

Nombre:

Código:

Fecha: 28 de Septiembre de 2018

### Unidad 1: Construcción de la interfaz gráfica

Al finalizar esta unidad, el estudiante estará en capacidad de:

OE1.1. Utilizar una arquitectura de tres capas para el desarrollo de un programa de computador, repartiendo de manera adecuada las responsabilidades entre la interfaz de usuario, el control de la interfaz y el modelo. El estudiante deberá poder explicar la importancia de mantener separadas las clases de estos tres dominios.

OE1.2. Construir las clases que implementan una interfaz de usuario.

### Enunciado

La alcaldía de Santiago de Cali desea implementar un sistema de pago de parqueadero público que permita cobrar el parqueo de vehículos en zonas públicas. Los dispositivos donde las personas podrán pagar se dispondrán en la calle y cada persona tendrá únicamente que indicar que tipo de vehículo ha parqueado (Moto o Carro), el dispositivo le emitirá una tarjeta con la cual el conductor pagará por el tiempo utilizado al regresar por su vehículo.

Los dispositivos de hardware que permiten el cobro emitiendo la tarjeta aún no están listos, pero usted ha sido encargado para desarrollar un prototipo que permita simular a grandes rasgos el procedimiento simple que se deberá seguir para el registro la llegada del vehículo y el cobro posterior por el tiempo utilizado. La interfaz debe visualizarse igual que en el pantallazo que aparece en la Figura 1. El usuario luego debe elegir en el tipo de vehículo, y en ese momento el prototipo genera aleatoriamente una hora de la mañana entre 7:00 y 11:59. Esa hora inicial será visualizada en el campo Hora de Inicio, mientras los demás campos permanecerán en blanco. Luego, cuando el conductor regrese por su vehículo presiona el botón Calcular, y en ese momento se genera aleatoriamente una hora entre 12:00 y 21:59 que es mostrada en el campo Hora Final. Inmediatamente el programa calcula el costo de parqueo escribiendo el resultado en el campo Valor a Pagar. El estado de la ventana al final del proceso se puede observar en la Figura 2.

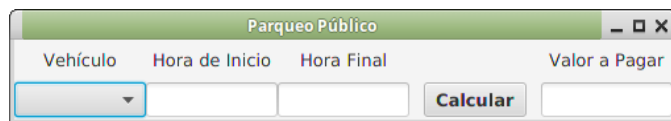


Figura 1. Ventana al iniciar el programa.

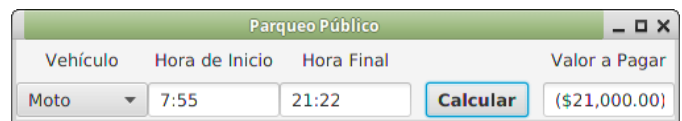


Figura 2. Ventana luego de terminado el proceso.

### Requerimientos Funcionales

El sistema debe estar en la capacidad de:

**R1. Registrar la hora de inicio de parqueo.** El sistema debe permitir elegir el tipo de vehículo entre Moto y Carro, luego de lo cual se debe generar aleatoriamente (por ser un prototipo) una hora entre 7:00 y 11:59 que será visualizada al usuario.

**R2. Registrar la hora final de parqueo.** El sistema debe permitir el registro de la finalización del parqueo. En este caso se genera aleatoriamente (por ser un prototipo) una hora entre 12:00 y 21:59 que será visualizada al usuario.

**R3. Calcular el valor a pagar.** El sistema debe permitir el cálculo del costo de parqueo teniendo en cuenta el tiempo transcurrido entre la hora de inicio y la hora final, y la tarifa de acuerdo con el tipo de vehículo. Este valor debe ser visualizada al usuario.

### Requerimiento NO Funcional

R1. El programa debe contar con una GUI desarrollada con Java FX.

### Ayuda sobre Java FX

Este par de líneas de código le ayudará a capturar el evento de cambio de valor en un control de opciones desplegable, es decir, cuando el usuario ha elegido una opción. Utilícelo donde lo considere apropiado.

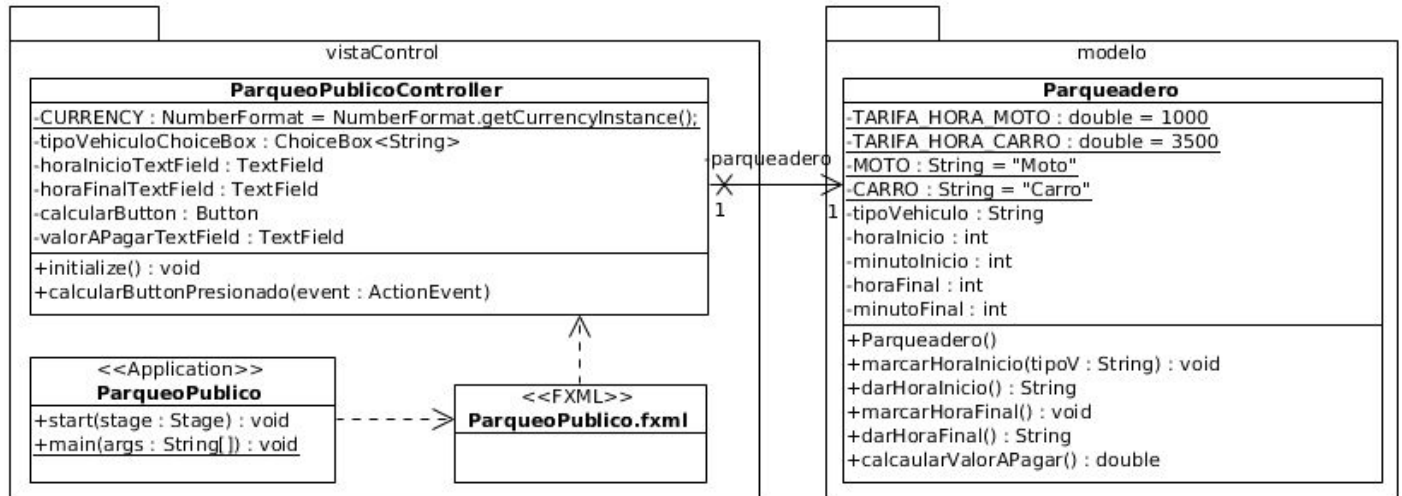
#### Línea 1.

```
combodesplegable.getSelectionModel().selectedIndexProperty().addListener(new ChangeListener<Number>() {
    @Override
    public void changed(ObservableValue<? extends Number> observableValue, Number number, Number number2) {
        System.out.println(combodesplegable.getItems().get((Integer) number2));
    }
});
```

#### Línea 2.

```
combodesplegable.setItems(FXCollections.observableArrayList("Elemento 1", "Elemento 2"));
```

## Diseño del Diagrama de Clases



## Entregable

Usted debe entregar la implementación del programa completamente funcional de acuerdo con las especificaciones del Enunciado, Requerimientos Funcionales y Diseño del Diagrama de Clases.

A continuación se les entrega la implementación del modelo.

```

1 package modelo;
2
3 public class Parqueadero {
4     public final static double TARIFA_HORA_MOTO = 1000;
5     public final static double TARIFA_HORA_CARRO = 3500;
6
7     public final static String MOTO = "Moto";
8     public final static String CARRO = "Carro";
9
10    private String tipoVehiculo;
11
12    private int horaInicio;
13    private int minutoInicio;
14    private int horaFinal;
15    private int minutoFinal;
16
17    public Parqueadero() {
18    }
19
20    public void marcarHoraInicio(String tipoV) {
21        tipoVehiculo = tipoV;
22        horaInicio = (int)(7+Math.random()*5);
23        minutoInicio = (int)(Math.random()*60);
24    }
25
26    public String darHoraInicio() {
27        return horaInicio+"."+minutoInicio;
28    }
29
30    public void marcarHoraFinal() {
31        horaInicio = (int)(12+Math.random()*10);
32        minutoInicio = (int)(Math.random()*60);
33    }
34
35    public String darHoraFinal() {
36        return horaInicio+"."+minutoInicio;
37    }
38
39    public double calcularValorAPagar() {
40        double tarifa;
41        if(tipoVehiculo.equals(MOTO)) {
42            tarifa = TARIFA_HORA_MOTO;
43        } else {
44            tarifa = TARIFA_HORA_CARRO;
45        }
46
47        double valorAPagar = (horaFinal-horaInicio)*tarifa;
48        if(minutoFinal>minutoInicio) {
49            valorAPagar += tarifa;
50        }
51        return valorAPagar;
52    }
53 }
  
```