



# PRÁCTICA 6. DIAGRAMAS EN JAVAFX 8

---

## DOCENCIA VIRTUAL

### Finalidad:

Prestación del servicio Público de educación superior (art. 1 LOU)

### Responsable:

Universitat Politècnica de València.

**Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:**

<http://www.upv.es/contenidos/DPD/>

### Propiedad intelectual:

Uso exclusivo en el entorno de aula virtual.

Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil



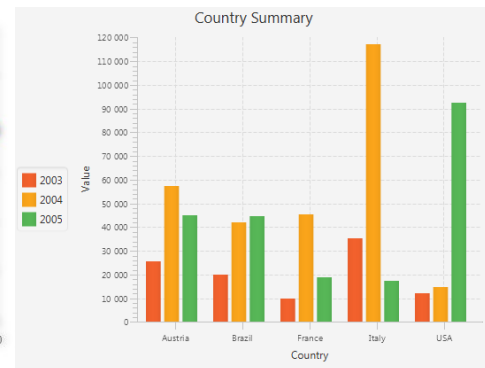
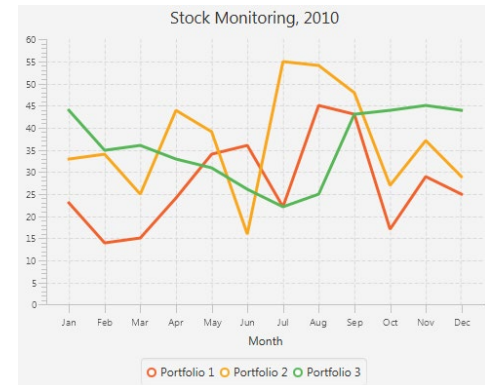
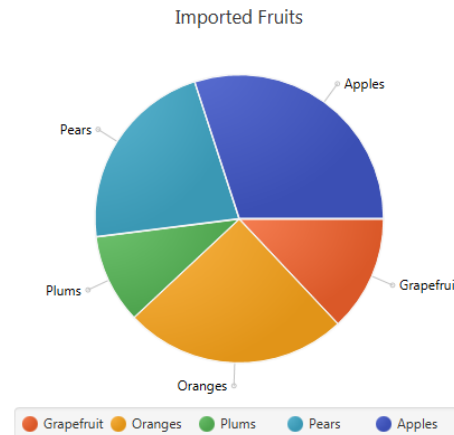
UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Interfaces Persona  
Computador  
Depto. Sistemas  
Informáticos y  
Computación  
UPV

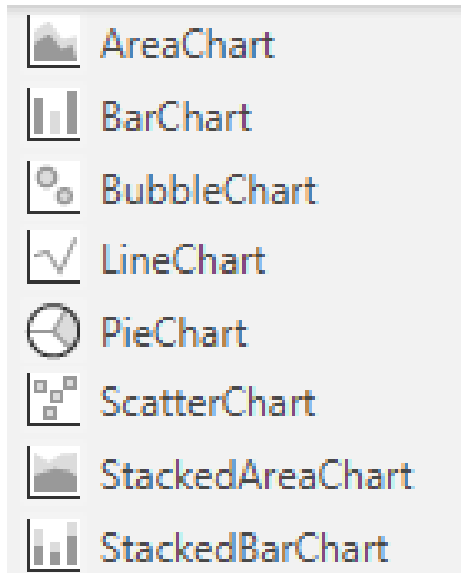
# Índice

- Diagramas en JavaFX 8
  - Diagrama de tarta
  - Diagrama de líneas
  - Diagrama de áreas
  - Diagrama de burbujas
  - Diagrama de puntos
  - Diagrama de barras
  - Operaciones con diagramas
- CSS en los diagramas
- Ejemplo



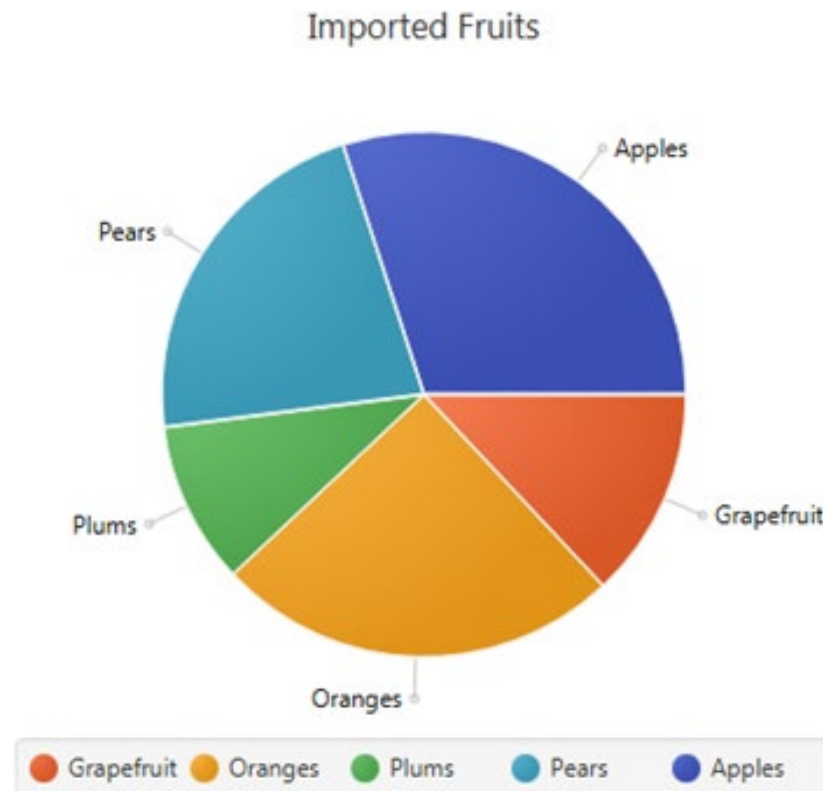
# Introducción

- Los diagramas se pueden programar por código o utilizando la herramienta Scene Builder
- Se recomienda utilizar el Scene Builder
- Todos los diagramas visualizan datos 2D, excepto el primero, el diagrama de tarta



# Diagrama de tarta

- Visualiza datos como un círculo dividido en trozos que representan un porcentaje correspondiente a un valor



# Diagrama de tarta

- Para visualizar el diagrama:
  - Crear un *PieChart*
  - Crear una lista observable para guardar los datos del diagrama
  - Los datos son del tipo *PieChart.Data* y cada uno contiene un *String* y el valor numérico asociado a ese *String*

```
PieChart chart = new PieChart();
ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList(
    new PieChart.Data("Grapefruit", 13),
    new PieChart.Data("Oranges", 25),
    new PieChart.Data("Plums", 10),
    new PieChart.Data("Pears", 22),
    new PieChart.Data("Apples", 30)
);
chart.setData(pieChartData);
chart.setTitle("Imported Fruits");
```

# Diagrama de tarta

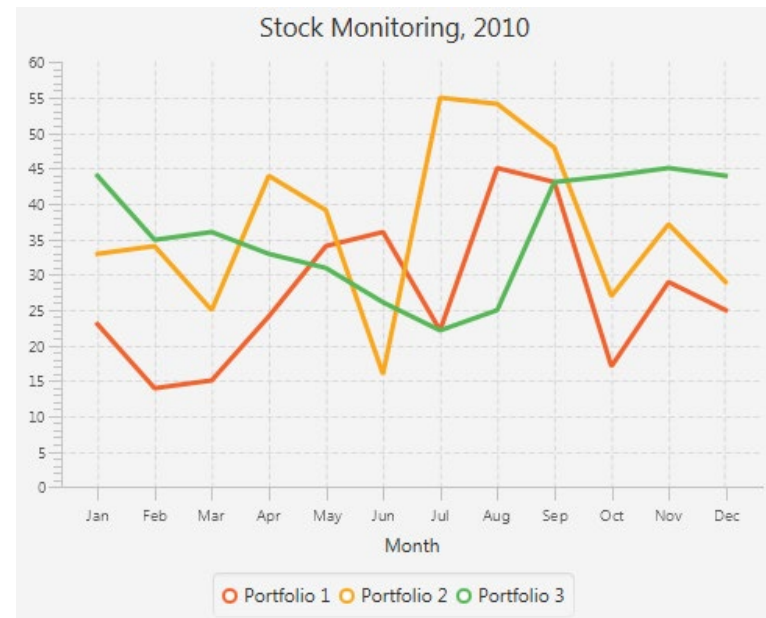
- Recorrer los elementos de la lista observable asociada a un PieChart y sumar 1 a su valor actual:

```
PieChart chart = new PieChart();
ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList(
    new PieChart.Data("Grapefruit", 13),
    new PieChart.Data("Oranges", 25),
    new PieChart.Data("Plums", 10),
    new PieChart.Data("Pears", 22),
    new PieChart.Data("Apples", 30)
);
chart.setData(pieChartData);
chart.setTitle("Imported Fruits");

for (PieChart.Data item : pieChartData) {
    String name = item.getName(); // "Grapefruit", "Oranges", etc.
    int valorActual = item.getPieValue() ; // 13, 25, etc.
    item.setPieValue(valorActual + 1); // Incrementar en 1 el valor actual
    // También Podemos cambiar el nombre con item.setName(nuevonombre)
}
```

# Diagrama de líneas

- Representa una serie de puntos conectados por rectas
- Tiene dos ejes, los puntos y las rectas, una leyenda y, opcionalmente, un título
- Puede tener una o más series de datos



# Diagrama de líneas

- Con una sola serie de datos:

```
// Crear ejes
```

```
CategoryAxis xAxis = new CategoryAxis();  
NumberAxis yAxis = new NumberAxis();  
xAxis.setLabel("Month");
```

```
ObservableList<XYChart.Data <String, Number> > lineChartData =  
    FXCollections.observableArrayList(  
        new XYChart.Data("Jan", 23),  
        new XYChart.Data("Feb", 14),  
        ...  
        new XYChart.Data("Nov", 29),  
        new XYChart.Data("Dec", 25)  
    );
```

```
XYChart.Series serie = new XYChart.Series(lineChartData); // Rellenar la serie con la  
ObservableList creada anteriormente  
serie.setName("My portfolio"); // Leyenda
```

```
LineChart<String, Number> lineChart = new LineChart<String, Number>(xAxis,yAxis);  
lineChart.setTitle("Stock Monitoring, 2010");  
lineChart.getData().add(serie);
```

```
Scene scene = new Scene(lineChart,800,600);
```





# XYChart.Series

- Recorrer los elementos de la lista observable asociada a una `XYChart.Series` y sumar 1 a su valor actual:

```
ObservableList<XYChart.Data<String, Number>> lineChartData =
    FXCollections.observableArrayList(
        new XYChart.Data("Jan", 23),
        new XYChart.Data("Feb", 14),
        ...
        new XYChart.Data("Nov", 29),
        new XYChart.Data("Dec", 25)
    );
XYChart.Series serie = new XYChart.Series(lineChartData); // Rellenar la serie con la
ObservableList creada anteriormente

for (XYChart.Data<String, Number> item : serie.getData() ){ // Podemos recorrer tanto
lineChartData como serie.getData() ya que ambas son la misma ObservableList
    String x = item.getXValue(); // "Jan", "Feb", etc.
    int valorActual = item.getYValue().intValue() ; // 23, 14, etc.
    item.setYValue(valorActual + 1);
}
```

# Diagrama de líneas

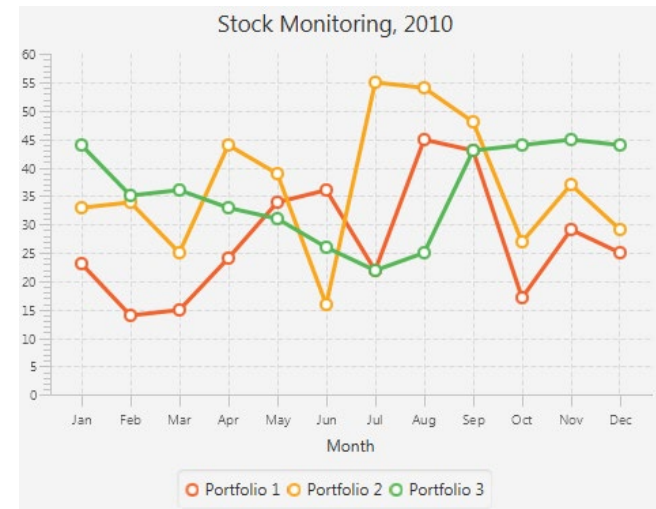
- Con tres series de datos:

```
// serie1 definida en las transparencias anteriores

// Constructor sin indicar la ObservableList
XYChart.Series serie2 = new XYChart.Series();
serie2.setName("Portfolio 2"); // Leyenda
// Rellenar la ObservableList interna de XYChart.Series
    elemento a elemento
serie2.getData().add(new XYChart.Data("Jan", 33));
...
serie2.getData().add(new XYChart.Data("Dec", 25));

XYChart.Series serie3 = new XYChart.Series();
serie3.setName("Portfolio 3"); // Leyenda
serie3.getData().add(new XYChart.Data("Jan", 44));
...
serie3.getData().add(new XYChart.Data("Dec", 44));

Scene scene = new Scene(lineChart, 800, 600);
lineChart.getData().addAll(serie1, serie2, serie3);
```



# Diagrama de líneas

- Opciones:

- Poner el eje X en la parte superior

```
xAxis.setSide(Side.TOP);
```

- Eliminar las marcas de los puntos de la gráfica

```
lineChart.setCreateSymbols(false);
```

- Utilizar números en el eje X

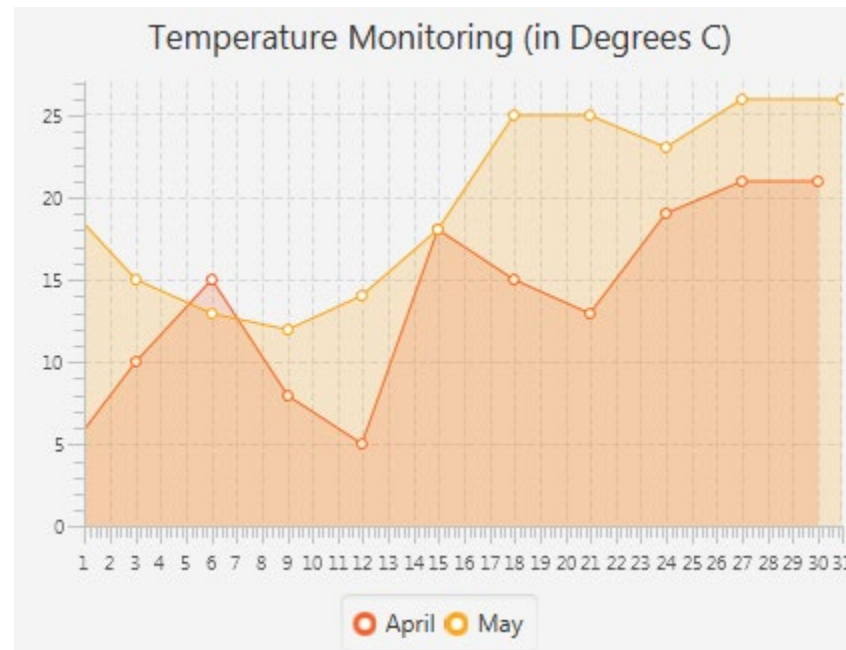
```
NumberAxis xAxis = new NumberAxis();
NumberAxis yAxis = new NumberAxis();
// Crear el diagrama
LineChart<Number, Number> lineChart =
    new LineChart<Number, Number>(xAxis,yAxis);

// Crear la serie
XYChart.Series series = new XYChart.Series();
// Rellenar la serie con un nuevo elemento
series.getData().add(new XYChart.Data(1, 23));
...
```



# Diagrama de áreas

- Similar al diagrama de líneas pero con el área bajo las líneas rellenada
- Permiten también una o varias series de datos



# Diagrama de áreas

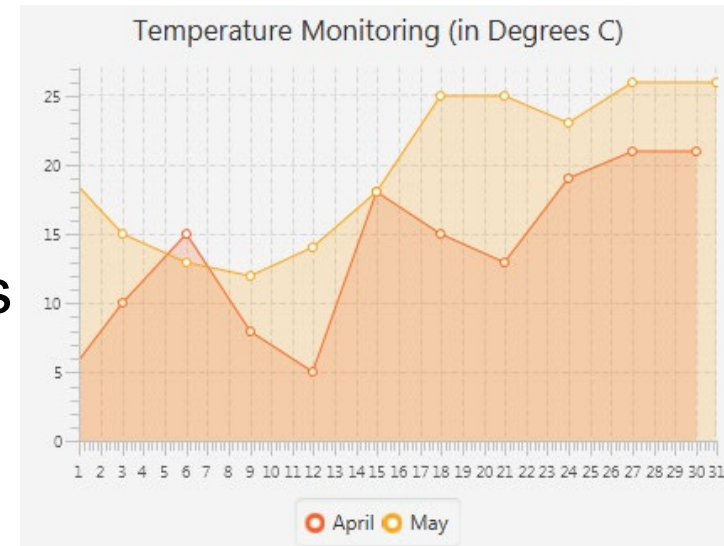
- Se crea como el diagrama de líneas

```
stage.setTitle("Area Chart Sample");
NumberAxis xAxis = new NumberAxis(1, 31, 1);
NumberAxis yAxis = new NumberAxis();
AreaChart<Number, Number> ac =
    new AreaChart<>(xAxis,yAxis);
ac.setTitle("Temperature Monitoring (in Degrees C)");
```

```
XYChart.Series seriesApril= new XYChart.Series();
seriesApril.setName("April"); // Leyenda
seriesApril.getData().add(new XYChart.Data(1, 4));
...
seriesApril.getData().add(new XYChart.Data(30, 21));
```

```
XYChart.Series seriesMay = new XYChart.Series();
seriesMay.setName("May"); // Leyenda
seriesMay.getData().add(new XYChart.Data(1, 20));
...
seriesMay.getData().add(new XYChart.Data(31, 26));
```

```
ac.getData().addAll(seriesApril, seriesMay);
```



# Diagrama de áreas

- Opciones

- En la declaración del eje X

```
NumberAxis xAxis = new NumberAxis(1, 31, 1);
```

se indican el mínimo (1), el máximo (31) y la distancia entre marcas (1)

- También se puede hacer explícitamente

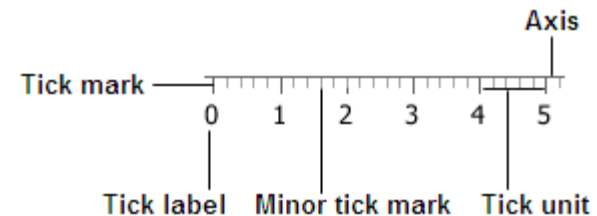
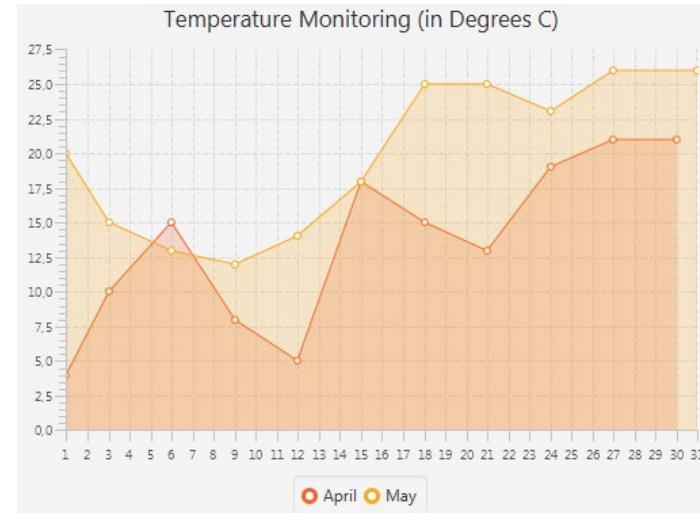
```
xAxis.setLowerBound(1);  
xAxis.setUpperBound(31);  
xAxis.setTickUnit(1);
```

- Para eliminar las marcas pequeñas

```
xAxis.setMinorTickCount(0);
```

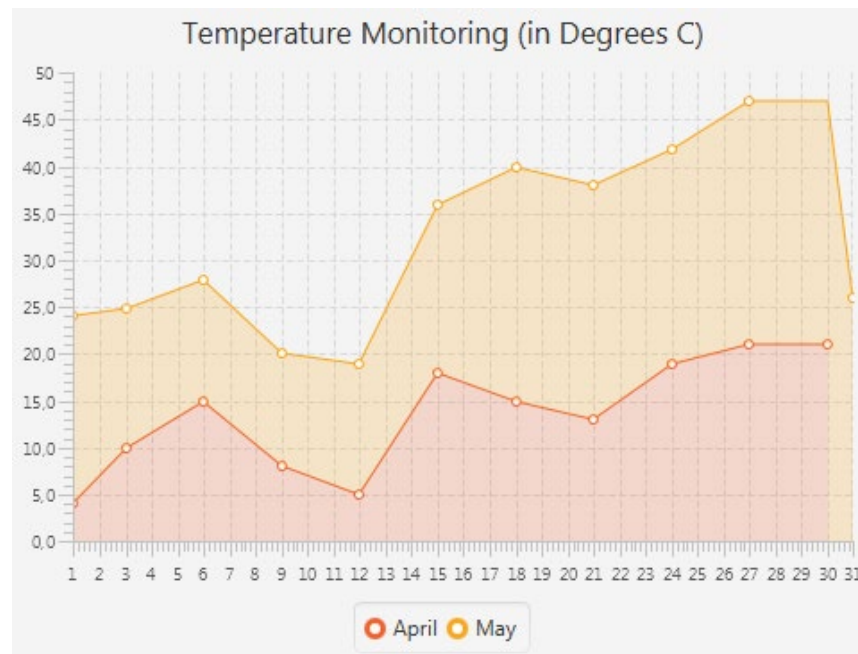
- Para eliminar marcas y etiquetas, respectivamente:

```
xAxis.setTickMarkVisible(false);  
xAxis.setTickLabelsVisible(false);
```



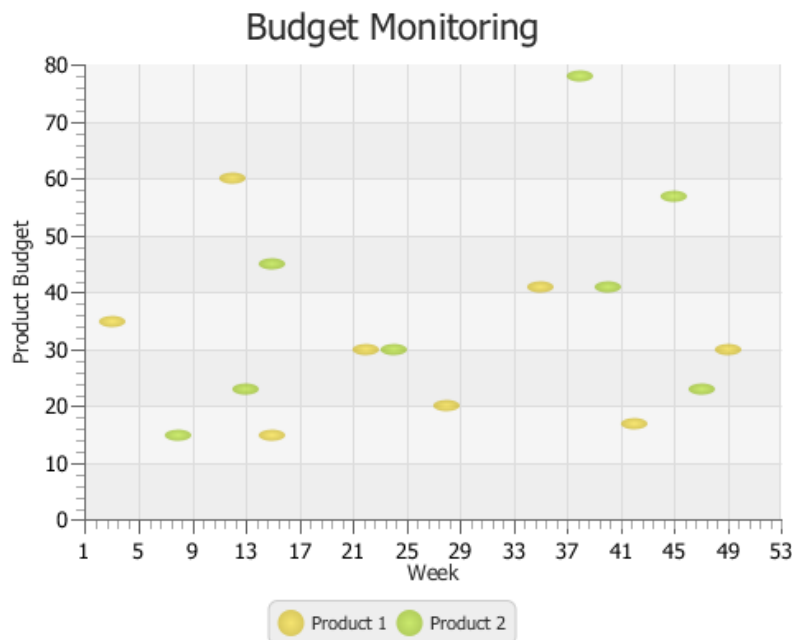
# Diagrama de áreas apiladas

- Es un diagrama de áreas donde la segunda serie y sucesivas se visualizan acumulando los valores de las series anteriores
- Utiliza la clase *StackedAreaChart* en lugar de *AreaChart*

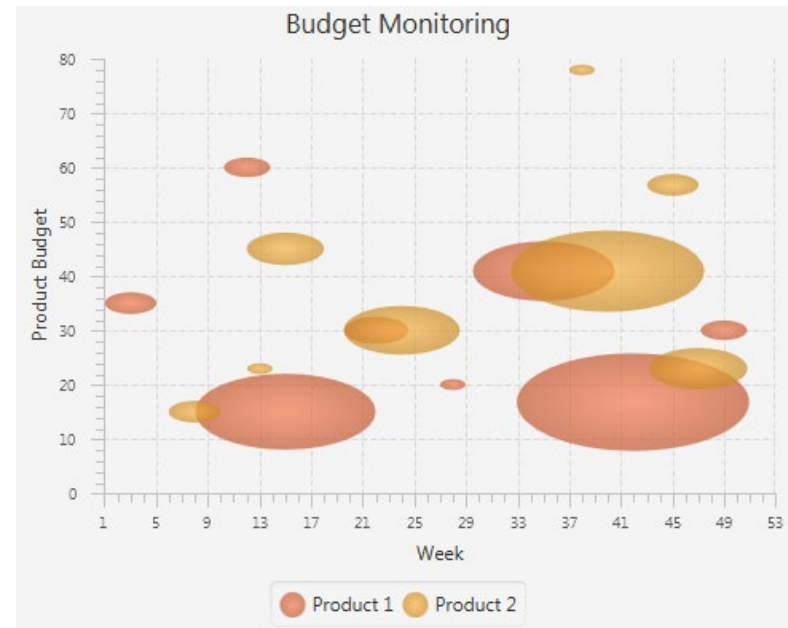


# Diagrama de burbujas

- Diagrama bidimensional que dibuja burbujas para los puntos de una serie
- Las burbujas pueden tener un radio distinto



`XYChart.Data(x, y)`



`XYChart.Data(x, y, radio)`



# Diagrama de burbujas

- El radio de cada burbuja se especifica como un parámetro más en la definición de los *XYChart.Data()*
- Debe ser de tipo **Number**

```
NumberAxis xAxis = new NumberAxis(1, 53, 4);  
NumberAxis yAxis = new NumberAxis(0, 80, 10);  
BubbleChart<Number, Number> blc = new BubbleChart<>(xAxis,yAxis);
```

```
XYChart.Series series1 = new XYChart.Series();  
series1.setName("Product 1"); // Leyenda  
series1.getData().add(new XYChart.Data(3, 35, 2));  
...
```

```
XYChart.Series series2 = new XYChart.Series();  
series2.setName("Product 2"); // Leyenda  
series2.getData().add(new XYChart.Data(8, 15, 2));  
...
```

```
blc.getData().addAll(series1, series2);
```

Radio de la burbuja



# XYChart.Series - ExtraValue

- Acceder al radio de la burbuja e incrementarlo en 1:

```
XYChart.Series series1 = new XYChart.Series();
series1.setName("Product 1"); // Leyenda
series1.getData().add(new XYChart.Data(3, 35, 2));
...

for (XYChart.Data<Number, Number> item : series1.getData() ){
    int x = item.getXValue().intValue(); // 3, etc.
    int y = item.getYValue().intValue() ; // 35, etc.
    int radioActual = item.getExtraValue(); // 2, etc.
    item.setExtraValue(radioActual + 1);
}
```

# Diagrama de burbujas

- Opciones:

- Formatear las etiquetas de un eje

```
yAxis.setTickLabelFormatter(new  
    NumberAxis.DefaultFormatter(yAxis,"$ ",null));
```

Prefijo

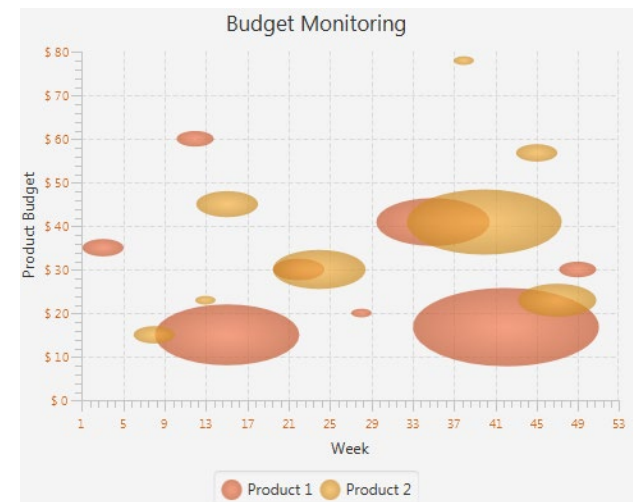
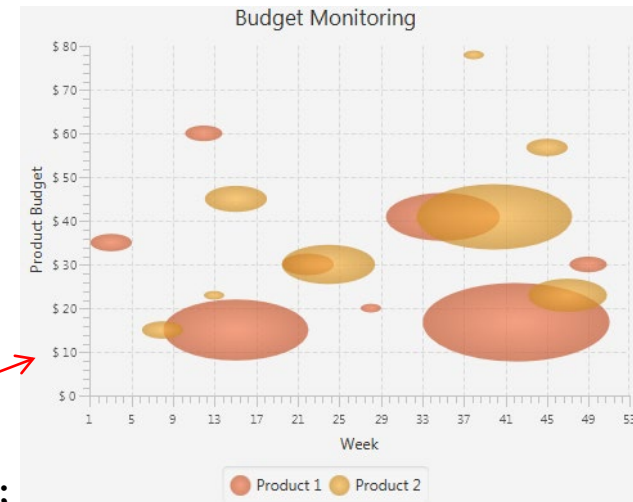
Sufijo

- Eliminar las líneas del fondo del diagrama

```
blc.setHorizontalGridLinesVisible(false);  
blc.setVerticalGridLinesVisible(false);
```

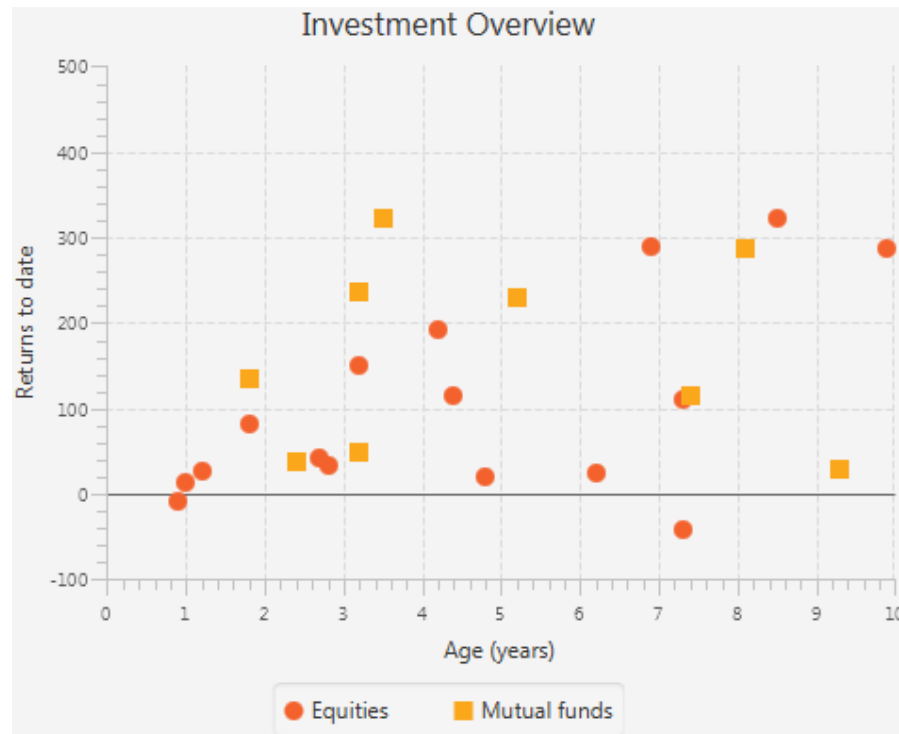
- Cambiar el color de las etiquetas de los ejes

```
xAxis.setTickLabelFill(Color.CHOCOLATE);  
yAxis.setTickLabelFill(Color.CHOCOLATE);
```



# Diagrama de puntos

- Diagrama bidimensional formado por puntos definidos por un par de valores X e Y
- Se crean igual que el resto de diagramas bidimensionales



# Diagrama de puntos

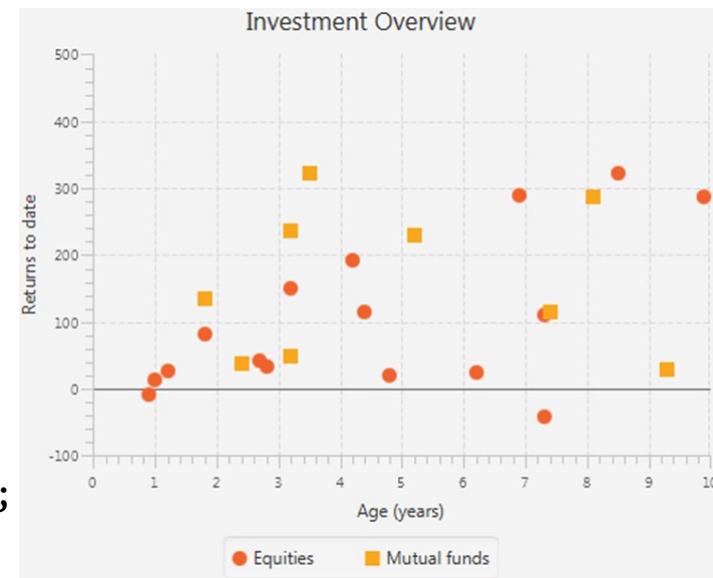
- Se crea igual que el resto de diagramas bidimensionales

```
NumberAxis xAxis = new NumberAxis(0, 10, 1);
NumberAxis yAxis = new NumberAxis(-100, 500, 100);
ScatterChart<Number, Number> sc = new
    ScatterChart<>(xAxis,yAxis);
xAxis.setLabel("Age (years)");
yAxis.setLabel("Returns to date");
sc.setTitle("Investment Overview");
```

```
XYChart.Series series1 = new XYChart.Series();
series1.setName("Equities"); // Leyenda
series1.getData().add(new XYChart.Data(4.2, 193.2));
...
```

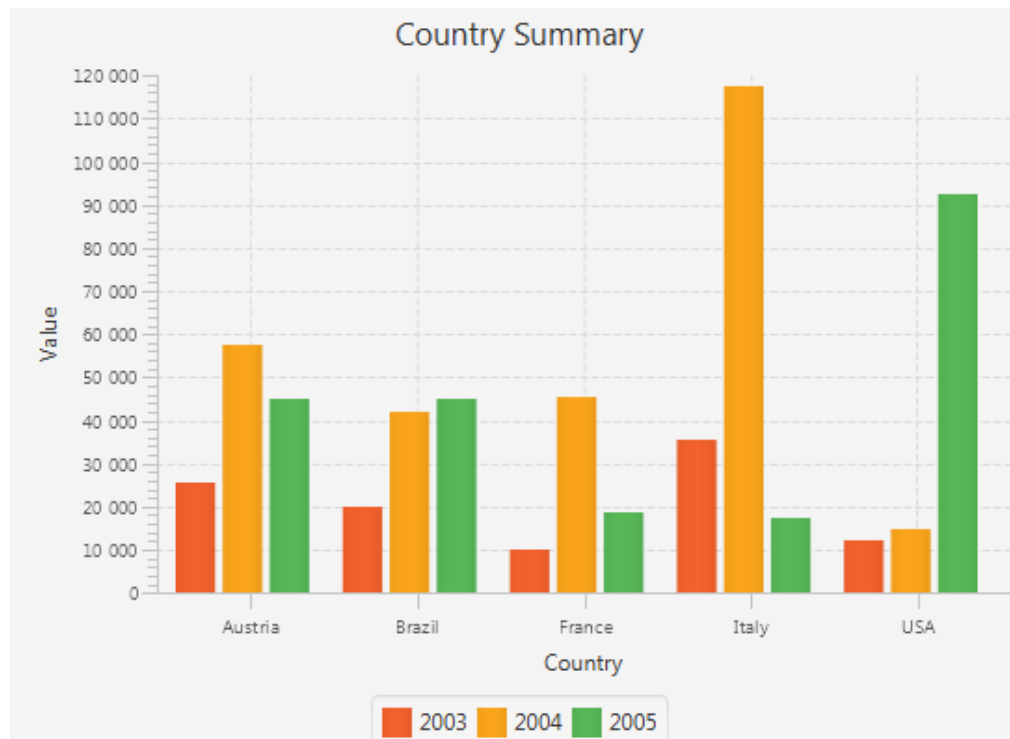
```
XYChart.Series series2 = new XYChart.Series();
series2.setName("Mutual funds"); // Leyenda
series2.getData().add(new XYChart.Data(5.2, 229.2));
...
```

```
sc.getData().addAll(series1, series2);
```



# Diagrama de barras

- Es un diagrama bidimensional donde los datos se representan como barras
- Soportan una o varias series

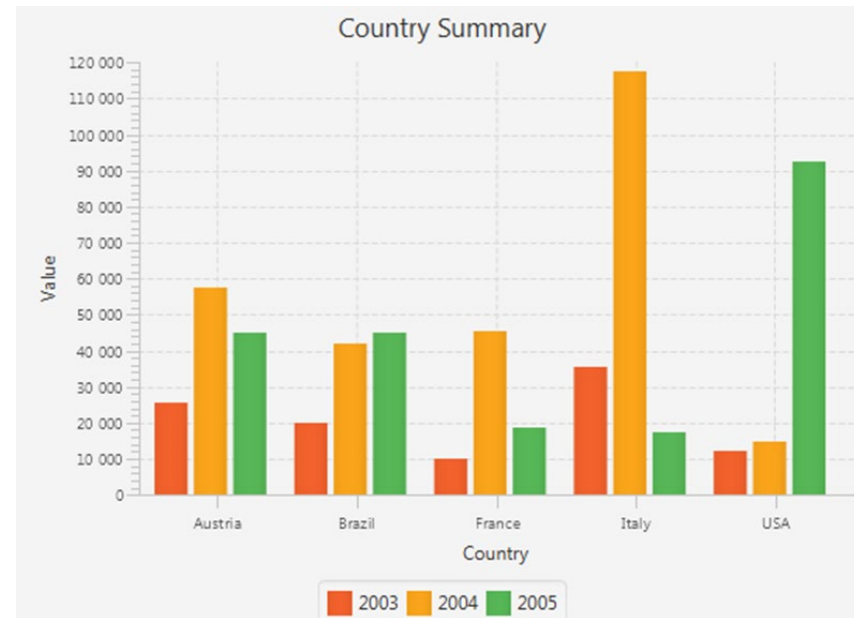


# Diagrama de barras

- Código del ejemplo

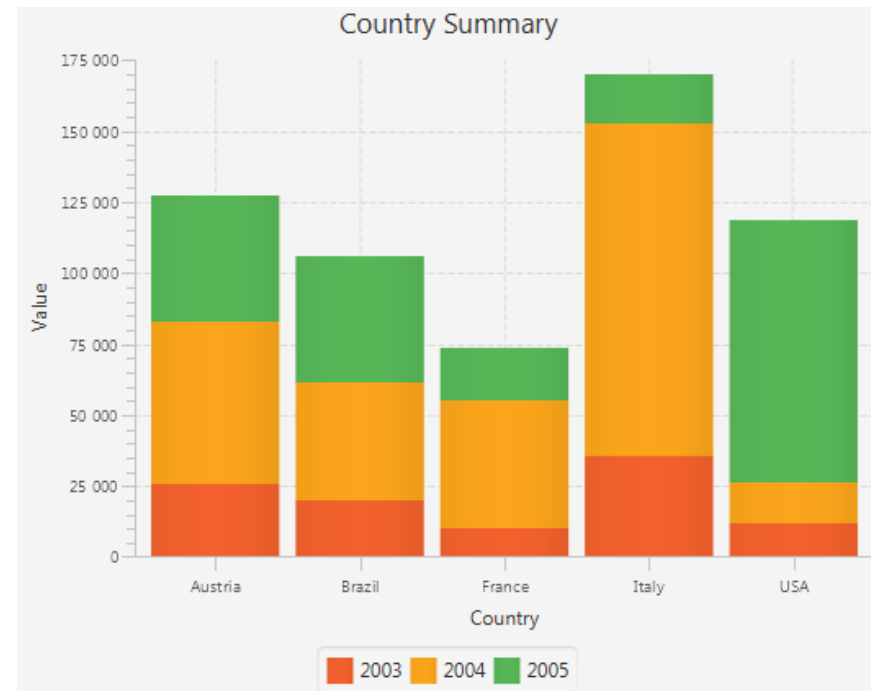
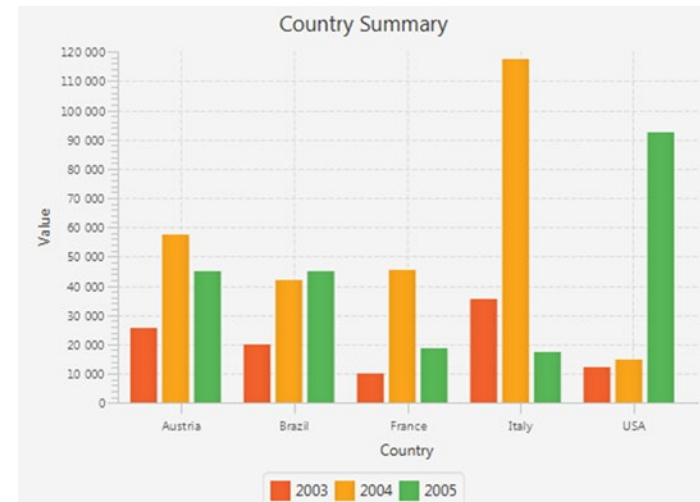
```
CategoryAxis xAxis = new CategoryAxis();
NumberAxis yAxis = new NumberAxis();
BarChart<String, Number> barChart =
    new BarChart<>(xAxis,yAxis);
barChart.setTitle("Country Summary");
xAxis.setLabel("Country");
yAxis.setLabel("Value");
```

```
XYChart.Series series1 = new XYChart.Series();
series1.setName("2003"); // Leyenda
series1.getData().add(new XYChart.Data("Austria", 25601.34));
...
XYChart.Series series2 = new XYChart.Series();
series2.setName("2004"); // Leyenda
series2.getData().add(new XYChart.Data("Austria", 57401.85));
...
XYChart.Series series3 = new XYChart.Series();
series3.setName("2005"); // Leyenda
series3.getData().add(new XYChart.Data("Austria", 45000.65));
...
barChart.getData().addAll(series1, series2, series3);
```



# Diagrama de barras

- Opciones:
  - Fijar separación entre barras  
`barChart.setBarGap(3);`
  - Fijar separación entre categorías  
`barChart.setCategoryGap(20);`
- Diagrama de barras apiladas:
  - En el eje vertical las áreas de las barras muestran valores acumulados
  - En el ejemplo: el valor de 125000 para Austria es el acumulado de 2003, 2004 y 2005
  - Utilizar *StackedBarChart*



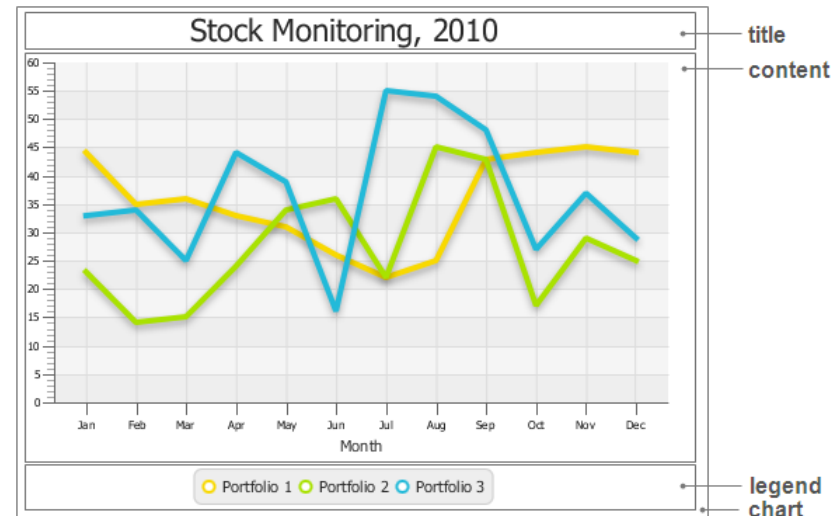


# Diagramas en JavaFX

- Se recomienda crearlos desde Scene Builder
- En Scene Builder en los diagramas bidimensionales:
  - El eje X es siempre un *CategoryAxis*
  - El eje Y es un *NumberAxis*
  - Por lo tanto, los datos son *XYChart.Data(String, Number)*
  - Excepto para el *BubbleChart* y el *StackedAreaChart* donde los dos ejes son *NumberAxis*
  - Puedes editar el FXML a mano para cambiar la definición de las áreas y conseguir tener los dos ejes como *Number*.
- Ojo: Los **datos** que vayan a mostrarse, deben estar en **listas observables** y **cada lista observable solo puede enlazarse a un único** diagrama.
- Más información sobre diagramas, así como los ejemplos completos, en:
  - <http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/charts.htm>

# CSS en los diagramas

- Todos los diagramas tienen propiedades en común:
  - `.chart`
  - `.chart-content`
  - `.chart-title`
  - `.chart-legend`
- A cada propiedad se le pueden cambiar distintas características:
  - Márgenes: `-fx-padding`
  - Background (color o imagen): `-fx-background-color` , `-fx-background-image`
  - Fuente: `fx-font-size`
  - Color del texto: `fx-text-fill`
- Más información y ejemplos: <https://docs.oracle.com/javafx/2/charts/css-styles.htm>
- Fichero oficial de CSS Modena disponible [aquí](#)
- Fichero oficial de CSS Caspian disponible [aquí](#)



# CSS en los diagramas: Colores

- Por defecto, los temas oficiales definen los 8 primeros colores de las series para todos los diagramas (disponibles en el repositorio [Github](#))

```
/* Chart Color Palette */  
CHART_COLOR_1: #f3622d;  
CHART_COLOR_2: #fba71b;  
CHART_COLOR_3: #57b757;  
CHART_COLOR_4: #41a9c9;  
CHART_COLOR_5: #4258c9;  
CHART_COLOR_6: #9a42c8;  
CHART_COLOR_7: #c84164;  
CHART_COLOR_8: #888888;
```

- En esa misma hoja de estilo, se indica el color para todos los diagramas disponibles.

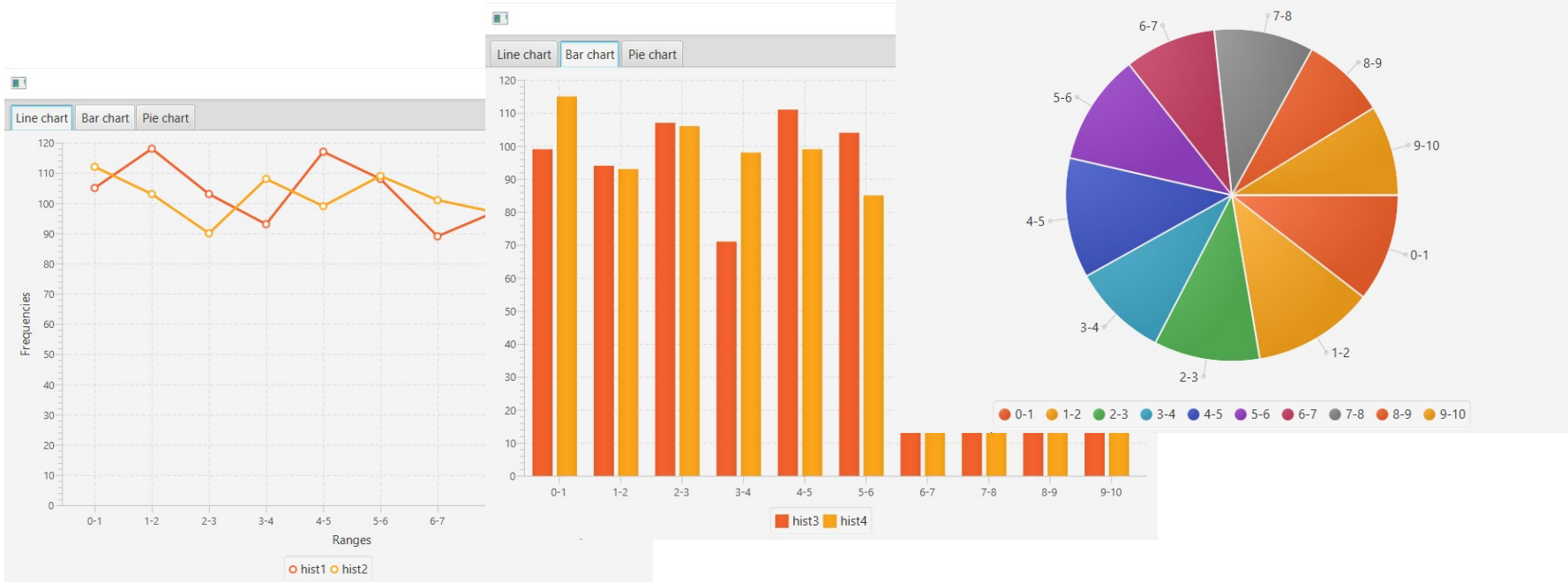
```
.default-color0.chart-pie { -fx-pie-color: CHART_COLOR_1; }  
.default-color0.chart-series-line { -fx-stroke: CHART_COLOR_1; }  
.default-color0.chart-series-area-line { -fx-stroke: CHART_COLOR_1; }  
.default-color0.chart-series-area-fill { -fx-fill: CHART_COLOR_1_TRANS_20; }  
.default-color0.chart-bubble { -fx-bubble-fill: CHART_COLOR_1_TRANS_70; }  
.default-color0.chart-bar { -fx-bar-fill: CHART_COLOR_1; }
```

- Colores disponibles:

[https://docs.oracle.com/cd/E17802\\_01/javafx/javafx/1.3/docs/api/javafx.scene/doc-files/cssref.html#typecolor](https://docs.oracle.com/cd/E17802_01/javafx/javafx/1.3/docs/api/javafx.scene/doc-files/cssref.html#typecolor)

# Ejemplo de diagrama

- Vamos a crear un diagrama con el histograma de un conjunto de números aleatorios
- Como los números son aleatorios el histograma revelará una distribución casi uniforme



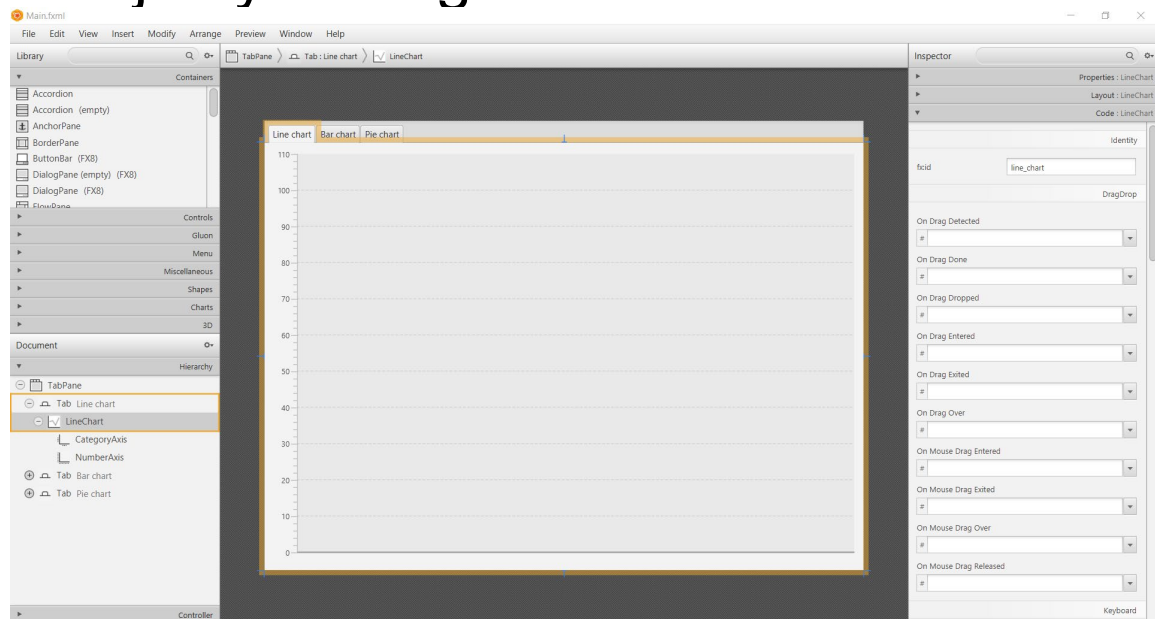
# Ejemplo de diagrama - LineChart

- Crear un TabPane y 3 Tabs
- Para crear el diagrama desde Scene Builder basta con elegirlo entre los *Charts* y arrastrarlo
- Asignar los *fx:id*'s a los ejes y al diagrama

```
@FXML
private LineChart<String,Number>
    lineChart;
```

```
@FXML
private CategoryAxis
    lineChart_xAxis;
```

```
@FXML
private NumberAxis lineChart_yAxis;
```



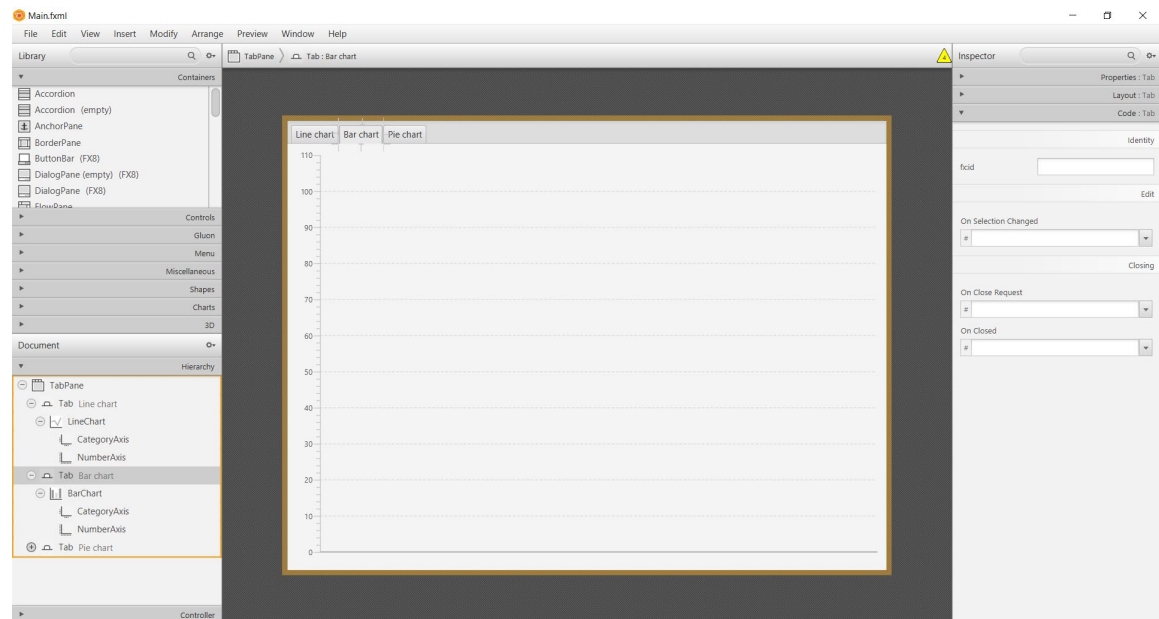
# Ejemplo de diagrama - BarChart

- Para crear el diagrama desde Scene Builder basta con elegirlo entre los *Charts* y arrastrarlo
- Asignar los *fx:id*'s a los ejes y al diagrama

```
@FXML
private BarChart<String,Number>
    barChart;
```

```
@FXML
private CategoryAxis
    barChart_xAxis;
```

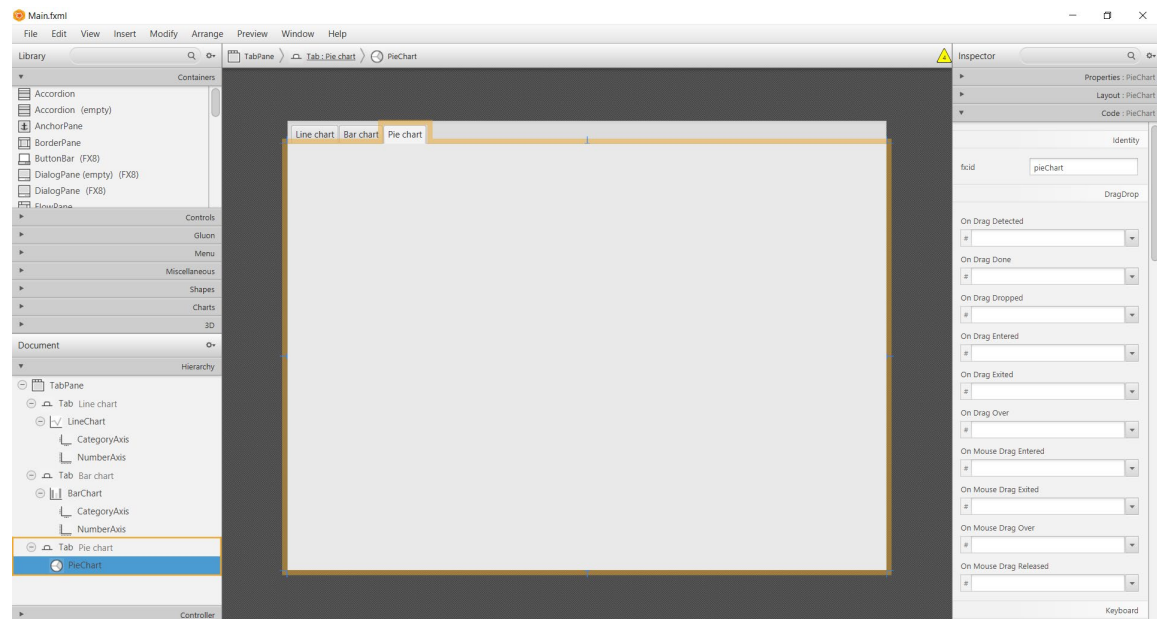
```
@FXML
private NumberAxis barChart_yAxis;
```



# Ejemplo de diagrama - PieChart

- Para crear el diagrama desde Scene Builder basta con elegirlo entre los *Charts* y arrastrarlo
- Asignar el *fx:id*'s al diagrama

@FXML  
private PieChart pieChart;



# Ejemplo – Generar histograma

- Generar los números aleatorios

```
public void initialize(URL url, ResourceBundle rb) {
```

```
    int hist[] = generate_histogram(10);  
    int hist2[] = generate_histogram(10);  
    int hist3[] = generate_histogram(10);  
    int hist4[] = generate_histogram(10);
```

...

```
public int[] generate_histogram(int lenght ){  
    int hist[] = new int[lenght];  
  
    // Generate histogram  
    for (int j = 0; j < 1000; j++) {  
        double value = Math.random() * lenght;  
        for (int i = 0; i < hist.length; i++){  
            if (i <= value && value < i+1) {  
                hist[(int)i]++;  
                break;  
            }  
        }  
    }  
    return hist;  
}
```



# Ejemplo – Generar histograma

- Inicializar y rellenar las listas observables.

...

```
ObservableList<PieChart.Data> pie_chart_data = FXCollections.observableArrayList();
ObservableList<XYChart.Data<String,Number>> data = FXCollections.observableArrayList();
ObservableList<XYChart.Data<String,Number>> data2 = FXCollections.observableArrayList();
ObservableList<XYChart.Data<String,Number>> data3 = FXCollections.observableArrayList();
ObservableList<XYChart.Data<String,Number>> data4 = FXCollections.observableArrayList();
```

```
for (int i = 0; i < hist.length; i++)
{
    String aux = i + "-" + (i+1);
    data.add(new XYChart.Data<String,Number>(aux, hist[i]));
    data2.add(new XYChart.Data<String,Number>(aux, hist2[i]));
    data3.add(new XYChart.Data<String,Number>(aux, hist3[i]));
    data4.add(new XYChart.Data<String,Number>(aux, hist4[i]));
    pie_chart_data.add ( new PieChart.Data(aux, hist[i]) );
}
```

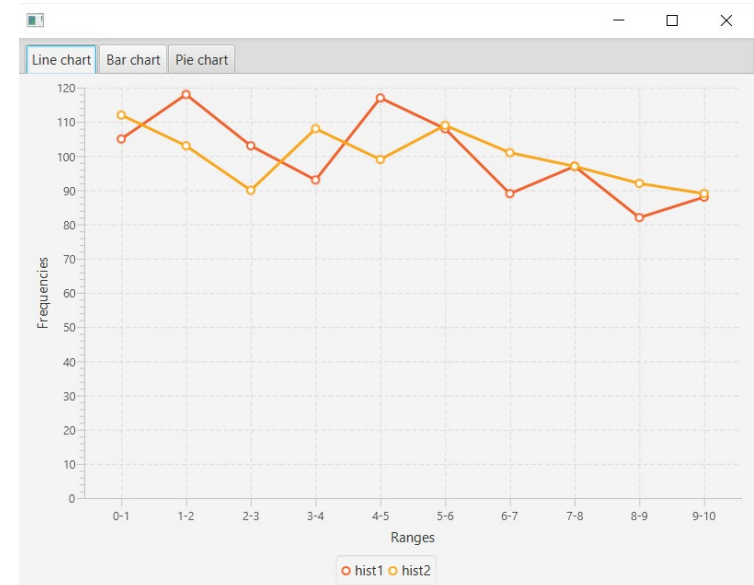
...

# Ejemplo – LineChart

- Crear las `XYChart.Series` a partir de las `ObservableList`.
- Añadir las `XYChart.Series` al diagrama y cambiar el nombre de los ejes

```
// XYChart.Series used in LineChart
XYChart.Series s1 = new XYChart.Series(data);
s1.setName("hist1");
XYChart.Series s2 = new XYChart.Series(data2);
s2.setName("hist2");

// Populate the LineChart
lineChart.getData().addAll(s1, s2);
// LineChart Axes names
lineChart_xAxis.setLabel("Ranges");
lineChart_yAxis.setLabel("Frequencies");
```

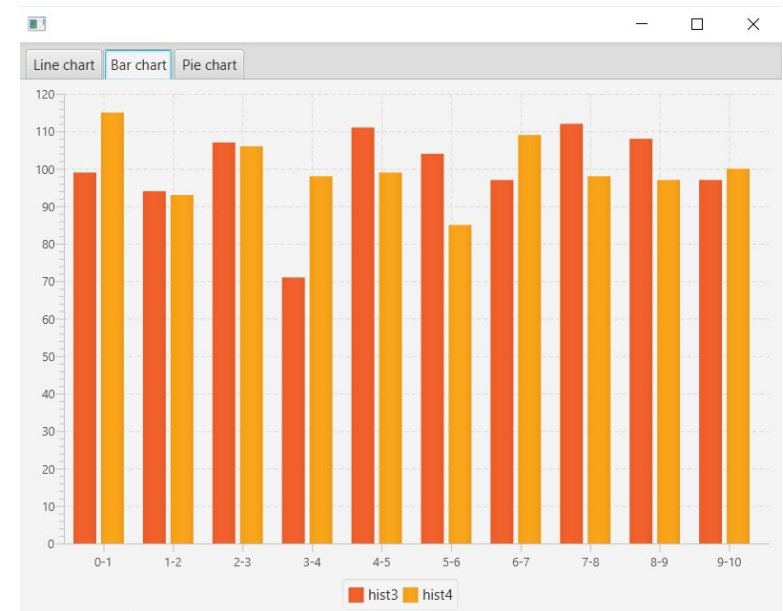


# Ejemplo – BarChart

- Crear las `XYChart.Series` a partir de las `ObservableList`.
- Añadir las `XYChart.Series` al diagrama y cambiar el nombre de los ejes

```
// XYChart.Series used in BarChart
XYChart.Series s3 = new XYChart.Series(data3);
s1.setName("hist3");
XYChart.Series s4 = new XYChart.Series(data4);
s2.setName("hist2");

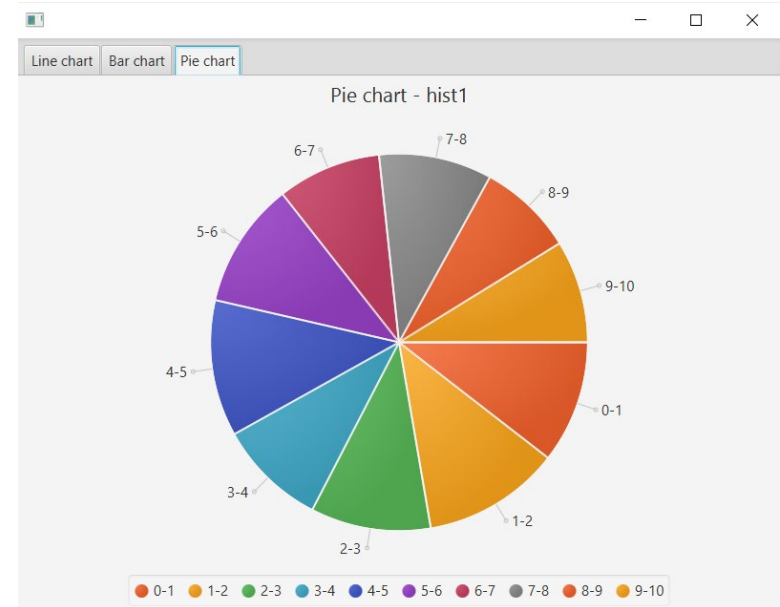
// Populate the BarChart
barChart.getData().addAll(s3, s4);
// BarChart Axes names
barChart_xAxis.setLabel("Ranges");
barChart_yAxis.setLabel("Frequencies");
```



# Ejemplo – PieChart

- Añadir la `ObservableList<PieChart.Data>` al `PieChart`
- Cambiar el título del gráfico

```
pieChart.setData(pie_chart_data);  
pieChart.setTitle("Pie chart - hist1");
```



# Bibliografía

- <https://docs.oracle.com/javafx/2/charts/css-styles.htm>
- <http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/charts.htm>
- <https://github.com/openjdk/jfx/blob/master/modules/javafx.controls/src/main/resources/com/sun/javafx/scene/control/skin/modena/modena.css>
- <https://github.com/openjdk/jfx/blob/master/modules/javafx.controls/src/main/resources/com/sun/javafx/scene/control/skin/caspian/caspian.css>