

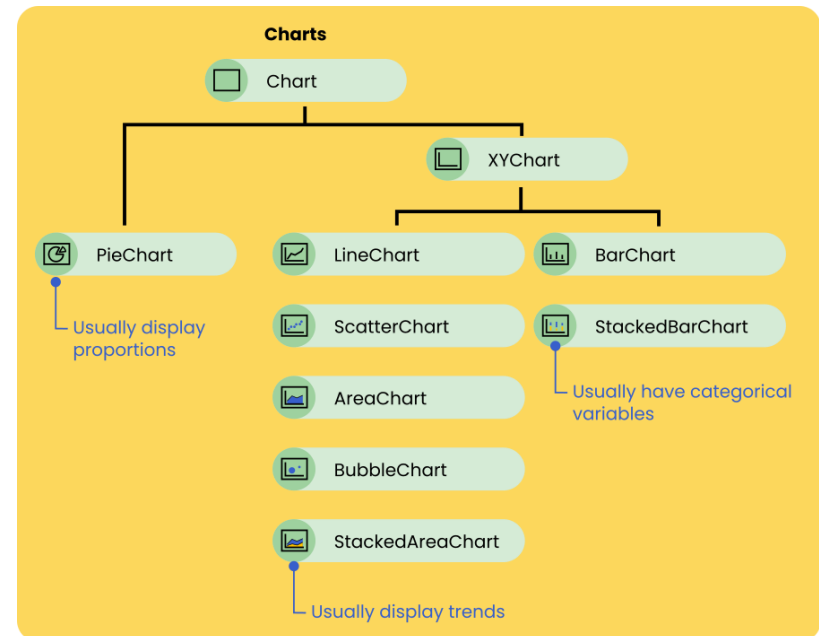
# JavaFX Charts

# Introduction:

*JavaFX fournit **8 graphiques** par défaut pour afficher les données.*

➤ *7 d'entre eux ont des axes et sont conçus pour afficher des données en deux dimensions le long de ces axes.*

- **AreaChart,**
- **BarChart,**
- **BubbleChart,**
- **LineChart,**
- **ScatterChart,**
- **StackedAreaChart,**
- **StackedBarChart**



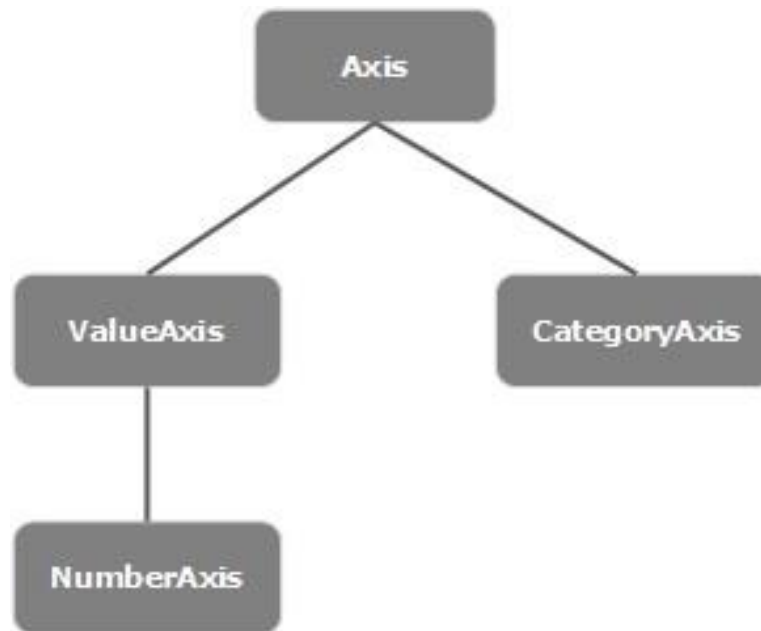
➤ ***PieChart** est le seul qui n'utilise pas d'axes. PieChart implémente directement la classe Chart et est responsable de toutes ses propres exigences de traçage.*

# Création d'un graphique:

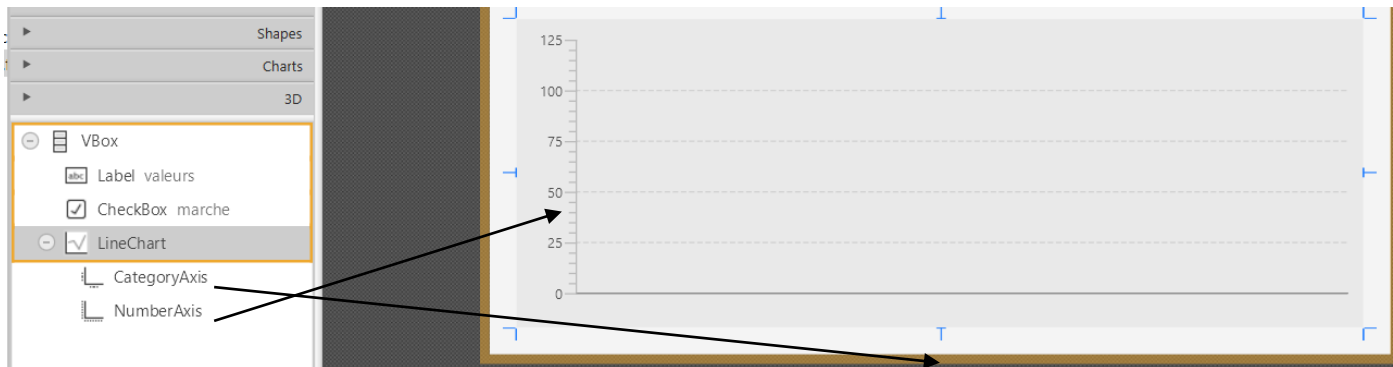
*Pour créer un graphique, vous devez:*

- *Définir l'axe du graphique*
- *Instancier la classe respective*
- *Préparer et transmettre les données au graphique*

# Création d'un graphique: définir l'axe du graphique



# Création d'un graphique: définir l'axe du graphique



*xml*

*java*

```
<LineChart fx:id="lineChart">
  <xAxis>
    <CategoryAxis side="BOTTOM" fx:id="xAxis" />
  </xAxis>
  <yAxis>
    <NumberAxis fx:id="yAxis" side="LEFT" />
  </yAxis>
</LineChart>
```

```
@FXML
CategoryAxis xAxis;
@FXML
NumberAxis yAxis;
@FXML
LineChart<String, Number> lineChart;
```

# Création d'un graphique: Préparer et transmettre...

## ➤ *Créer l'objet serie et le lier à lineChart:*

```
XYChart.Series<String,Number> serie = new XYChart.Series<>();  
...  
...  
lineChart.getData().add(serie);
```

## ➤ *Fabriquer des données pour la série:*

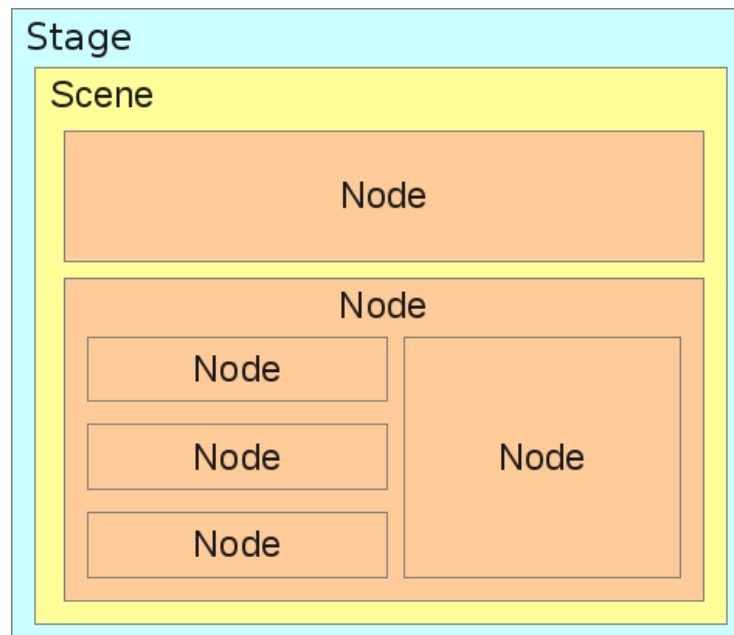
```
serie.getData().add(new XYChart.Data<>(X,Y));
```

# Afficher les coordonnées

## ➤ *Rappel: Scene Graph en JavaFX (DOM de Node)*

*Les applications JavaFX se composent d'un **Stage** et d'une ou plusieurs **Scene(s)***

- *Le **Stage** est le conteneur de niveau supérieur de votre application.*
- *La(les) Scène(s) en revanche, contiennent tout le contenu (éléments de l'Interface Utilisateur) de votre application .*
- *Un **nœud** est un élément du graphique de scène.*



# Afficher les coordonnées

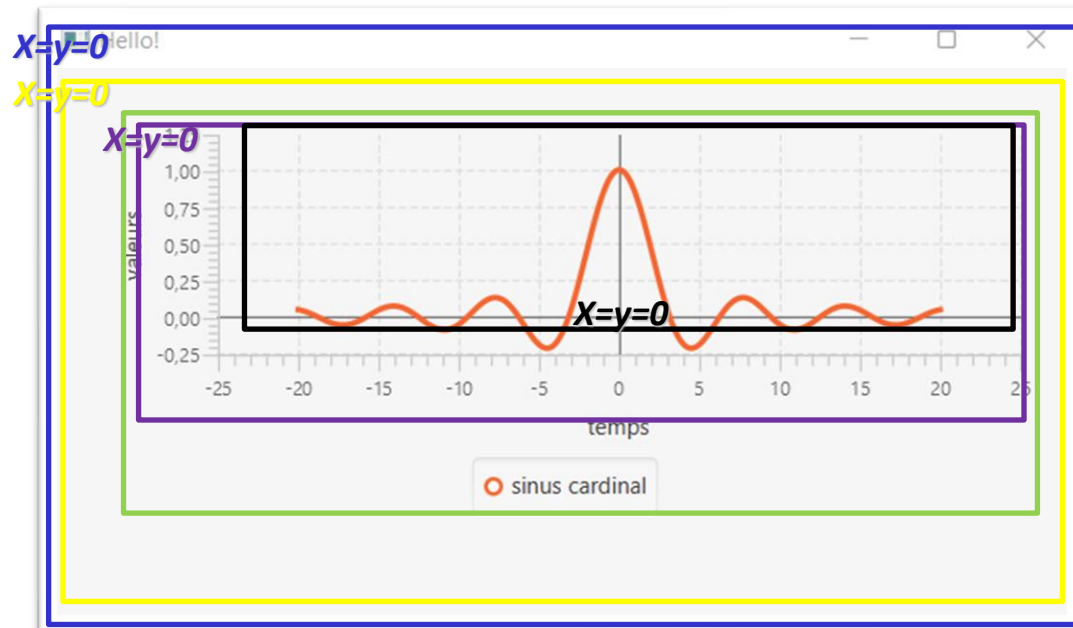
## ➤ Rappel: Scene Graph en JavaFX (DOM de Node)

### Stage

- Scene
  - lineChart
    - chart-legend
      - « sinus cardinal »
    - chart-content
      - yAxis
      - xAxis
      - chart-plot-background
      - XYChart
    - chart--title
  - label

**ATTENTION**

Pour avoir des valeurs cohérentes de coordonnées, il faut donc être dans le contexte du **node chart-plot-background**








# Afficher les coordonnées


➤ *Récupérer le Node pour y lier les « EventHandler » de la souris:*


```
final Node chartBackground = lineChart.lookup(".chart-plot-background");
```

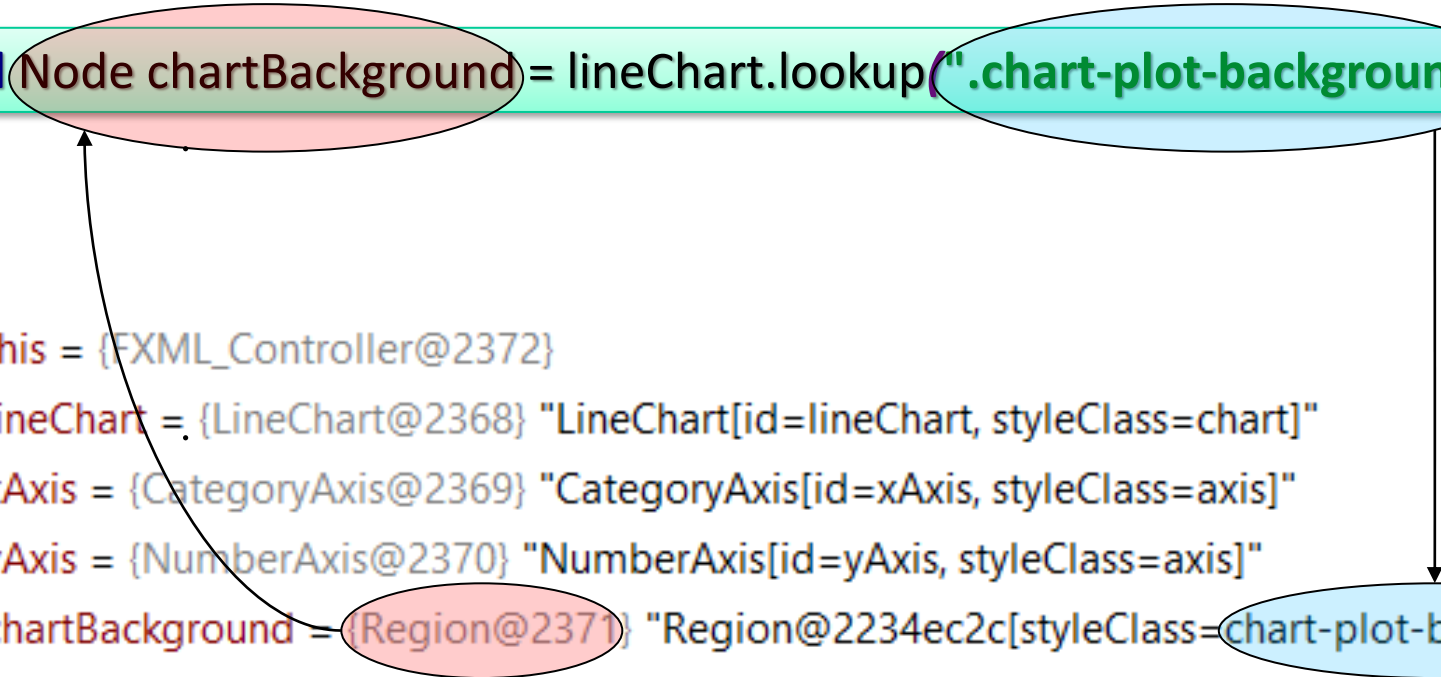
>  this = {FXML\_Controller@2372}

>  lineChart = {LineChart@2368} "LineChart[id=lineChart, styleClass=chart]"

>  xAxis = {CategoryAxis@2369} "CategoryAxis[id=xAxis, styleClass=axis]"

>  yAxis = {NumberAxis@2370} "NumberAxis[id=yAxis, styleClass=axis]"

>  chartBackground = {Region@2371} "Region@2234ec2c[styleClass=chart-plot-background]"



# Afficher les coordonnées

- *Rendre les nœuds (avec tous leurs enfants) sensibles aux événements de la souris:*

```
for ( Node n : chartBackground.getParent().getChildrenUnmodifiable() ) {  
    if (( n != chartBackground )) {  
        n.setMouseTransparent( true );  
    } else n.setMouseTransparent( false );  
}
```

*La propriété mouseTransparent rend le "nœud (avec tous ses enfants) complètement transparent aux événements de souris."*

*Si le curseur de la souris n'est pas visible:*

```
lineChart.setCursor(Cursor.CROSSHAIR);
```

# Afficher les coordonnées

## ➤ Lier les « *EventHandler* » de la souris:

```
chartBackground.setOnMouseEntered(event ->cursorCoords.setVisible(true));  
chartBackground.setOnMouseExited(event ->cursorCoords.setVisible(false) );  
chartBackground.setOnMouseMoved(event ->cursorCoords.setText(  
    String.format("%.2f, %.2f",  
        xAxis.getValueForDisplay(event.getX()),  
        yAxis.getValueForDisplay(event.getY())));
```

### getValueForDisplay

```
public abstract T getValueForDisplay(double displayPosition)
```

Get the data value for the given display position on this axis. If the axis is a CategoryAxis this will be the nearest value.

#### Parameters:

`displayPosition` - A pixel position on this axis

#### Returns:

the nearest data value to the given pixel position or null if not on axis;


# LineChart affichage temps réel:

- *A chaque évènement (matériel, temporel...) :*



```
serie.getData().add(new XYChart.Data<>(x_valeur,y_valeur));
```

- *Si on veut toujours le même nombre de points:*



```
final int TAILLE_FENETRE = 15;  
...  
...  
serie.getData().add(new XYChart.Data<>(x_valeur,y_valeur));  
if (serie.getData().size() > TAILLE_FENETRE)  
    serie.getData().remove(0);  
...
```