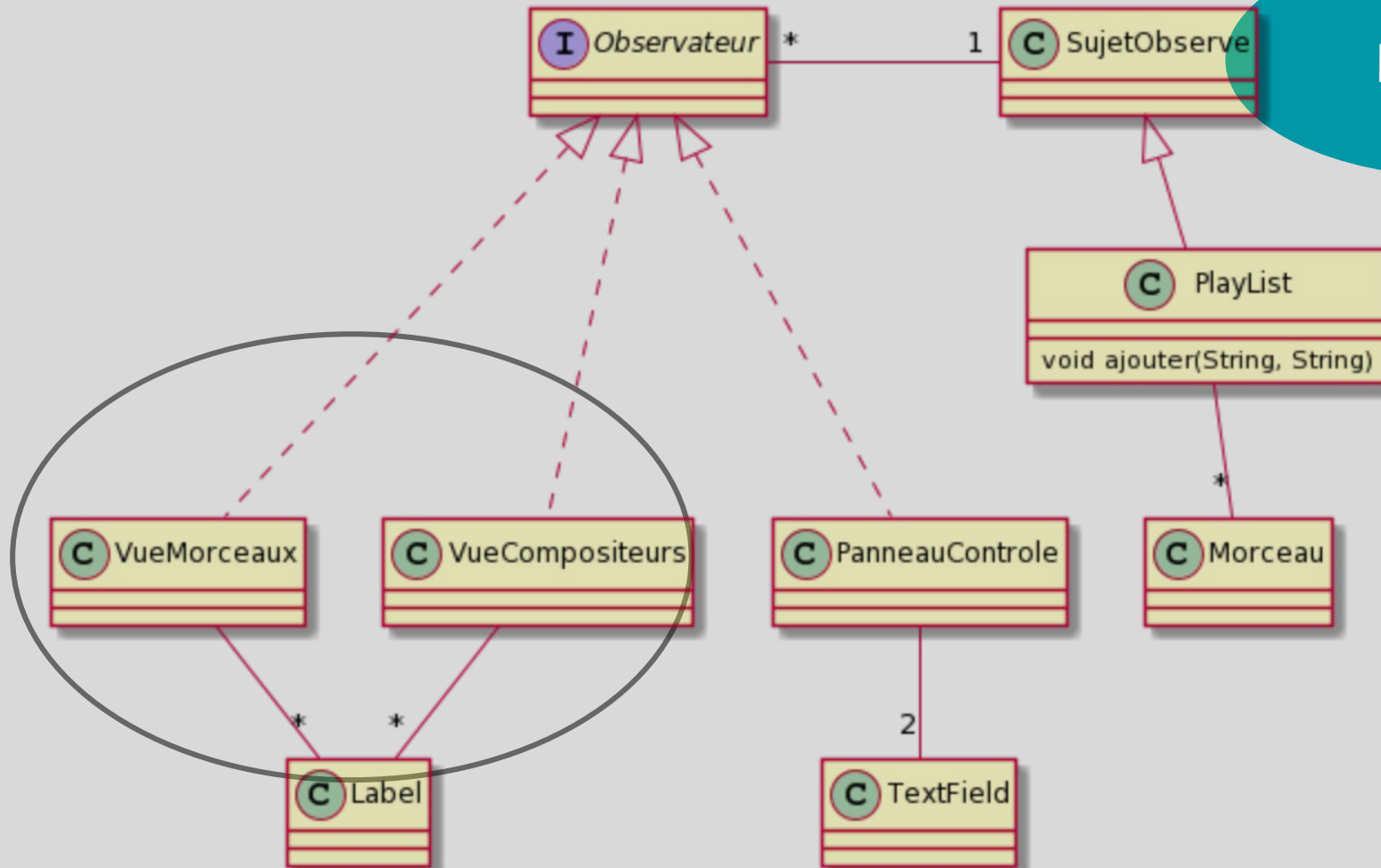


# JavaFX & FXML

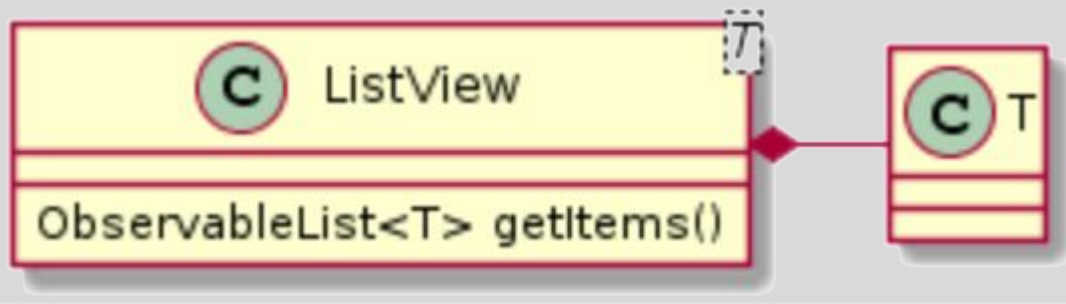
[martine.gautier@univ-lorraine.fr](mailto:martine.gautier@univ-lorraine.fr)

# Extrait

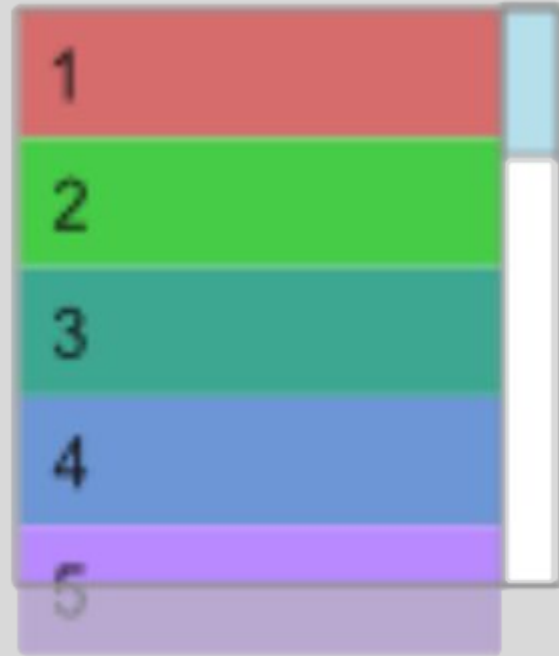
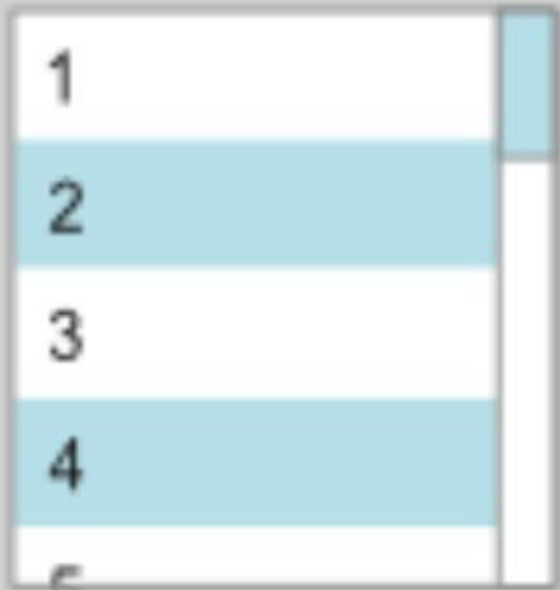


- ▲ Utilisation d'un `GridPane` pour y ranger chaque élément de la liste, item après item
- ▲ Utilisation conjointe des modes déclaratif et procédural
- ▲ Avantage : facile à mettre en place
- ▲ Inconvénient : aucune sélection d'item n'est possible

- ▲ Composant évolué générique `ListView<T>`
  - Look & feel adaptable
- ▲ Attaché à une collection d'objets de type `T`



- ▲ Lorsque la collection est modifiée, le composant `ListView` est rafraîchi automatiquement.



Même si la liste contient 7 éléments, seules 5 cellules de listes sont nécessaires ; en cas de scroll, elles sont réutilisées.

```

package playList.view ;
class VueCompositeurs implements Observateur {
    private PlayList pl;
    @FXML
    private ListView<String> compositeurs ;
    public VueCompositeurs(PlayList pl) {
        this.pl = pl;
        pl.ajouterObservateur(this) ;
    }
    public void reagir() {
        compositeurs.getItems().clear();
        this.pl.getCompositeurs().
            forEach(c->compositeurs.getItems().add(c));
    }
}

```

- ▲ Modification de l'apparence d'une `ListView<T>`  
= utiliser une fabrique de cellules

```
// Autoriser la sélection multiple
compositeurs.getSelectionModel()
    .setSelectionMode(SelectionMode.MULTIPLE);

// Rendre de cellules
compositeurs.setCellFactory(lv -> new CompoCell());
```



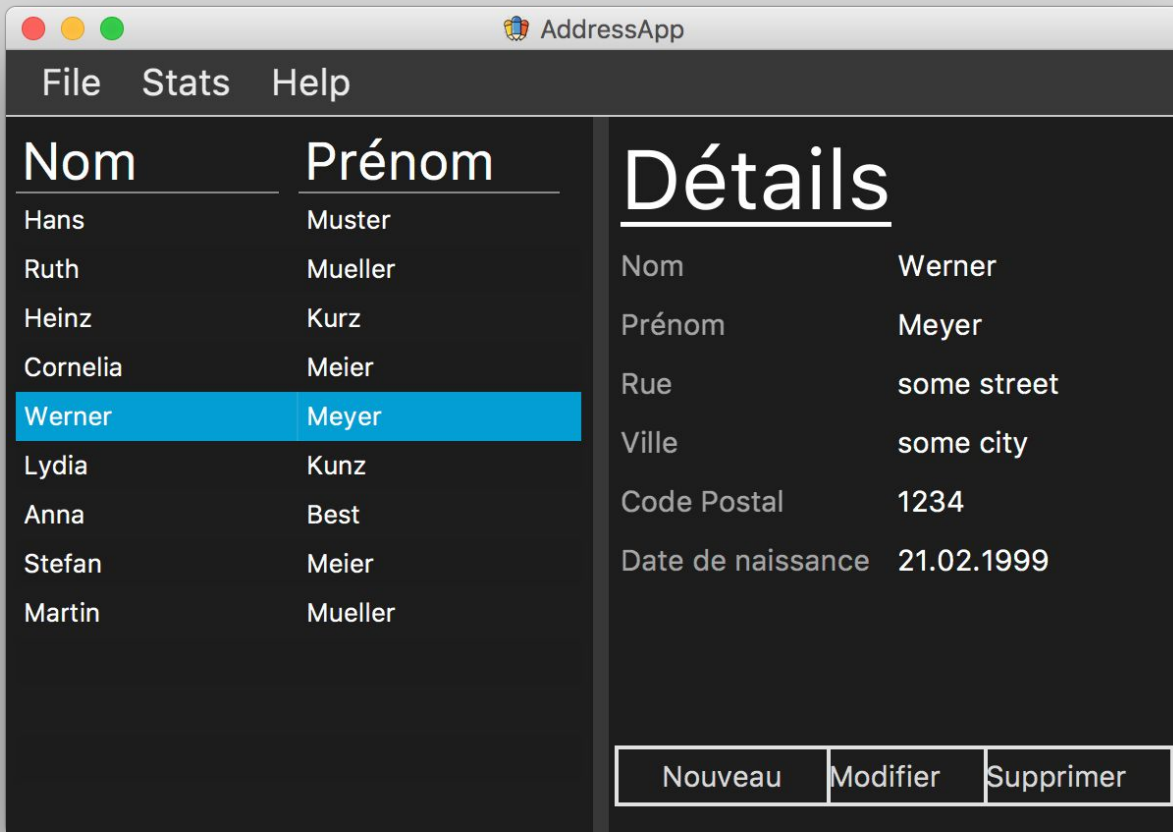
**CompoCell.java + compoCell.xml**

```
package playList.view ;  
class CompoCell extends ListCell<String> {  
    @FXML  
    private Label compo ;  
    @FXML  
    private Label count ;  
    private PlayList playList ;  
    public CompoCell(PlayList pl) {  
        // load du fichier xml;  
    }  
    public void updateItem(String item, boolean empty) {  
        // mise à jour des composants  
    }  
}
```



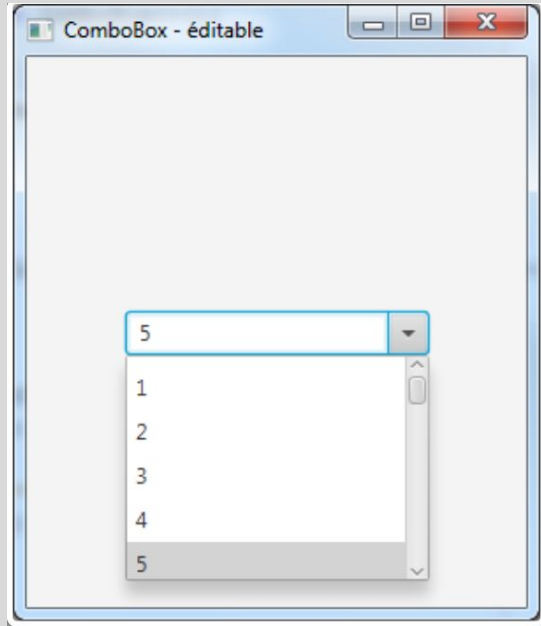
## ▲ Ajouter un écouteur

```
// La valeur de chaque cellule sélectionnée est affichée  
compositeurs.getSelectionModel()  
    .selectedItemProperty()  
    .addListener(  
        (observable, oldValue, newValue) ->  
            System.out.println(newValue)) ;
```



# TableView

# Autres composants



A screenshot of a Java Swing window titled "TableView". It displays a table with four columns: "Couleur", "Rouge", "Vert", and "Bleu". The table contains 10 rows of data, each representing a color and its corresponding RGB values. The first row has empty cells for the color name and values. The subsequent rows show various colors with their respective RGB values.

Couleur	Rouge	Vert	Bleu
	1.0	1.0	1.0
	0.0	0.0	0.0
	1.0	0.843137264251709	0.0
	1.0	0.0	0.0
	0.0	0.501960813999176	0.0
	0.0	0.0	1.0
	1.0	0.0	1.0
	1.0	1.0	0.0
	0.0	1.0	1.0

Un tuto ?

TreeTableView								
Nom	Type	Taille	Création	Modifié	Dernier accès	Lisible ?	Modifiable ?	Exécutable ?
▼ 0165 dv...	File folder	0	2014-04-02T02:...	2014-05-06T03:...	2014-05-06T03:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
▶ build	File folder	0	2014-05-06T02:...	2014-05-07T04:...	2014-05-07T04:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
▶ dist	File folder	0	2014-05-06T03:...	2014-05-07T04:...	2014-05-07T04:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
▶ nbproj...	File folder	0	2014-04-02T02:...	2014-04-02T02:...	2014-04-02T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
▶ src	File folder	0	2014-04-02T02:...	2014-05-06T02:...	2014-05-06T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
build.x...	XML Document	3150	2014-04-02T02:...	2014-04-02T02:...	2014-04-02T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
conne...	SVG File	8940	2014-04-15T22:...	2014-04-16T03:...	2014-04-15T22:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
manife...	MF File	85	2014-04-02T02:...	2014-04-02T02:...	2014-04-02T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
scenic...	PROPERTIES File	525	2014-04-11T00:...	2014-04-16T00:...	2014-04-11T00:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
virt.png	PNG File	37526	2014-04-02T23:...	2014-04-03T02:...	2014-04-02T23:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
virt.svg	SVG File	96911	2014-04-02T22:...	2014-04-03T02:...	2014-04-02T22:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
virt2.p...	PNG File	2395	2014-04-03T02:...	2014-04-03T02:...	2014-04-03T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
virt3.p...	PNG File	5726	2014-04-03T02:...	2014-04-03T02:...	2014-04-03T02:...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

```
public void start(Stage primaryStage) throws Exception{  
    WebView ww = new WebView();  
    WebEngine we = ww.getEngine() ;  
    we.load("http://.....") ;  
    primaryStage.setTitle("Web");  
    primaryStage.setScene(new Scene(new VBox(ww), 300, 275));  
    primaryStage.show();  
}
```

Composant utilisé pour charger les données affichées dans le navigateur.

```
public void start(Stage primaryStage) throws Exception{  
    WebView ww = new WebView();  
    WebEngine we = ww.getEngine() ;  
    we.load("http://.....") ;  
    primaryStage.setTitle("Web");  
    primaryStage.setScene(new Scene(new VBox(ww), 300, 275));  
    primaryStage.show();  
}
```

- ▲ Processus qui fait varier une propriété quelconque d'une valeur initiale à une valeur finale, au cours d'un certain laps de temps
  - Déplacer un composant d'un point à un autre
  - Modifier la couleur d'un composant entre le bleu et le vert
- ▲ Régie par une **Timeline**, composée d'une succession de **KeyFrame**
  - Implicite
  - Explicite, avec possibilité d'ajouter des événements

## Tutoriel

<https://www.genuinecoder.com/javafx-animation-tutorial/>



## Application systématique du DP Observer

- Le modèle est un `SujetObserver`.
- Chaque vue est un `Observateur` ; elle s'inscrit auprès du modèle (si besoin).  
`modele.ajouterObservateur(this)`
- A chaque transformation du modèle, les vues sont prévenues ; pour se rafraîchir, elles consultent le modèle.

## ▲ Avantages

- Facile à mettre en oeuvre
- Séparation claire entre Modèle et Vues/Contrôleurs

## ▲ Inconvénients

- Hériter de `SujetObserver`
- Toutes les vues sont prévenues à chaque mise à jour du modèle ; charge à elles de se rafraîchir ou non

## ▲ Alternative ?

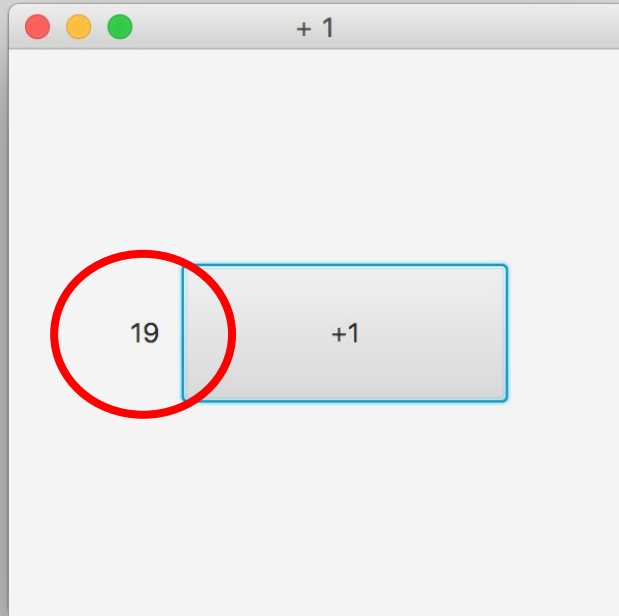
## ▲ Propriétés

- Emballage de type de base (wrapper)  
IntegerProperty DoubleProperty StringProperty ...
- Interfaces avec implémentation simple  
SimpleIntegerProperty SimpleDoubleProperty ...
- *getter/setter*  
intValue, setValue

## ▲ Binding

- Lier deux propriétés, de sorte que la mise à jour de l'une soit automatiquement propagée à l'autre
- Binding simple ou bidirectionnel

## Exemple



Lier l'attribut du modèle et le  
label de la vue

```
<GridPane alignment="center" hgap="10" vgap="10" ...  
    fx:controller="compteur.VueCompteur">
```

```
<children>
```

```
    <Button mnemonicParsing="false" onAction="#plusUn" />
```

```
    <Label fx:id="valeurCpt" text="Valeur du compteur" />
```

```
</children>
```

```
</GridPane>
```

## Exemple

```
public class Compteur {  
    private IntegerProperty valeur ;  
    public Compteur() {  
        this.valeur = new SimpleIntegerProperty(18) ;  
    }  
    public void incrementer() {  
        valeur.setValue(valeur.intValue()+1);  
    }  
    public IntegerProperty getPropertyValue() {  
        return valeur ;  
    }  
    public int getValue() { return valeur.intValue();}  
}
```

## Exemple

```
public class VueCompteur {  
    @FXML  
    private Label valeurCpt ;  
    private Compteur compteur ;  
    public VueCompteur(Compteur cpt) { this.compteur = cpt ;}  
    @FXML  
    public void initialize(){  
        valeurCpt.textProperty()  
            .bind(compteur.getPropertyValue().asString());  
    }  
    public void plusUn() { this.compteur.incrementer(); }  
}
```

## ▲ Avantages du binding

- Gestion automatique de la mise à l'écoute et de la notification
- Notification ciblée

## ▲ Inconvénients

- Utilisation des propriétés dans le modèle, à la place des types de base  
`IntegerProperty`, `ObservableList<T>`