Instituto Tecnológico Nacional de México en Celaya

Ing. Sistemas Computacionales

Solis Guatemala Jorge Adrián

No. Control 18031123

Practica No 1.

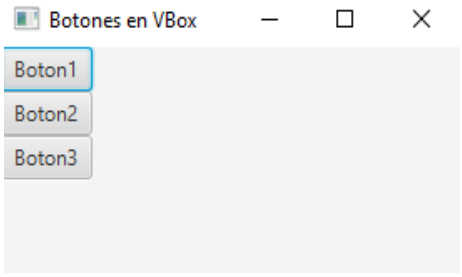
Objetivo.

Poder practicar algunos paneles y funciones de javaFx para poder usarlas en un proyecto más grande.

Cree un package con el nombre “views”, ahí pondré todas las clases donde probare cada panel de javaFx

```java
package com.example.views;

import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Botones extends Stage {

    private VBox vBox;
    private Button btn1,btn2,btn3;
    private Scene escena;

    public Botones(){
        CrearUI();
        this.setTitle("Botones en VBox");
        this.setScene(escena);
        this.show();
    }

    private void CrearUI() {
        btn1=new Button("Boton1");
        btn2=new Button("Boton2");
        btn3=new Button("Boton3");
        vBox=new VBox();
        vBox.getChildren().addAll(btn1,btn2,btn3);
        escena=new Scene(vBox,250,200);
    }
}
```

En esta clase se muestran 3 botones contenidos en un vBox y se espera mostrar 3 botones en forma vertical.

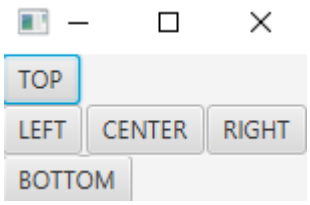


```java
package com.example.views;

import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Botones extends Stage {

    private VBox vBox;
    private Button btn1,btn2,btn3;
    private Scene escena;

    public Botones(){
        CrearUI();
        this.setTitle("Botones en VBox");
        this.setScene(escena);
        this.show();
    }

    private void CrearUI() {
        btn1=new Button( text: "Boton1");
        btn2=new Button( text: "Boton2");
        btn3=new Button( text: "Boton3");
        vBox=new VBox( spacing: 15);
        vBox.setPadding(new Insets( topRightBottomLeft: 15));
        vBox.getChildren().addAll(btn1,btn2,btn3);
        escena=new Scene(vBox);
    }
}
```

Aquí se busca separar los botones entre sí y además tratar de centrarlos.

```java
package com.example.views;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
public class Botones2 extends Stage {
    private BorderPane raiz;
    private Scene scene;
    public Botones2(){
        CrearUI();
        this.setTitle("Botones en 4 direcciones");
        this.setScene(scene);
        this.show();
    }
    private void CrearUI() {
        raiz = new BorderPane();
        raiz.setTop(new Button( text: "TOP"));
        raiz.setBottom(new Button( text: "BOTTOM"));
        raiz.setLeft(new Button( text: "LEFT"));
        raiz.setRight(new Button( text: "RIGHT"));
        raiz.setCenter(new Button( text: "CENTER"));

        scene = new Scene(raiz);
    }
}
```

Aquí se crea una nueva clase y en ella creamos un BorderPane que contendrá 4 botones y se colocaran en las 4 direcciones que existen.



```java
package com.example.views;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class FlowspAne extends Stage {

    private FlowPane flow;
    private Button btn1,btn2,btn3;
    private Scene escena;

    public FlowspAne(){
        CrearUI();
        this.setTitle("PruebaFlow");
        this.setScene(escena);
        this.show();
    }
    private void CrearUI() {
        btn1=new Button( text: "Boton1");
        btn2=new Button( text: "Boton2");
        btn3=new Button( text: "Boton3");
        flow=new FlowPane();
        flow.getChildren().addAll(btn1,btn2,btn3);
        escena=new Scene(flow);
    }
}
```

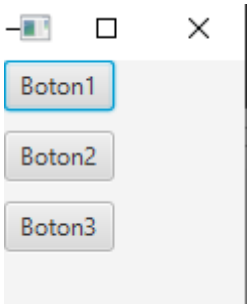
Aquí estamos probando el panel llamado FlowPane que funciona igual a un HBox y VBox, en este caso lo usare como si fuera un HBox

Para ver el resultado en vertical solo cambiamos esta línea

```java
flow=new FlowPane(Orientation.VERTICAL);
```

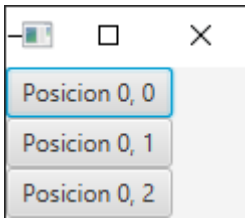
Para poder separar o espaciar se agrega la siguiente línea

```
flow.setVgap(10);
```



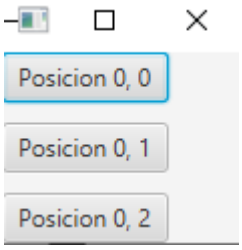
Ahora utilizare un gridpane que sirve para crear una matriz

```
package com.example.views;
import javafx.geometry.Orientation;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class FlowspAne extends Stage {

    private FlowPane flow;
    private Button btn1,btn2,btn3;
    private Scene escena;

    public FlowspAne(){
        CrearUI();
        this.setTitle("PruebaFlow");
        this.setScene(escena);
        this.show();
    }
    private void CrearUI() {
        btn1=new Button( text: "Boton1");
        btn2=new Button( text: "Boton2");
        btn3=new Button( text: "Boton3");
        flow=new FlowPane(Orientation.VERTICAL);
        flow.setVgap(10);
        flow.getChildren().addAll(btn1,btn2,btn3);
        escena=new Scene(flow);
    }
}
```



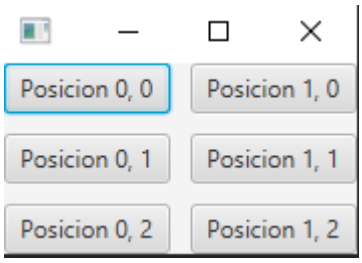
Al igual que el FlowPane, VBox y VBox se puede espaciar horizontal o vertical

```
grid.setVgap(10);
```



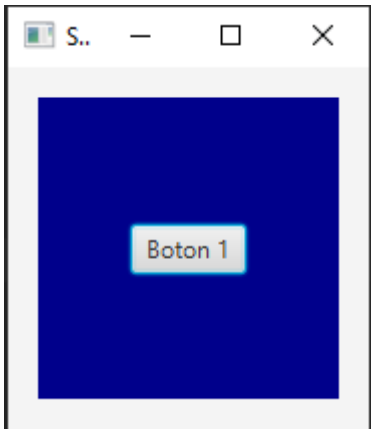
Recordando que es una matriz agregamos otros 3 botones y los colocamos en la posición 1 para demostrar las dos separaciones.

```
private void CrearUI() {
    btn1=new Button( text: "Posicion 0, 0");
    btn2=new Button( text: "Posicion 0, 1");
    btn3=new Button( text: "Posicion 0, 2");
    btn4=new Button( text: "Posicion 1, 0");
    btn5=new Button( text: "Posicion 1, 1");
    btn6=new Button( text: "Posicion 1, 2");
    grid=new GridPane();
    grid.setVgap(10);
    grid.setHgap(10);
    grid.add(btn1,  columnIndex: 0,  rowIndex: 0);
    grid.add(btn2,  columnIndex: 0,  rowIndex: 1);
    grid.add(btn3,  columnIndex: 0,  rowIndex: 2);
    grid.add(btn4,  columnIndex: 1,  rowIndex: 0);
    grid.add(btn5,  columnIndex: 1,  rowIndex: 1);
    grid.add(btn6,  columnIndex: 1,  rowIndex: 2);
    escena=new Scene(grid);
}
```



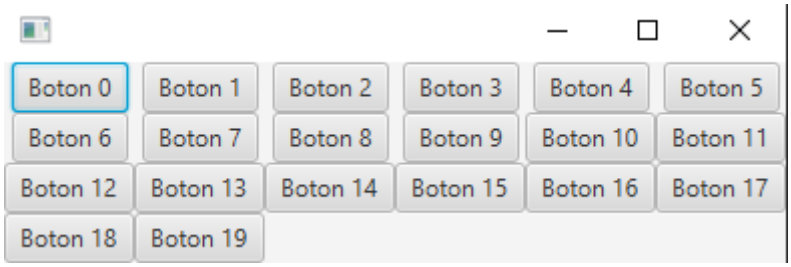
Ahora probare el StackPane que sirve para sobreponer cosas y se puede usar en cosas en específico como, por ejemplo:

```
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class LayStackPane extends Stage {
    private StackPane sp;
    private Scene escena;
    public LayStackPane(){
        CrearUI();
        this.setTitle("StackPane");
        this.setScene(escena);
        this.show();
    }
    private void CrearUI() {
        sp=new StackPane();
        sp.setPadding(new Insets( topRightBottomLeft: 15));
        sp.getChildren().addAll(new Rectangle( width: 150,  height: 150, Color.DARKBLUE),new Button( text: "Boton 1"));
        escena=new Scene(sp);
    }
}
```



Ahora usare TilePane que al igual que FlowPane y los demás se usa para poder añadir nodos, pero la diferencia radica en que en esta se pueden agregar columnas y filas.

```java
package com.example.views;

import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class LayStackPane extends Stage {
    private StackPane sp;
    private Scene escena;
    public LayStackPane(){
        CrearUI();
        this.setTitle("StackPane");
        this.setScene(escena);
        this.show();
    }
    private void CrearUI() {
        sp=new StackPane();
        sp.setPadding(new Insets( topRightBottomLeft: 15));
        sp.getChildren().addAll(new Rectangle( width: 150,  height: 150, Color.DARKBLUE),new Button( text: "Boton 1"));
        escena=new Scene(sp);
    }
}
```



Conclusiones

Los paneles para javaFx son muy dinámicos y útiles para poder crear buenas Interfaces de Usuario, además se pueden hacer dinámica ya que cada panel se usa para cosas en especifico.