جامعة طرابلس
كلية تقنية المعلومات
قسم هندسة البرمجيات

البرمجة المرئية Visual Programming
ربيع 2025

المحاضرة السادسة – الرسم البياني

مواضيع المحاضرة

- فهم الرسومات البيانية **Charts**
- ما هو الرسم البياني **chart؟**
- ما هو chart API في JavaFX؟
- كيفية إنشاء أنواع مختلفة من الرسومات البيانية Charts.
- كيفية تنسيق الرسم البياني باستخدام CSS

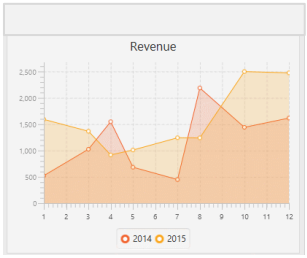
ما هو الرسم البياني Chart؟

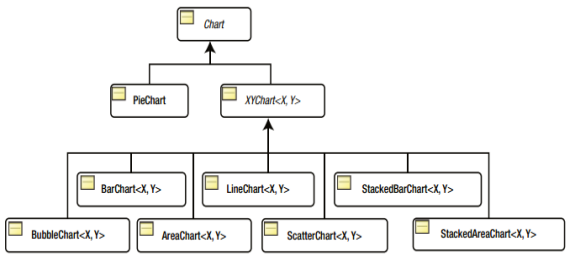
- الرسم البياني هو تمثيل رسومي للبيانات data. توفر الرسومات البيانية طريقة أسهل لتحليل حجم كبير من البيانات بشكل مرئي. عادة ، يتم استخدامها لأغراض إعداد التقارير.
- توجد أنواع مختلفة من الرسومات البيانية، وتختلف في طريقة تمثيل البيانات مثل الرسم البياني الشريطي Bar Chart، الرسم البياني الدائري Pie Chart، الرسم البياني الخطي Line Chart، الرسم البياني المبعثر Scatter Chart، إلخ.

3

فهم Chart API

- تتكون واجهة برمجة تطبيقات الرسم البياني Chart API من عدد من الفئات المحددة مسبقًا في حزمة javafx.scene.chart ، يوضح الشكل التالي مخططًا لفئة الفئات التي تمثل أنواعًا مختلفة من الرسومات البيانية.





A class diagram for the classes representing charts in JavaFX

4

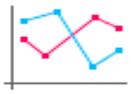
أصناف الرسومات البيانية Charts

- تحتوي فئة الرسم البياني على خصائص وطرق لجميع أنواع الرسم البياني. تصنف JavaFX الرسوم البيانية إلى صنفين:
  1. مخططات بلا محور no-axis.
  2. مخططات بها محور س x-axis ومحور ص y-axis.
- توجد فئة الرسم البياني الدائري ضمن الفئة الأولى. لا تحتوي على محور، ويتم استخدامها للرسم الدائري.
- تقع فئة XYChart في الفئة الثانية. إنها فئة أساسية لجميع الرسوم البيانية التي لها محورين. فئاتها الفرعية ، على سبيل المثال ، الرسم البياني الخطي Line Chart، الرسم البياني الشريطي Bar Chart، إلخ.

5

أنواع فئات Classes الرسومات البيانية

- يقدم الجدول التالي وصفًا لمختلف الرسومات البيانية (الفئات) التي توفرها JavaFX:
- **الرسم البياني الدائري Pie Chart**: هو تمثيل للقيم كشرائح في شكل دائرة وكل شريحة بلون مختلف. تتم عنونة هذه الشرائح ويتم تمثيل القيم المقابلة لكل شريحة في الرسم البياني.
  - في JavaFX، يتم تمثيل الرسم البياني الدائري بفئة تسمى PieChart، تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.
- **الرسم البياني الخطي Line Chart**: هو رسم بياني خطي يعرض المعلومات كسلسلة من نقاط البيانات (علامات markers) متصلة بواسطة خط مستقيم. يُظهر الرسم البياني الخطي كيف تتغير البيانات بتردد زمني متساو.
  - في JavaFX، يتم تمثيل الرسم البياني الخطي بواسطة فئة تسمى LineChart، تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.



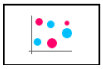
6

أنواع فئات Classes الرسومات البيانية

- **الرسم البياني المساحي Area Chart**: يستخدم لرسم رسم بياني قائم على المساحة. يرسم المنطقة الواقعة بين سلسلة النقاط المحددة والمحور. بشكل عام ، يتم استخدام هذا الرسم البياني لمقارنة كميتين.
  - في JavaFX، يتم تمثيله بواسطة فئة تسمىAreaChart،
  - تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.

- **الرسم البياني الشريطي Bar Chart**: يستخدم لتمثيل البيانات المجمعة باستخدام أشرطة مستطيلة، طول هذه الأشرطة يمثل القيم. يمكن رسم الأشرطة في الرسم البياني الشريطي عموديًا أو أفقيًا.
  - في JavaFX، يتم تمثيله بواسطة فئة تسمى BarChart،
  - تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.

- **الرسم البياني الفقاعي Bubble Chart**: يستخدم لرسم بيانات ثلاثية الأبعاد. يتم تمثيل البعد الثالث بحجم (نصف قطر) الفقاعة.
  - في JavaFX، يتم تمثيله بواسطة فئة تسمى BubbleChart، تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.



7

أنواع فئات Classes الرسومات البيانية

- **الرسم البياني المبعثر Scatter Chart**: هو نوع من الرسم البياني الذي يستخدم قيم من متغيرين يتم رسمهما في مستوى ديكارتي. يستخدم عادة لاكتشاف العلاقة بين متغيرين.
  - في JavaFX، يتم تمثيله بواسطة فئة تسمىScatterChart،
  - تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.

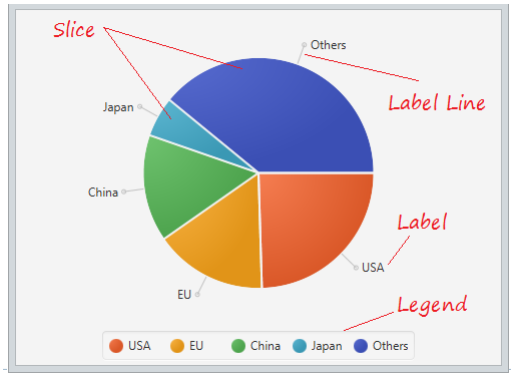
- **الرسم البياني المكدس Stacked Area Chart**: يتم تمثيل بواسطة فئة تسمى StackedAreaChart،
  - تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.
- **الرسم البياني الشريطي المكدس Stacked Bar Chart**: يتم تمثيله بواسطة فئة تسمى StackedBarChart،
  - تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.

8

الخصائص العامة لأنواع الرسومات البيانية

- تحدد الأنواع المختلفة من الرسومات البيانية بياناتها بشكل مختلف. تحتوي فئة الرسم البياني Chart على **الخصائص التالية العامة لجميع أنواع الرسومات البيانية :**

1. title
2. titleSide
3. legend
4. legendSide
5. legendVisible
6. Animated



٩

الخصائص العامة لأنواع الرسومات البيانية

- Title تحدد هذه الخاصية عنوان الرسم البياني.
- titleSide تحدد موقع العنوان. بشكل افتراضي، يتم وضع العنوان فوق محتوى الرسم البياني. قيمته الافتراضية TOP ويقبل القيم RIGHT و BOTTOM و LEFT .
- Legend تبين وسيلة الإيضاح للرموز مع أوصافها. بشكل افتراضي ، يتم وضع وسيلة الإيضاح أسفل محتوى الرسم البياني.

١٠

الخصائص العامة لأنواع الرسومات البيانية

- legendSide تحدد موقع وسيلة الإيضاح ، وتأخذ القيم TOP و RIGHT و BOTTOM و LEFT .
- legendVisible تحدد ما إذا كانت وسيلة الإيضاح مرئية أم لا. بشكل افتراضي ، يكون مرئيًا visible.
- Animated تحدد ما إذا كان سيتم عرض التغيير في محتوى المخطط مع نوع من الحركات. افتراضيا تأخذ القيمة true .

11

تنسيق الرسم البياني Chart مع CSS

- يمكن تنسيق جميع أنواع الرسومات البيانية باستخدام CSS. اسم فئة نمط CSSالافتراضي للرسم البياني هو chart.
- يمكنك تحديد خصائص legendSide و legendVisible وtitleSide لجميع الرسوم البيانية في CSS كما هو موضح في الشكل التالي:

```
.chart {
        -fx-legend-side: top;
        -fx-legend-visible: true;
        -fx-title-side: bottom;
}
```

12

يتكون الرسم البياني من هيكلين فرعيين

- كل رسم بياني يحتوي على هيكلين فرعيين:
  1. عنوان الرسم البياني chart-title.
  2. محتوى الرسم البياني chart-content.
- يقوم النمط التالي بتعيين لون الخلفية لجميع الرسومات البيانية إلى اللون الأصفر وخط العنوان إلى Arial بحجم 16 بكسل غامق.

```
.chart-content {
        -fx-background-color: yellow;
}

.chart-title {
        -fx-font-family: "Aeial";
        -fx-font-size: 16px;
        -fx-font-weight: bold;
}
```

13

يتكون الرسم البياني من هيكلين فرعيين

- اسم فئة النمط الافتراضي لوسيلة الإيضاح للرسم البياني هو chart-legend.
- يضبط النمط التالي لون خلفية وسيلة الإيضاح إلى اللون الفاتح.

```
.chart-legend {
        -fx-background-color: lightgray;
}
```

14

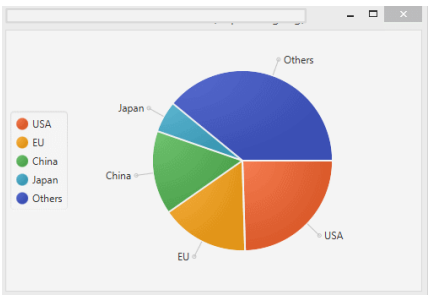
الرسم البياني الدائري Pie Chart

- الرسم البياني الدائري هو تمثيل للقيم كشرائح slices من دائرة بألوان مختلفة. تتم عنونة هذه الشرائح ويتم تمثيل القيم المقابلة لكل شريحة في الرسم البياني.
- يتكون الرسم البياني الدائري من دائرة مقسمة إلى قطاعات sectors من زوايا مركزية مختلفة. تعرف القطاعات أيضًا باسم قطع الدائرة pie pieces أو شرائح الدائرة pie slices.
- يمثل كل قطاع في الدائرة كمية من نوع ما.

١٥

خصائص الرسم البياني الدائري Pie Chart

- في JavaFX، يتم تمثيل الرسم البياني الدائري بفئة تسمى PieChart، تنتمي هذه الفئة إلى الحزمة javafx.scene.chart.
- **تحتوي فئة الرسم البياني الدائري PieChart على خمس خصائص هي:**



١. data
٢. labelsVisible
٣. labelLineLength
٤. startAngle
٥. clockwise

١٦

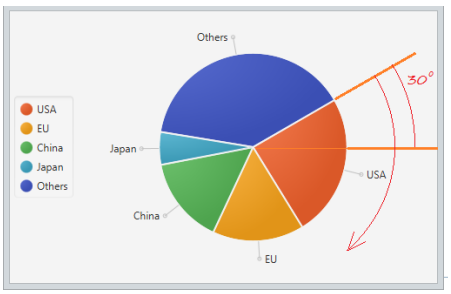
خصائص الرسم البياني الدائري Pie Chart

- data : يتم استخدام الكائن ObservableList ، والذي يقوم بالاحتفاظ ببيانات الرسم البياني الدائري. باستخدام
- ObservableList<PieChart.Data>
- labelsVisible: تحدد الخاصية ما إذا كانت تسميات الشرائح مرئية أم لا. يتم عرض تسميات الشرائح بالقرب من الشريحة ويتم وضعها خارج الشرائح. بشكل افتراضي ، تأخذ القيمة true.
- labelLineLength: تحدد الخاصية طول تلك الخطوط التي تربط العنوان والشريحة. قيمتها الافتراضية 20.0 بكسل.

17

خصائص الرسم البياني الدائري Pie Chart

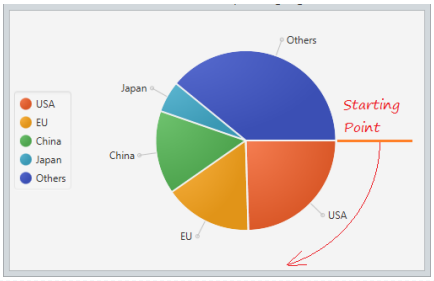
- startAngle تحدد الخاصية الزاوية التي تبدأ منها أول شريحة دائرية. بشكل افتراضي ، تكون درجة الصفر. التي تتوافق مع الساعة الثالثة. يتم قياس زاوية البداية الإيجابية عكس اتجاه عقارب الساعة. على سبيل المثال، ستبدأ زاوية البدء 90 درجة عند الساعة 12.



18

خصائص الرسم البياني الدائري Pie Chart

- clockwise: تحدد الخاصية اتجاه عقارب الساعة ما إذا كانت الشرائح ستوضع في اتجاه عقارب الساعة بدءًا من زاوية البداية. افتراضيا القيمة true. سيتم ترتيب شرائح البيانات في الرسم البياني الدائري في اتجاه عقارب الساعة بدءًا من زاوية بداية الرسم البياني الدائري.



19

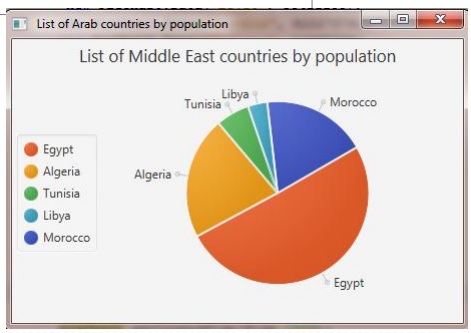
مثال1: الرسم البياني الدائري Pie Chart

```
public class PieChartList extends Application {
   @Override
   public void start(Stage primaryStage) throws Exception {
      //preparing ObservableList object
      ObservableList<PieChart.Data> piechart=FXCollections.observableArrayList(
       new PieChart.Data("Egypt", 98421139),
      new PieChart.Data("Algeria", 42227376),
      new PieChart.Data("Tunisia", 11565038),
      new PieChart.Data("Libya", 6678435),
      new PieChart.Data("Morocco", 36028537));
      // creating pie chart
      PieChart pieChart = new PieChart(piechart);
      //setting visible labels of pie chart
      pieChart.setLabelsVisible(true);
      //setting visible Legend of pie chart
      pieChart.setLegendVisible(true);
      //setting lenght of label line
       pieChart.setLabelLineLength(10);
       //setting direction to arrange the data
       pieChart.setClockwise(true);
       //setting startangle of pie chart
       pieChart.setStartAngle(30);
       //setting position of legend
      pieChart.setLegendSide(Side.LEFT);
```

20

مثال1: الرسم البياني الدائري Pie Chart

```
    //setting title of pie chart
    pieChart.setTitle("List of Middle East countries by population");
    //setting title of stage
    primaryStage.setTitle(" List of Arab countries by population ");
    StackPane root = new StackPane(pieChart);
    Scene scene = new Scene(root, 400, 200);
    primaryStage.setScene(scene);
    primaryStage.show();
}
```



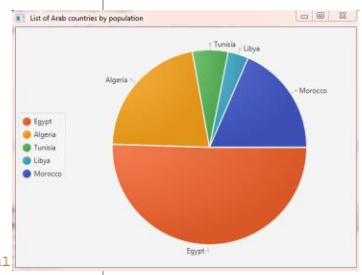
21

مثال2: الرسم البياني الدائري Pie Chart

```
public void start(Stage primaryStage) throws Exception {
    PieChart pieChart = new PieChart();

    PieChart.Data slice1 = new PieChart.Data("Egypt", 98421139);
    PieChart.Data slice2 = new PieChart.Data("Algeria", 42227376);
    PieChart.Data slice3 = new PieChart.Data("Tunisia", 11565038);
    PieChart.Data slice4 = new PieChart.Data("Libya", 6678435);
    PieChart.Data slice5 = new PieChart.Data("Morocco", 36028537);

    pieChart.getData().add(slice1);
    pieChart.getData().add(slice2);
    pieChart.getData().add(slice3);
    pieChart.getData().add(slice4);
    pieChart.getData().add(slice5);
    //pieChart.setLabelsVisible(false);
    //pieChart.setLegendVisible(false);
     pieChart.setLabelLineLength(10);
     //pieChart.setClockwise(false);
     //pieChart.setStartAngle(30);
    pieChart.setLegendSide(Side.LEFT);
    primaryStage.setTitle(" List of Arab countries by popul
    StackPane root = new StackPane(pieChart);
    Scene scene = new Scene(root, 400, 200);
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

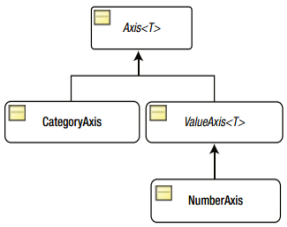


22

الرسم البياني XYChart

- الفئة XYChart هي فئة أساسية لجميع المخططات المرسومة على المحورين السيني والصادي XY.
- في JavaFX ، المحور هو فئة تمثل المحور X أوY . تحتوي على فئتين فرعيتين لتحديد كل نوع من أنواع المحاور XY ، وهما:

1. CategoryAxis
2. NumberAxis



- كما هو موضح في الرسم البياني التالي:

23

الفئة الفرعية من الرسم البياني XYChart

- CategoryAxis: يمكن تحديد (إنشاء) محور X أو Y حيث تمثل كل قيمة على طوله. يمكن تحديد محور الفئة عن طريق إنشاء مثيل لهذه الفئة كما هو موضح أدناه:

```
CategoryAxis xAxis = new CategoryAxis();
```

24

الفئة الفرعية من الرسم البياني XYChart

- NumberAxis: تحدد هذه الفئة (إنشاء) محور X أو Y حيث تمثل كل قيمة على طوله قيمة عددية.

```
//Defining the axis
NumberAxis yAxis = new NumberAxis();

//Setting labelto the axis
yAxis.setLabel("name of the axis");
```

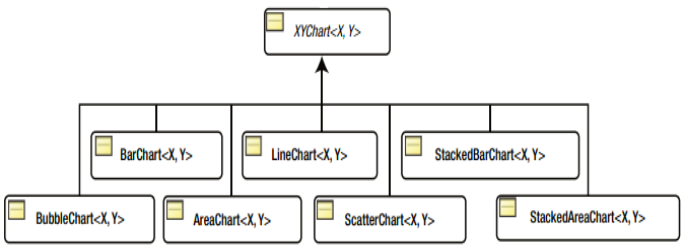
- مثال:

```
CategoryAxis xAxis = new CategoryAxis();
xAxis.setLabel("Country");
NumberAxis yAxis = new NumberAxis();
yAxis.setLabel("Population (in millions)");
```

25

أنواع الرسم البياني XYChart

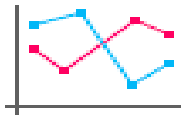
- يتضمن الرسم البياني XYChart التي يتم تمثيلها على مستوى س ص XY أنواع الرسومات البيانية التالية:
- الرسم البياني المساحي AreaChart، الرسم البياني الشريطي BarChart، الرسم البياني الفقاعي BubbleChart، الرسم البياني الخطي LineChart، الرسم البياني المبعثر ScatterChart، إلخ.



26

الرسم البياني الخطي LineChart

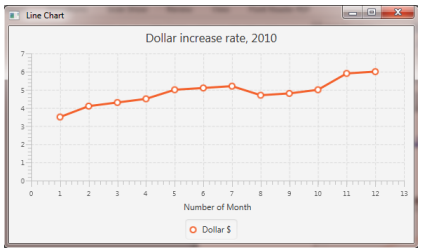
- يعرض الرسم البياني الخطي المعلومات على شكل سلسلة من نقاط البيانات (علامات markers) متصلة بواسطة خط مستقيم. يوضح الرسم البياني الخطي كيف تتغير البيانات عند تكرار الوقت نفسه.
- يتم إنشاء الرسم البياني الخطي عن طريق الفئة LineChart من الحزمة javafx.scene.chart.



27

مثال2: الرسم البياني الخطي LineChart

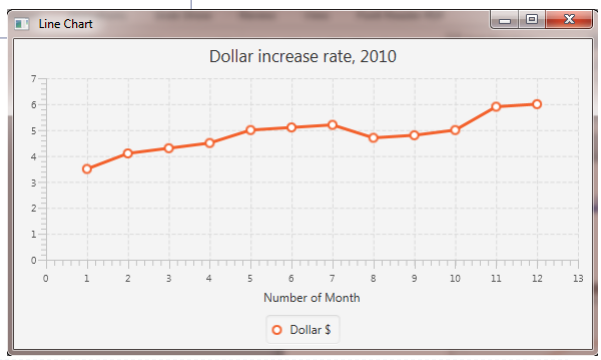
```
public void start(Stage stage) {
    stage.setTitle(" Line Chart ");
    //defining the axes
    final NumberAxis xAxis = new NumberAxis();
    final NumberAxis yAxis = new NumberAxis();
    xAxis.setLabel("Number of Month");
    //creating the chart
    final LineChart<Number,Number> lineChart =
    new LineChart<Number,Number>(xAxis,yAxis);
    lineChart.setTitle(" Dollar increase rate, 2010 ");
    //defining a series
    XYChart.Series series = new XYChart.Series();
    series.setName(" Dollar $ ");
    //populating the series with data
    series.getData().add(new XYChart.Data(1, 3.5));
    series.getData().add(new XYChart.Data(2, 4.1));
    series.getData().add(new XYChart.Data(3, 4.3));
    series.getData().add(new XYChart.Data(4, 4.5));
    series.getData().add(new XYChart.Data(5, 5.0));
    series.getData().add(new XYChart.Data(6, 5.1));
    series.getData().add(new XYChart.Data(7, 5.2));
    series.getData().add(new XYChart.Data(8, 4.7));
    series.getData().add(new XYChart.Data(9, 4.8));
    series.getData().add(new XYChart.Data(10, 5.0));
    series.getData().add(new XYChart.Data(11, 5.9));
    series.getData().add(new XYChart.Data(12, 6.0));
    Scene scene = new Scene(lineChart,800,600);
```



28

مثال2: الرسم البياني الخطي LineChart

```
        lineChart.getData().add(series);
        stage.setScene(scene);
        stage.show();
    }
    public static void main(String[] args) {
    launch(args);
    }
```



29

مثال: الرسم البياني الخطي LineChart

```
 * @author marwa
 */
public class LineChartExample extends Application {

    @Override
    public void start(Stage primaryStage) {

        //Defining the x axis
      NumberAxis xAxis = new NumberAxis(1960, 2020, 10);
      xAxis.setLabel("Years");
      //Defining the y axis
      NumberAxis yAxis = new NumberAxis(0, 350, 50);
      yAxis.setLabel("No.of schools");
      //Creating the line chart
      LineChart linechart = new LineChart(xAxis, yAxis);
      //Prepare XYChart.Series objects by setting data
      XYChart.Series series = new XYChart.Series();
      series.setName("No of schools in an year");

      series.getData().add(new XYChart.Data(1970, 15));
      series.getData().add(new XYChart.Data(1980, 30));
      series.getData().add(new XYChart.Data(1990, 60));
      series.getData().add(new XYChart.Data(2000, 120));
      series.getData().add(new XYChart.Data(2013, 240));
      series.getData().add(new XYChart.Data(2014, 300));
```

30

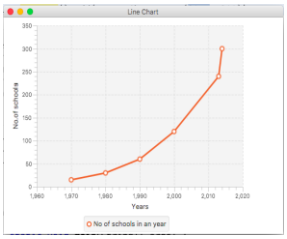
مثال: الرسم البياني الخطي LineChart

```
        //Setting the data to Line chart
        linechart.getData().add(series);
        //Creating a Group object
        Group root = new Group(linechart);
        //Creating a scene object
        Scene scene = new Scene(root, 600, 400);


        //Setting title to the Stage
        primaryStage.setTitle("Line Chart");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```



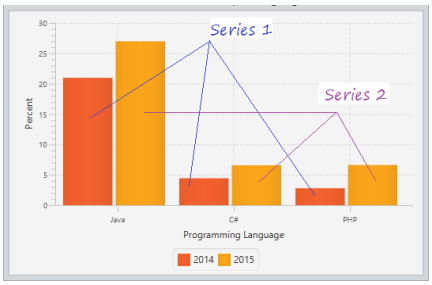
31

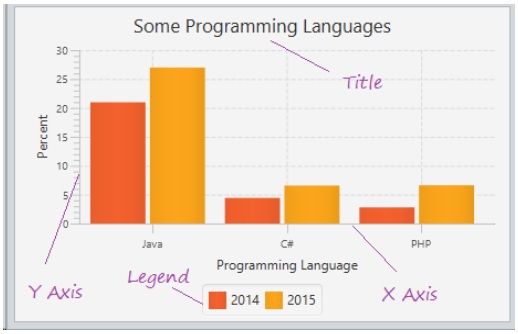
الرسم البياني الشريطي Bar Chart

- يعرض الرسم البياني الشريطي عناصر البيانات كأشرطة مستطيلة أفقية أو عمودية. أطوال الأشرطة تتناسب مع قيمة عناصر البيانات.
- في الرسم البياني الشريطي ، يجب أن يكون أحد المحورين هو CategoryAxis والآخر NumberAxis.
- يتم رسم الأشرطة عموديًا أو أفقيًا، اعتمادًا على ما إذا كان CategoryAxis هو المحور x أو المحور y .

32

Bar Chart الرسم البياني الشريطي



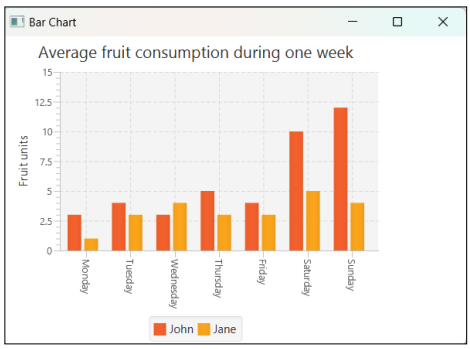


33

Bar Chart مثال: الرسم البياني الشريطي

```
public class ChartsBar extends Application {
    @Override
    public void start(Stage stage) {
        //Defining the X axis
        CategoryAxis xAxis = new CategoryAxis();
        //defining the y Axis
        NumberAxis yAxis = new NumberAxis(0, 15, 2.5);
        yAxis.setLabel("Fruit units");
        //Creating the Area chart
        BarChart<String, Number> barChart = new BarChart(xAxis, yAxis);
        barChart.setTitle("Average fruit consumption during one week");
        //Prepare XYChart.Series objects by setting data
        XYChart.Series series1 = new XYChart.Series();
        series1.setName("John");
        series1.getData().add(new XYChart.Data("Monday", 3));
        series1.getData().add(new XYChart.Data("Tuesday", 4));
        series1.getData().add(new XYChart.Data("Wednesday", 3));
        series1.getData().add(new XYChart.Data("Thursday", 5));
        series1.getData().add(new XYChart.Data("Friday", 4));
        series1.getData().add(new XYChart.Data("Saturday", 10));
        series1.getData().add(new XYChart.Data("Sunday", 12));

        XYChart.Series series2 = new XYChart.Series();
        series2.setName("Jane");
        series2.getData().add(new XYChart.Data("Monday", 1));
        series2.getData().add(new XYChart.Data("Tuesday", 3));
        series2.getData().add(new XYChart.Data("Wednesday", 4));
        series2.getData().add(new XYChart.Data("Thursday", 3));
        series2.getData().add(new XYChart.Data("Friday", 3));
        series2.getData().add(new XYChart.Data("Saturday", 5));
        series2.getData().add(new XYChart.Data("Sunday", 4));
        //Setting the XYChart.Series objects to area chart
        barChart.getData().addAll(series1,series2);
        //Creating a Group object
        Group root = new Group(barChart);
        //Creating a scene object
        Scene scene = new Scene(root, 600, 400);
        //Setting title to the Stage
        stage.setTitle("Bar Chart");
        //Adding scene to the stage
        stage.setScene(scene);
        //Displaying the contents of the stage
        stage.show();
```



ملخص المحاضرة

- الرسم البياني هو تمثيل رسومي للبيانات data. توفر JAVAFX مجموعة من الفئات Classes للرسومات البيانية.
- تختلف هذه الفئات في طريقة تمثيل البيانات ومن أمثلتها الرسم البياني الشريطي Bar Chart، الرسم البياني الدائري Pie Chart، الرسم البياني الخطي Line Chart، الرسم البياني المبعثر Scatter Chart.

35

نهاية المحاضرة

36