Diseño y Programación Orientados a Objetos

1er. Semestre 2024

Certamen I ELO-329 Diseño y Programación Orientada a Objetos (1s2024)

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Segunda parte, **con apuntes**.

Tercera pregunta (35 pts.) Pauta

Desarrolle una simulación de una planta virtual, que puede ser regada, mediante una interfaz gráfica utilizando JavaFX. La planta tendrá varios estados y sus indicadores cambiarán con el paso del tiempo y la interacción del usuario. A continuación puede ver un ejemplo de la interfaz.

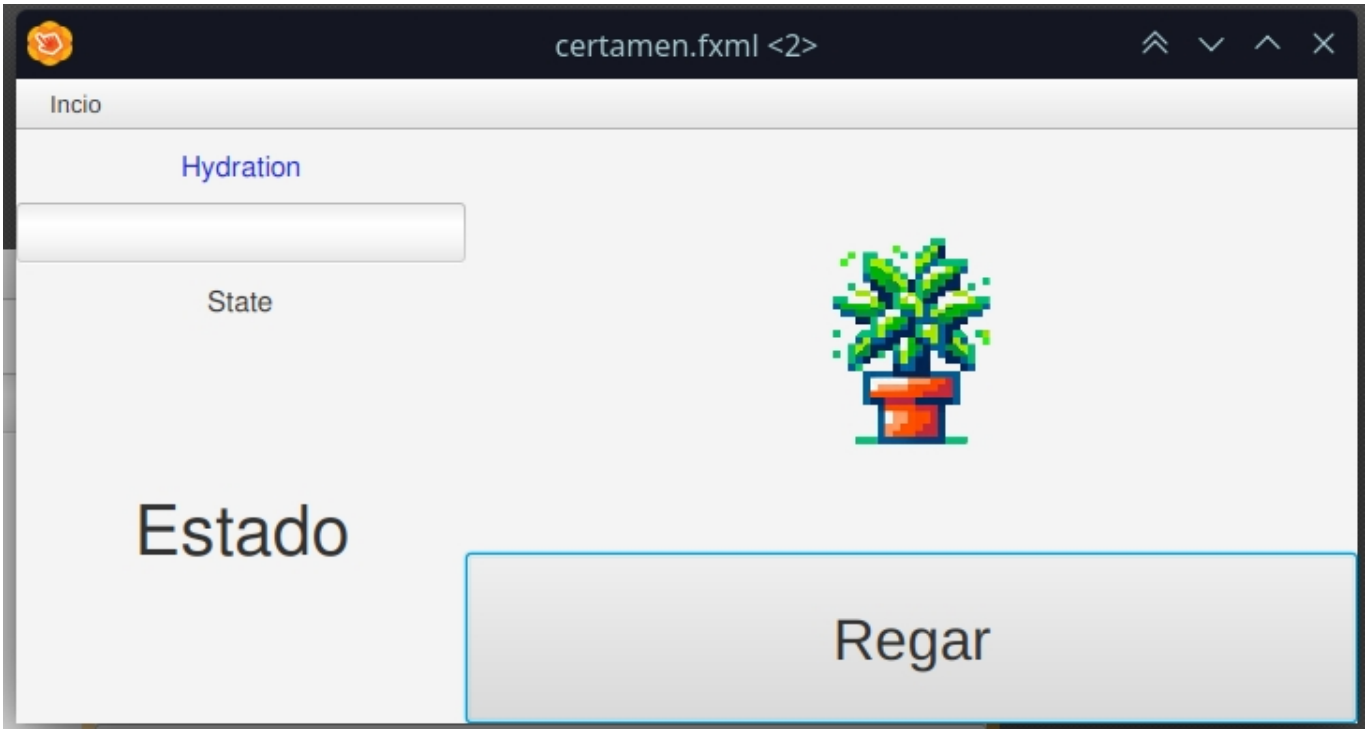


Figura 1: Ejemplo de visualización de la interfaz gráfica. La imagen de la planta fue generada por el modelo de AI DALL-E, de la empresa OpenAI.

La planta también puede estar en diferentes estados. Estos estados están definidos como un dato tipo `enum` en los códigos de ayuda. A continuación se presentan la lista de estados posibles según orden de prioridad (de menor a mayor), así como sus condiciones específicas:

Estados de la Planta (`enum PlantState`)

- `HEALTHY`: La planta tiene un valor para **hydration** > 30 .
- `THIRSTY`: La planta tiene un valor para **hydration** <= 30.
- `DEAD`: El valor de **hydration** de la planta es igual a 0.

También, considere el efecto del botón **Regar**, el cual incrementa el atributo **hydration** de `Plant` en 20 puntos por cada uso.

Puede utilizar los siguientes códigos de ayuda para su desarrollo:

- **Plant.java**
- **PlantState.java**

Pregunta 3.1

(10 pts.) Recree la interfaz gráfica para el simulador. Tome como base la siguiente imagen.

Para la imagen de la planta, utilice aquella disponible en el siguiente enlace.

```
Puntajes
En esta pregunta se puede utilizar tanto Scene Builder como JavaFX puro. Puntos
- 1pto por cada elemento gráfico (7 ptos en total)
    - 1pto barra de menú
    - 1pto etiqueta de hidratación
    - 1pto barra de progreso
    - 1pto etiqueta de estado
    - 1pto etiqueta de estado actual
    - 1pto imagen de planta
    - 1pto botón regar
- 3ptos por similitud de posicionamiento y despliegue con lo solicitado (no es
necesario similitud 100% para obtener todo el puntaje, pero si consistencia con
lo pedido)
```

Pregunta 3.2

(10 pts.) Cree todo el código correspondiente a la clase `Plant`, incluyendo los siguientes atributos:

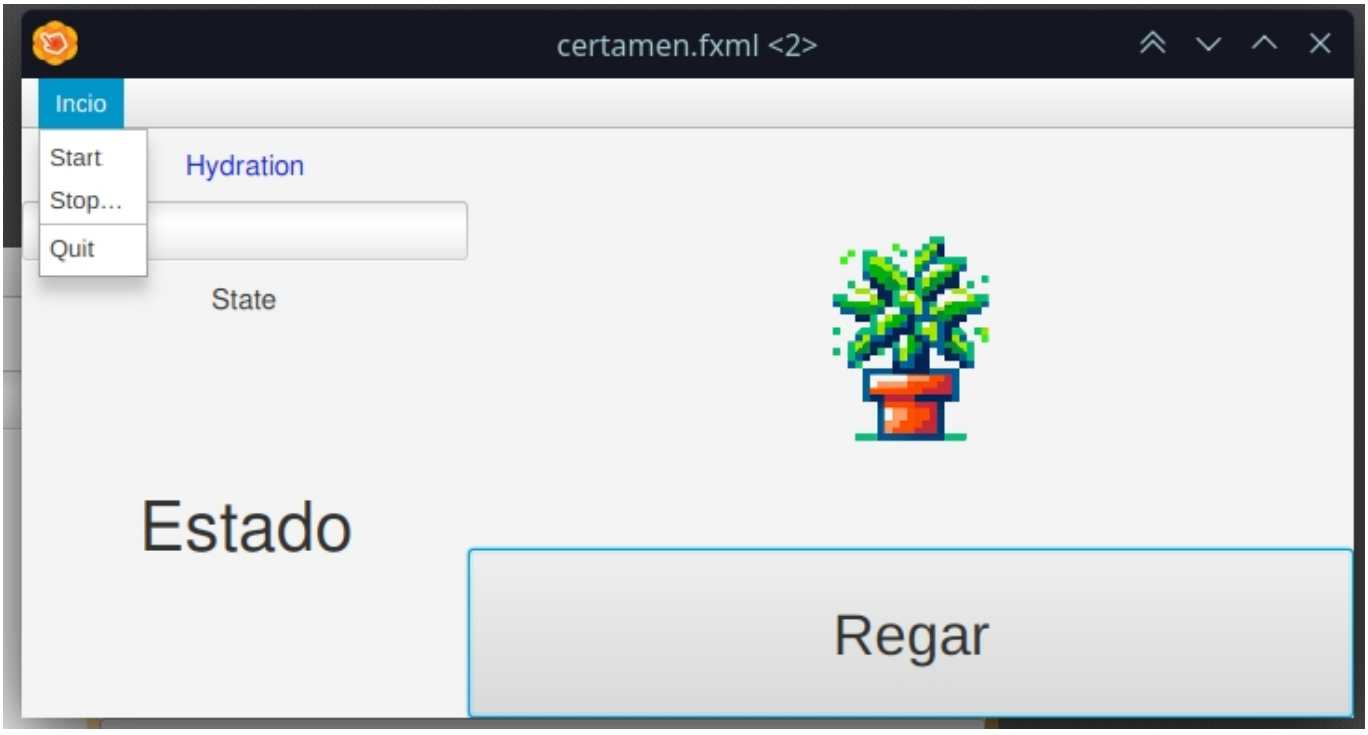
- **int hydration** (0 a 100), el cual representa su estado de hidratación. La planta debe iniciar con **hydration = 100**.
- **PlantState state**, dato `enum` que representa los diferentes estados de la planta. La planta debe iniciar con estado `HEALTHY`.

```
Puntajes
- 3pts Definición de clase y atributos privados solicitados e inicialización
correspondiente
- 3pts Método encargado del riego
- 3pts actualización de estado
- 1pto otros métodos necesarios para correcto funcionamiento
```

Pregunta 3.3

(10 pts.) Implemente un mecanismo que simule el paso del tiempo utilizando la clase Timeline vista en clases. Configure el elemento `Timeline` para que el atributo **hydration** de `Plant` disminuya en 5 puntos cada 0.5 segundos.

Además, agregue el menú de opciones de **Inicio**. Este menú permite **Iniciar** y **Detener** la simulación, así como **salir del programa**. Use la siguiente imagen como base para el menú.




```
Puntajes
- 1pto creación de una clase controlador
- 3ptos creación y configuración correcta de elemento `Timeline`
- 3ptos modificación al atributo de hidratación de la planta cada 0.5s
- 3ptos conexión de los botones **Iniciar**, **Detener** y **salir del
programa** con su respectivo funcionamiento.
```

Pregunta 3.4

(5 pts.) Conectar los diferentes elementos de la interfaz con sus respectivos elementos de la clase `Plant`. Esto incluye el funcionamiento del botón **Regar** sobre los atributos de la planta. Asegúrese que el indicador de hidratación y el estado de la planta se actualicen automáticamente en la interfaz gráfica.

```
Puntajes
- 5ptos Completar funcionamiento de clase controlador (o equivalente según
código) para conexión de elementos gráficos y la planta.
```

Ejemplo de código solución

p3.fxml (archivo interfaz gráfica)

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.ProgressBar?>
<?import javafx.scene.control.SeparatorMenuItem?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox prefHeight="344.0" prefWidth="716.0" xmlns="http://javafx.com/javafx/19"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="PlantController">
  <children>
    <MenuBar VBox.vgrow="NEVER">
      <menus>
        <Menu mnemonicParsing="false" text="Incio">
          <items>
            <MenuItem fx:id="start" mnemonicParsing="false" text="Start" />
            <MenuItem fx:id="stop" mnemonicParsing="false" text="Stop…" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem fx:id="quit" mnemonicParsing="false" text="Quit" />
          </items>
        </Menu>
      </menus>
    </MenuBar>
      <BorderPane prefHeight="343.0" prefWidth="676.0">
         <left>
            <VBox prefHeight="258.0" prefWidth="240.0"
BorderPane.alignment="CENTER">
               <children>
```

```
                  <Label alignment="CENTER" prefHeight="39.0" prefWidth="268.0"
text="Hydration" textAlignment="CENTER" textFill="#1825db">
                     <font>
                        <Font name="Khmer OS System" size="15.0" />
                     </font>
                  </Label>
                  <ProgressBar fx:id="health" prefHeight="33.0"
prefWidth="271.0" progress="0.0" />
                  <Label alignment="CENTER" prefHeight="39.0" prefWidth="268.0"
text="State" textAlignment="CENTER">
                     <font>
                        <Font name="Khmer OS System" size="15.0" />
                     </font>
                  </Label>
                  <Label fx:id="state" alignment="CENTER" prefHeight="204.0"
prefWidth="264.0" text="Estado" textAlignment="CENTER">
                     <font>
                        <Font name="Khmer OS System" size="37.0" />
                     </font>
                  </Label>
               </children>
            </VBox>
         </left>
         <center>
            <VBox prefHeight="200.0" prefWidth="100.0"
BorderPane.alignment="CENTER">
               <children>
                  <AnchorPane prefHeight="234.0" prefWidth="476.0">
                     <children>
                        <ImageView fitHeight="145.0" fitWidth="144.0"
layoutX="166.0" layoutY="41.0">
                           <image>
                              <Image url="@imagenes/plant.png" />
                           </image>
                        </ImageView>
                     </children>
                  </AnchorPane>
                  <HBox prefHeight="100.0" prefWidth="200.0">
                     <children>
                        <Button fx:id="waterButton" mnemonicParsing="false"
prefHeight="91.0" prefWidth="483.0" text="Regar">
                           <font>
                              <Font size="31.0" />
                           </font>
                        </Button>
                     </children>
                  </HBox>
               </children>
            </VBox>
         </center>
      </BorderPane>
  </children>
</VBox>
```

PlantSimulator.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class PlantSimulator extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            // Cargar la interfaz gráfica desde el archivo FXML
            Parent root = FXMLLoader.load(getClass().getResource("p3.fxml"));
            Scene scene = new Scene(root);
            primaryStage.setTitle("Simulador de Planta");
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Plant.java

```
// Posible implementación de clase Plant
public class Plant {
    private int hydration;
    private PlantState state;

    // Inicialización de atributos
    public Plant() {
        this.hydration = 100;
        this.state = PlantState.HEALTHY;
    }

    // Getter y Setter para hidratación
    public int getHydration() {
        return hydration;
    }

    public void setHydration(int hydration) {
        this.hydration = Math.min(hydration,100);
```

```
        if(this.hydration < 0 ) this.hydration = 0;
        updateState();
    }

    // Getter para estado
    public PlantState getState() {
        return state;
    }

    // Método encargado de actualizar estado
    private void updateState() {
        if (hydration > 30) {
            state = PlantState.HEALTHY;
        } else if (hydration > 0) {
            state = PlantState.THIRSTY;
        } else {
            state = PlantState.DEAD;
        }
    }

    // Método que se llama cuando riega la planta
    public void water() {
        setHydration(hydration + 20);
    }
}
```

PlantController.java

```
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.MenuItem;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;
import javafx.util.Duration;

// Ejemplo de implementación de clase controlador
public class PlantController {
    // Elementos visuales de la interfaz
    @FXML
    private Label state;
    @FXML
    private Label hydrationLabel;
    @FXML
    private Button waterButton;
    @FXML
    private MenuItem start;
    @FXML
    private MenuItem stop;
```

```
    @FXML
    private MenuItem quit;
    @FXML
    private ProgressBar health;

    private Plant plant;
    private Timeline timeline;

    @FXML
    public void initialize() {
        plant = new Plant();
        updateLabels();

        // Configurar el botón de regar
        waterButton.setOnAction(e -> waterPlant());

        // Configurar la simulación del tiempo
        timeline = new Timeline(new KeyFrame(Duration.seconds(0.5), e -> {
                // Cada vez que avanza el tiempo, se actualiza la planta y las
etiquetas de estado
                plant.setHydration(plant.getHydration() - 5);
                updateLabels();
        }));
        timeline.setCycleCount(Timeline.INDEFINITE);

        // Configurar los botones de inicio y detención
        start.setOnAction(e -> startSimulation());
        stop.setOnAction(e -> stopSimulation());
        quit.setOnAction(e -> System.exit(0));
    }

    private void waterPlant() {
        plant.water();
        updateLabels();
    }

    private void startSimulation() {
        timeline.play();
    }

    private void stopSimulation() {
        timeline.stop();
    }

    private void updateLabels() {
        state.setText(plant.getState().name());
        health.setProgress(plant.getHydration()/100.0);
    }
}
```

PlantState.java

```
// Enum para los estados de la planta
public enum PlantState {
    HEALTHY, THIRSTY, DEAD;
}
```