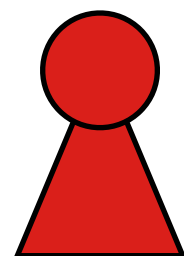
input

update

## Model

### TemperatureModel

- int celsius
- int fahrenheit
- setCelsiusValue(int value)
- setFahrenheitValue(int value)
- getCelsiusValue()
- getFahrenheitValue()



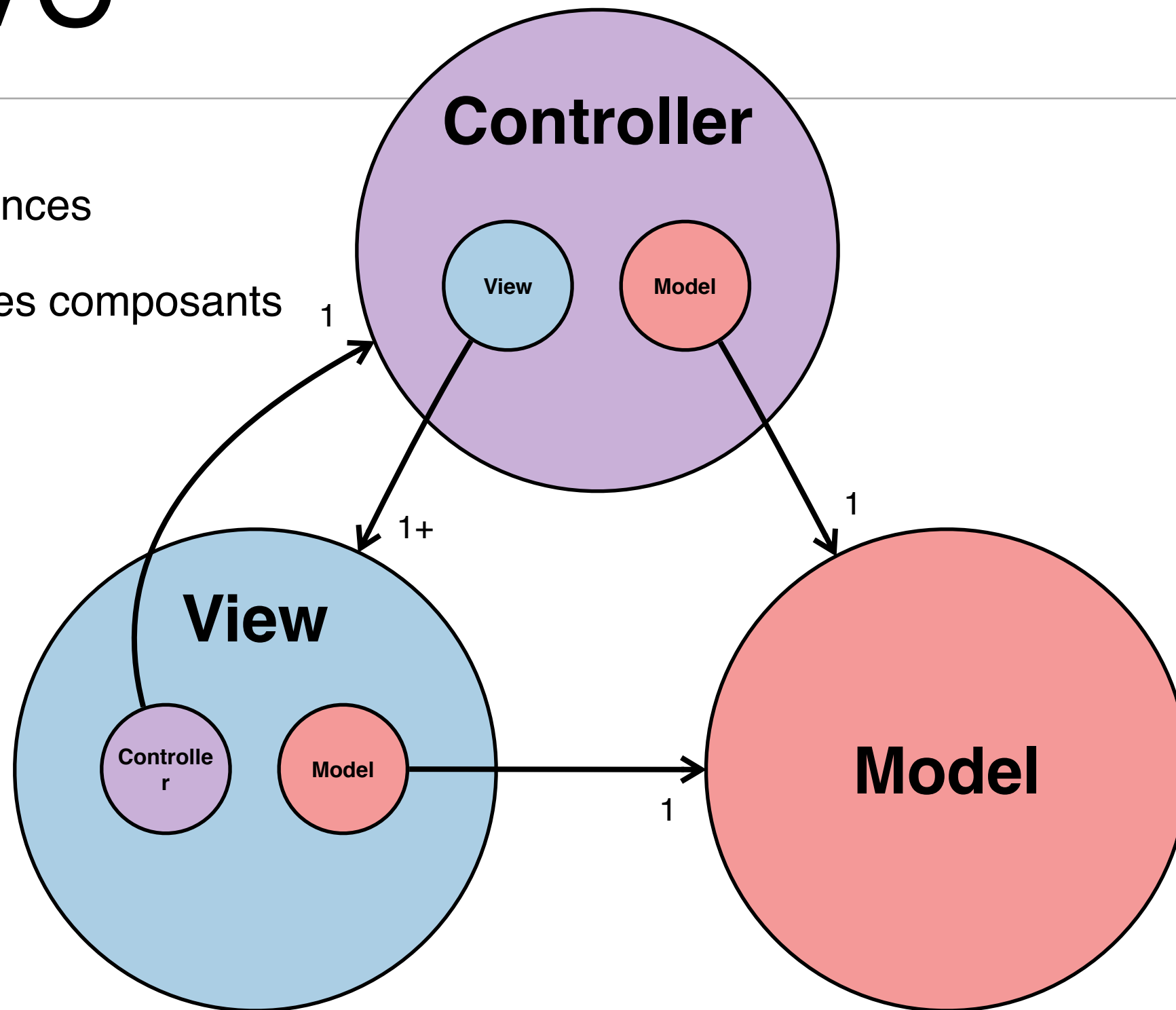
User



# MVC

Références

entre les composants



# Correction Diagramme UML

