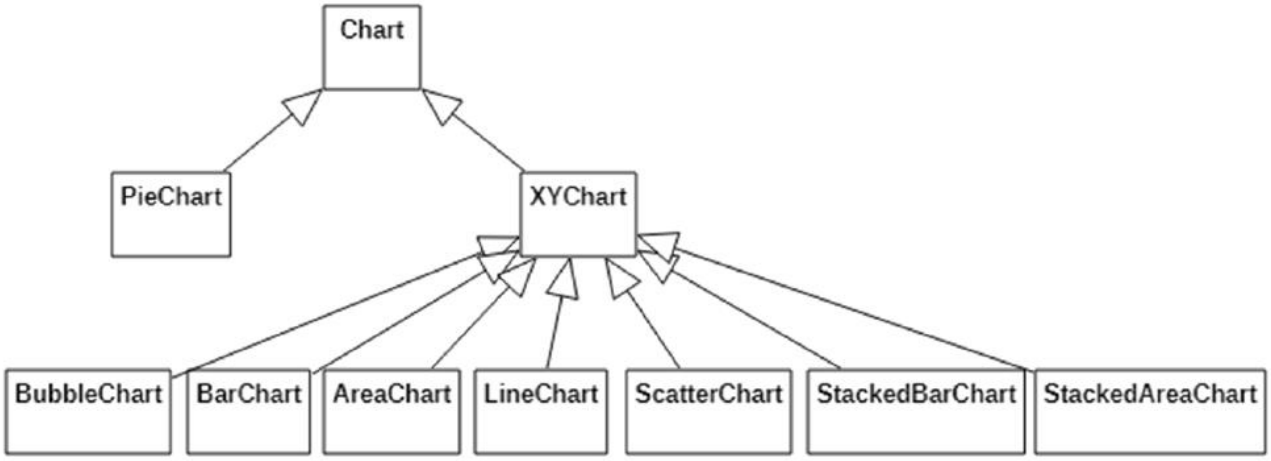
Gráficas

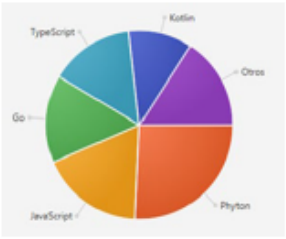
Las gráficas son representaciones visuales de datos que proporcionan una manera más fácil de analizar un gran volumen de datos. Existen diferentes tipos de gráficos que difieren en la forma en que representan los datos. JavaFX incluye gráficos que pueden integrarse en una aplicación Java escribiendo pocas líneas de código. JavaFX contiene una API de gráficos completa y extensible que proporciona soporte integrado para varios tipos de gráficos.

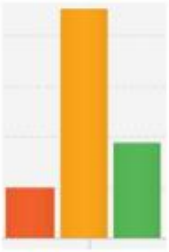
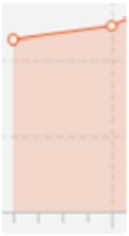
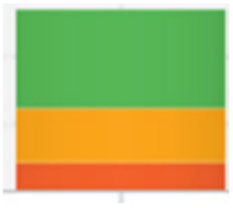
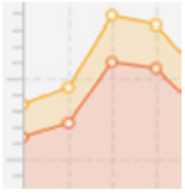
La API *Chart* de JavaFX consta de varias clases predefinidas en el paquete `javafx.scene.chart`. La figura muestra un diagrama de clase que representa diferentes tipos de gráficas definidos en JavaFX.



Jerarquía de clase gráficas para representar datos

En la tabla se describen brevemente las principales clases utilizadas en JavaFX para hacer diferentes gráficas.

Clase	Descripción	Gráfica
Chart	Clase base para todas las gráficas. Tiene tres partes: título, leyenda y contenido de la gráfica.	-
PieChart	Muestra un gráfico circular. El contenido del gráfico se rellena con sectores circulares basados en los datos establecidos.	
XYChart<X,Y>	Clase base para todas las gráficas de 2 ejes. Es responsable de dibujar los dos ejes y el contenido de la gráfica.	-
BubbleChart<X,Y>	Tipo de gráfica que traza burbujas para los puntos de datos en una serie.	

Clase	Descripción	Gráfica
BarChart<X,Y>	Una gráfica que traza barras que indican valores de datos para una categoría. Las barras pueden ser verticales u horizontales.	
AreaChart<X,Y>	Traza una área entre la línea que conecta los puntos de datos y la línea 0 en el eje Y.	
LineChart<X,Y>	Traza una línea que conecta los puntos de datos en una serie.	
ScatterChart<X,Y>	Tipo de gráfico que traza símbolos para los puntos de datos en una serie.	
StackedBarChat<X,Y>	Variación de *BarChart* que traza barras que indican valores de datos para una categoría.	
StackedAreaChat<X,Y>	Variación de *AreaChart* que muestra las tendencias de la contribución de cada valor, por ejemplo, el tiempo.	

Objetivos de aprendizaje

Al finalizar este ejercicio, serás capaz de:

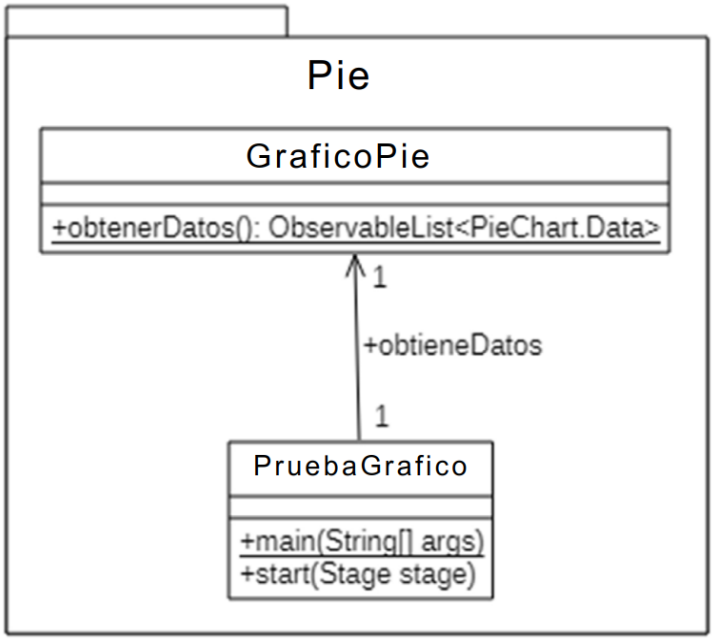
- Comprender la API *Chart* de JavaFX.
- Crear diferentes tipos de gráficas utilizando API *Chart* de JavaFX.

Enunciado: Gráfica de torta

Se desea desarrollar una gráfica de torta (*PieChart*) que muestre en formato gráfico los siguientes datos correspondientes a porcentajes de uso de lenguajes de programación:

- Phyton: 25.7 %
- JavaScript: 17.8 %
- Go: 15 %
- TypeScript: 14.6 %
- Kotlin: 11 %
- Otros: 15.8 %

Diagrama de clases



Explicación del diagrama de clases

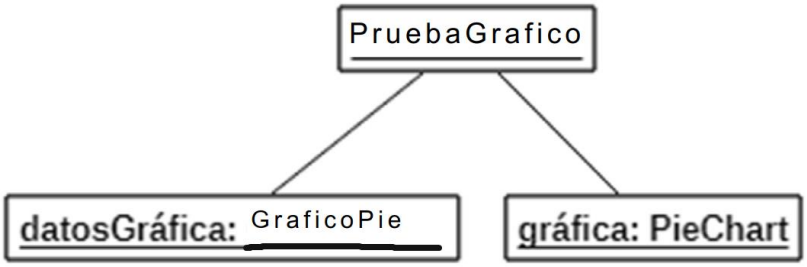
El diagrama de clases muestra un paquete denominado “GráficaPie”, que contiene dos clases denominadas “GráficoPie” y “PruebaGráfico”.

En primer lugar, la clase GráficaPie no tiene atributos pero cuenta con un método estático (indicado por su texto en subrayado) y público (indicado con el símbolo `+`) llamado obtenerDatos, el cual retorna un objeto *ObservableList* con los datos de la gráfica de quesito.

En segundo lugar, la clase PruebaGráfico tampoco tiene atributos. Sin embargo, posee dos métodos: `start` que inicia la aplicación gráfica y el método `main` (punto de entrada a la aplicación). El método `main` es un método estático, por lo cual se representa con su texto subrayado. Los dos métodos son públicos, lo cual se indica con el símbolo `+`.

La clase PruebaGráfico utiliza la clase GráficoPie, esto se expresa por medio de la relación de asociación entre las dos clases, denotada por una línea continua que conecta las dos clases y con el nombre de la relación que expresa que obtiene datos de esta clase. El sentido de la flecha que conecta las dos clases muestra la “navegabilidad” de la relación, que indica que la clase PruebaGráfico conoce e invoca la clase GráficoPie, pero la clase GráficoPie no conoce nada sobre la clase PruebaGráfico.

Diagrama de objetos



Ejecución del programación

Ejercicios propuestos

- Desarrollar un programa para generar una ventana gráfica con un diagrama de barras (*BarChart*) con los datos de los lenguajes de programación del ejercicio anterior.
- Desarrollar una línea de tendencias (*ScatterChat*) con los datos de venta de un producto de la tabla que se muestra a continuación.

Datos de venta de un producto

Mes	Unidades vendidas
Enero	205
Febrero	304
Marzo	348
Abril	386
Mayo	404
Junio	498

Instrucciones

Clase	Método	Descripción
ObservableList<E>	*ObservableList<PieChart.Data>*	Crea una lista que permite a los oyentes rastrear los cambios cuando ocurren en el *PieChart*.
	boolean add(Element e)	Añade el elemento especificado al final de la lista.
<E> FXCollections <E>	*observableArrayList()*	Crea una nueva lista observable vacía que está respaldada por un *arrayList*.
PieChart	*PieChart()*	Constructor de gráfica de torta.
	void setTitle(String valor)	Establece el título de la gráfica.
	void setLegendSide(Side valor)	Establece el lado donde se colocará la leyenda de la gráfica.
	void setData(ObservableList <PieChart.Data>)	Establece los valores de la gráfica.
StackPane	*StakedPane(Node valor)*	Constructor del objeto *StackedPane*.
Scene	*Scene (Node valor)*	Construye una escena para un nodo específico.
Stage	*void setScene(Scene valor)*	Especifica la escena utilizada en este escenario.
	void setTitle(String valor)	Establece el título del escenario.
	void show()	Muestra la ventana.