



UNIVERSIDAD TECNOLÓGICA DE PANAMÁ



FACULTAD DE INGENIERÍA DE SISTEMAS
COMPUTACIONALES

LICENCIATURA EN CIBERSEGURIDAD

PROGRAMACION 1

LABORATORIO 2

PREPARADO POR

ADRIAN JIMENEZ 4-839-2413

JUSTIN HE 8-1045-2230

A CONSIDERACIÓN DE:

NAPOLEON IBARRA

GRUPO: 2S3111

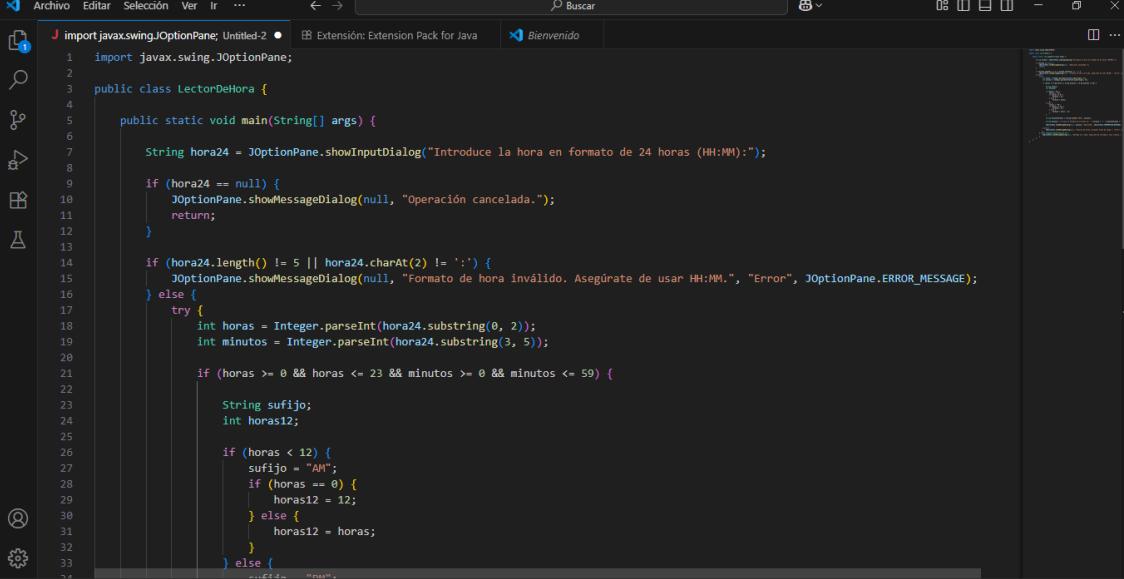
FECHA: 8-9-25

Programa 1: Leer la hora asignada e imprimirla

Objetivo

Leer la hora que un usuario le asigna al programa desarrollado en notación de 24 horas, para luego imprimirla en 12.

Código Desarrollado



```
import javax.swing.JOptionPane; Untitled-2
1 import javax.swing.JOptionPane;
2
3 public class LectorDeHora {
4
5     public static void main(String[] args) {
6
7         String hora24 = JOptionPane.showInputDialog("Introduce la hora en formato de 24 horas (HH:MM):");
8
9         if (hora24 == null) {
10             JOptionPane.showMessageDialog(null, "Operación cancelada.");
11             return;
12         }
13
14         if (hora24.length() != 5 || hora24.charAt(2) != ':') {
15             JOptionPane.showMessageDialog(null, "Formato de hora inválido. Asegúrate de usar HH:MM.", "Error", JOptionPane.ERROR_MESSAGE);
16         } else {
17             try {
18                 int horas = Integer.parseInt(hora24.substring(0, 2));
19                 int minutos = Integer.parseInt(hora24.substring(3, 5));
20
21                 if (horas >= 0 && horas <= 23 && minutos >= 0 && minutos <= 59) {
22
23                     String sufijo;
24                     int horas12;
25
26                     if (horas < 12) {
27                         sufijo = "AM";
28                         if (horas == 0) {
29                             horas12 = 12;
30                         } else {
31                             horas12 = horas;
32                         }
33                     } else {
```

Explicación del código

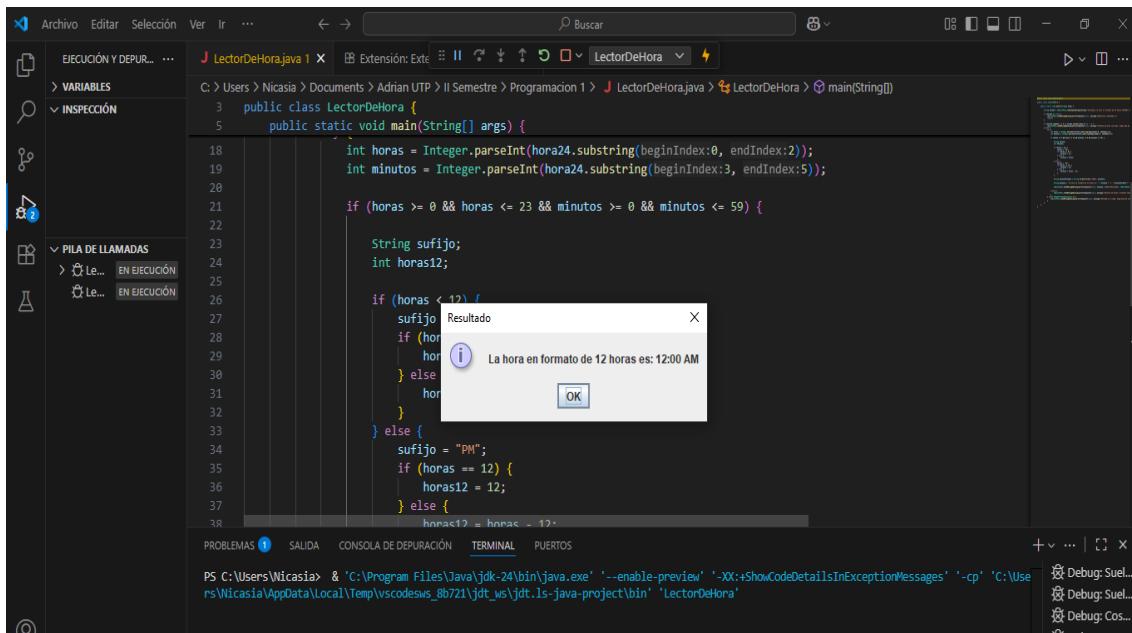
Primeramente se coloca el comando **import javax.swing.JOptionPane** porque es fundamental para se pueda abrir las ventanas a la hora de ejecutar el código.

Se utiliza el comando **JOptionPane.showInputDialog** para pedir la hora al usuario en una ventana. También **JOptionPane.showMessageDialog** para mostrar los mensajes de error en otra ventana si se escribe mal la hora.

También se ingresa el comando **Integer.parseInt** para convertir las horas y los minutos que el usuario escribió a números, para poder hacer los cálculos y **.substring** para cortar el texto de la hora para separar las horas de los minutos.

Y aparte también se utiliza el comando **.length** para verificar que la hora que se ingresó tenga 5 caracteres.

Ejecución del código



Programa 2: Costo diario de conducir y ahorro al compartir

Objetivo

Calcular el costo total diario de usar el automóvil para ir al trabajo y estimar el ahorro por persona si se comparte el viaje. Se consideran combustible, estacionamiento y peajes.

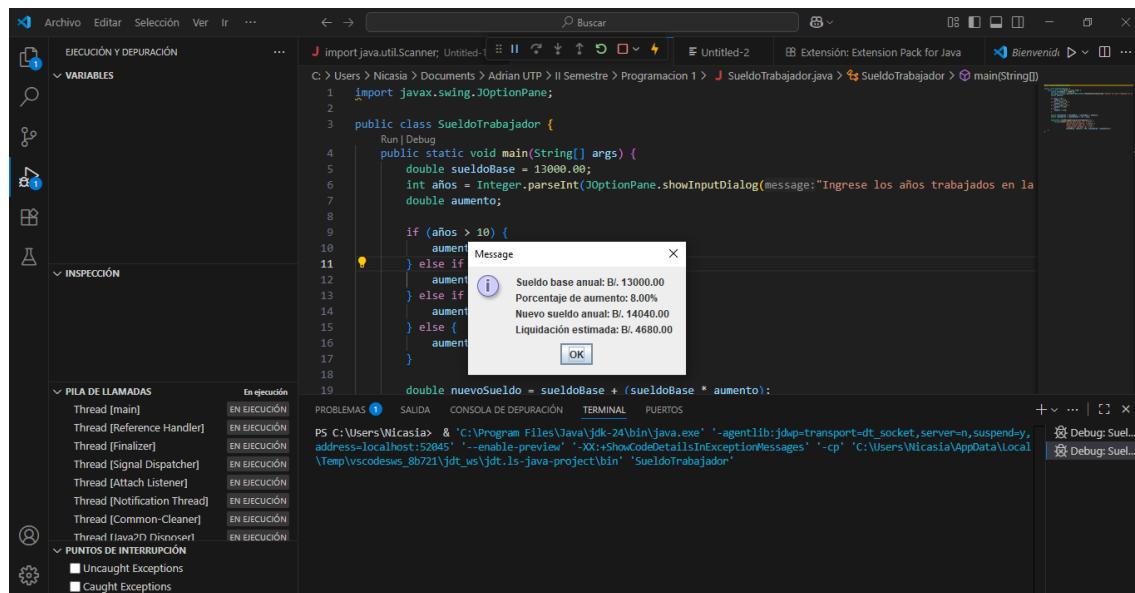
Código fuente: CostoAuto.java

```
1 import javax.swing.JOptionPane;
2
3 public class CostoAuto {
4     Run | Debug
5     public static void main(String[] args) {
6         double kmPorDia = Double.parseDouble(JOptionPane.showInputDialog(message:"Ingrese el total de kilómetros conducidos por día:"));
7         double costoLitro = Double.parseDouble(JOptionPane.showInputDialog(message:"Ingrese el costo por litro de combustible:"));
8         double kmPorLitro = Double.parseDouble(JOptionPane.showInputDialog(message:"Ingrese el promedio de kilómetros por litro:"));
9         double estacionamiento = Double.parseDouble(JOptionPane.showInputDialog(message:"Ingrese la cuota de estacionamiento por día:"));
10        double peaje = Double.parseDouble(JOptionPane.showInputDialog(message:"Ingrese el peaje por día:"));
11        int personas = Integer.parseInt(JOptionPane.showInputDialog(message:"Ingrese entre cuántas personas se comparte el auto (1 si no
12
13        double litrosConsumidos = kmPorDia / kmPorLitro;
14        double costoCombustible = litrosConsumidos * costoLitro;
15        double costoTotal = costoCombustible + estacionamiento + peaje;
16        double costoCompartido = costoTotal / personas;
17        double ahorro = costoTotal - costoCompartido;
18
19        JOptionPane.showMessageDialog(parentComponent:null,
20             String.format("Costo total diario sin compartir: B/. %.2f\n" +
21             "Costo diario por persona al compartir: B/. %.2f\n" +
22             "Ahorro estimado: B/. %.2f",
23             costoTotal, costoCompartido, ahorro));
24
25    }
}
```

Explicación del código

- Se piden los datos al usuario con JOptionPane.showInputDialog(...).
- Se convierten a double/int con Double.parseDouble(...) y Integer.parseInt(...).
- Se calculan litrosConsumidos, costoCombustible y costoTotal.
- Si hay más de 1 persona, se divide el costoTotal entre personas para obtener costoCompartido.
- Se calcula el ahorro como la diferencia entre manejar solo y el costo por persona compartiendo.
- Se muestran los resultados con 2 decimales usando String.format("%.2f").

Ejemplo de prueba rápida

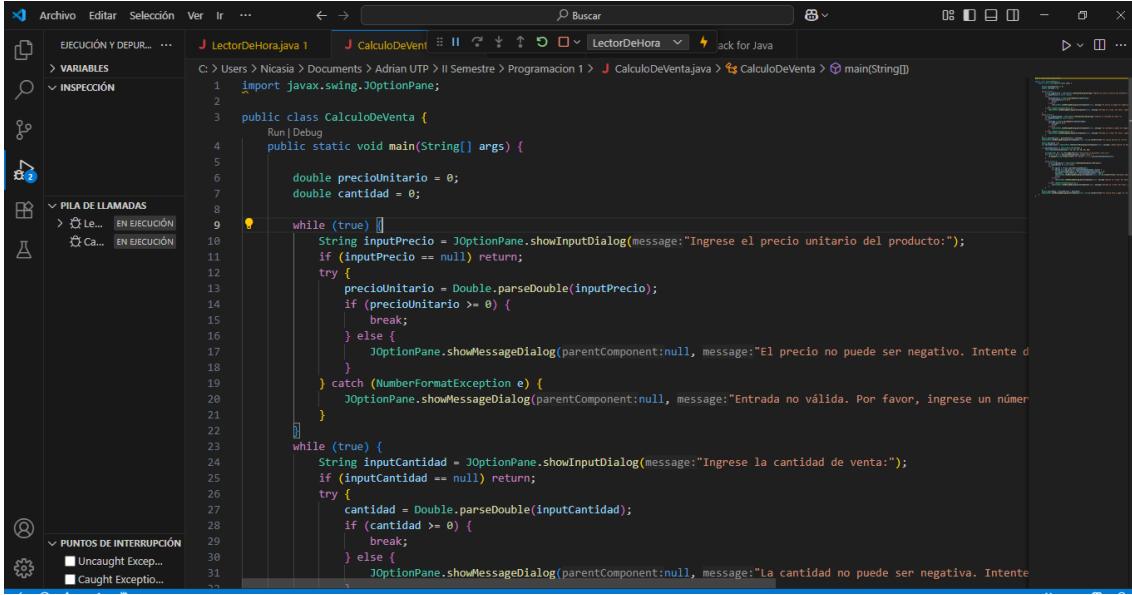


Programa 3: Calcular precio de un producto

Objetivo

Calcular el precio de un producto, y agregar un descuento solo si el vendedor lo requiera.

Código



The screenshot shows the Eclipse IDE interface with the following details:

- Menu Bar:** Archivo, Editar, Selección, Ver, Ir, ...
- Toolbar:** Buscar, Run, Stop, Refresh, Minimize, Maximize, Close.
- Left Sidebar:**
 - EJECUCIÓN Y DEPURACIÓN...
 - VARIABLES
 - INSPECCIÓN
 - PILA DE LLAMADAS
 - > LectorDeHora.java EN EJECUCIÓN
 - & CalculoDeVenta.java EN EJECUCIÓN
 - PUNTOS DE INTERRUPCIÓN
 - Uncaught Exception
 - Caught Exception
- Central Area:** Código fuente de la clase `CalculoDeVenta`. El código solicita al usuario el precio unitario y la cantidad de venta, verifica si son válidos y muestra mensajes de error si no lo son. Luego, pregunta si el usuario quiere un descuento.

```
import javax.swing.JOptionPane;
public class CalculoDeVenta {
    public static void main(String[] args) {
        double precioUnitario = 0;
        double cantidad = 0;
        while (true) {
            String inputPrecio = JOptionPane.showInputDialog("Ingrese el precio unitario del producto:");
            if (inputPrecio == null) return;
            try {
                precioUnitario = Double.parseDouble(inputPrecio);
                if (precioUnitario >= 0) {
                    break;
                } else {
                    JOptionPane.showMessageDialog(null, "El precio no puede ser negativo. Intente de nuevo");
                }
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Entrada no válida. Por favor, ingrese un número");
            }
        }
        while (true) {
            String inputCantidad = JOptionPane.showInputDialog("Ingrese la cantidad de venta:");
            if (inputCantidad == null) return;
            try {
                cantidad = Double.parseDouble(inputCantidad);
                if (cantidad >= 0) {
                    break;
                } else {
                    JOptionPane.showMessageDialog(null, "La cantidad no puede ser negativa. Intente de nuevo");
                }
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Entrada no válida. Por favor, ingrese un número");
            }
        }
        String respuesta = JOptionPane.showConfirmDialog(null, "¿Desea aplicar un descuento?", "Confirmación", JOptionPane.YES_NO_OPTION);
        if (respuesta == JOptionPane.YES_OPTION) {
            // Implementación del descuento
        }
    }
}
```

Explicación

Usamos `JOptionPane.showInputDialog` para pedir el precio y la cantidad en una ventana emergente. Este método siempre devuelve un **String o null** si el usuario cancela.

Para mostrar resultados o errores, usamos `JOptionPane.showMessageDialog`. Esto presenta la información en una ventana emergente.

Para preguntar si el usuario quiere un descuento, se utiliza `JOptionPane.showConfirmDialog`, que muestra una ventana con las opciones "Sí" y "No".

Se usa `StringBuilder` para construir el texto del menú de descuentos.

Se mantienen los bucles **while** y los bloques **try-catch** para asegurar que las entradas de precio, cantidad y la opción de descuento sean números válidos.

Ejecución del código

The screenshot shows two Java code editors side-by-side. Both editors contain the same code for calculating a product price and applying a discount.

```
lic static void main(String[] args) {  
    double precioUnitario = 0;  
    double cantidad = 0;  
  
    while (true) {  
        String inputPrecio = JOptionPane.showInputDialog(message:"Ingrese el precio unitario del producto");  
        if (inputPrecio == null) {  
            break;  
        } else {  
            try {  
                precioUnitario = Double.parseDouble(inputPrecio);  
                if (precioUnitario < 0) {  
                    JOptionPane.showMessageDialog(parentComponent,null, message:"El precio no puede ser negativo");  
                } else {  
                    JOptionPane.showMessageDialog(parentComponent,null, message:"El precio es válido");  
                }  
            } catch (NumberFormatException e) {  
                JOptionPane.showMessageDialog(parentComponent:null, message:"Entrada no válida. Por favor, ingrese un número");  
            }  
        }  
    }  
}  
  
SAUDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS  
casia> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages'  
data\LocalTemp\vscodews_8b721\jdt_ws\jdt.ls-java-project\bin'\CalculoDeVenta'
```

```
public class CalculoDeVenta {  
    Run|Debug  
    public static void main(String[] args) {  
  
        double precioUnitario = 0;  
        double cantidad = 0;  
  
        while (true) {  
            String inputPrecio = JOptionPane.showInputDialog(message:"Ingrese el precio unitario del producto");  
            if (inputPrecio == null) {  
                break;  
            } else {  
                try {  
                    precioUnitario = Double.parseDouble(inputPrecio);  
                    if (precioUnitario < 0) {  
                        JOptionPane.showMessageDialog(parentComponent,null, message:"El precio no puede ser negativo");  
                    } else {  
                        JOptionPane.showMessageDialog(parentComponent,null, message:"El precio es válido");  
                    }  
                } catch (NumberFormatException e) {  
                    JOptionPane.showMessageDialog(parentComponent:null, message:"Entrada no válida. Por favor, ingrese un número");  
                }  
            }  
        }  
    }  
}
```

The execution results show two dialog boxes. The first dialog, "Precio Parcial", displays "El precio es válido". The second dialog, "Descuento Aplicado", displays "Descuento aplicado: \$1.20".

The IDE interface includes toolbars, menus, and various debugging and monitoring tools like "EJECUCIÓN Y DEPURACIÓN", "VARIABLES", and "INSPECCIÓN". The code editor tabs show "LectorDeHora.java" and "CalculoDeVenta.java". The status bar at the bottom provides command-line information about the Java environment and project path.

Programa 4: Sueldo con aumento y liquidación

Objetivo

Calcular el nuevo sueldo anual de un trabajador que tiene un sueldo base de B/. 13,000 anuales, aplicando un aumento según la antigüedad y estimar una liquidación simple.

Criterios de aumento

- Más de 10 años: +15%
- Más de 5 y hasta 10 años: +10%
- Más de 3 y hasta 5 años: +8%
- 3 años o menos: +5%

Suposición para la liquidación (simplificada)

Se asume 1 sueldo mensual por cada año trabajado: liquidación = (nuevoSueldo / 12) × años. Esta fórmula es didáctica y simplificada para el ejercicio.

Código fuente: SueldoTrabajador.java

```
1 import javax.swing.JOptionPane;
2
3 public class SueldoTrabajador {
4     public static void main(String[] args) {
5         double sueldoBase = 13000.00;
6         int años = Integer.parseInt(JOptionPane.showInputDialog("Ingrese los años trabajados en la empresa:"));
7         double aumento;
8
9         if (años > 10) {
10             aumento = 0.15;
11         } else if (años > 5) {
12             aumento = 0.10;
13         } else if (años > 3) {
14             aumento = 0.08;
15         } else {
16             aumento = 0.05;
17         }
18
19         double nuevoSueldo = sueldoBase + (sueldoBase * aumento);
20         double liquidacion = (nuevoSueldo / 12) * años;
21
22         JOptionPane.showMessageDialog(null,
23             String.format("Sueldo base anual: B/. %.2f\n" +
24                 "Porcentaje de aumento: %.2f%%\n" +
25                 "Nuevo sueldo anual: B/. %.2f\n" +
26                 "Liquidación estimada: B/. %.2f",
27                 sueldoBase, aumento * 100, nuevoSueldo, liquidacion));
28     }
29 }
30 }
```

Explicación paso a paso

- Se solicita la antigüedad (años) vía JOptionPane.
- Según el valor de años, se asigna el porcentaje de aumento con if / else-if / else.
- Se calcula el nuevo sueldo anual: nuevoSueldo = sueldoBase + (sueldoBase × aumento).
- Se estima la liquidación: (nuevoSueldo / 12) × años.
- Se muestran los resultados formateados a 2 decimales.

Ejemplo de prueba rápida

