UNIDAD DE APRENDIZAJE
5

SEMANAS
10-11

Contadores y acumuladores

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al finalizar la unidad, el alumno, mediante el uso de variables locales y globales, diseña programas que involucren procesos de conteo y acumulación.

TEMARIO

1. Operadores de incremento y decremento
2. Operadores de asignación compleja
3. Contadores
4. Acumuladores

ACTIVIDADES

Los alumnos desarrollan programas usando contadores y acumuladores.

1. Operadores de incremento y decremento

Son operadores que permiten incrementar o decrementar en una unidad el valor de una variable numérica.

Operador	Uso	Equivalencia
++	a++;	a = a + 1;
--	a--;	a = a + 1;

Ejemplo 1

```
// Incrementa en uno el valor de x (Forma 1)
x = x + 1;

// Incrementa en uno el valor de x (Forma 2)
x++;

// Decrementa en 1 el valor de la variable z (Forma 1)
z = z - 1;

// Decrementa en 1 el valor de la variable z (Forma 2)
z--;
```

2. Operadores de asignación compleja

Son operadores que permiten asignar a una variable el valor de la variable mas, menos, por o entre el valor de otra variable.

Operador	Ejemplo	Equivalencia
+=	a += b;	a = a + b;
-=	a -= b;	a = a – b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;

Ejemplo 2

```
// Incrementa en 2 el valor de la variable z (Forma 1)
z = z + 2;

// Incrementa en 2 el valor de la variable z (Forma 2)
z += 2;

// Decrementa en 5 el valor de la variable m (Forma 1)
m = m – 5;

// Decrementa en 5 el valor de la variable m (Forma 2)
m -= 5;
```

3. Contadores

Un ***contador*** es una variable que se utiliza para contar el número de ocurrencias de un suceso o el número de veces que se cumple una determinada condición. El proceso de conteo se efectúa, generalmente, de uno en uno y consiste en incrementar en 1 a la variable conteo cada vez que ocurre el evento o suceso que se pretende contar. Antes de iniciar el proceso de conteo, el contador debe ser inicializado en 0.

Por ejemplo, se necesita un **contador** para determinar:

- La cantidad de veces que se hizo clic en un botón
- La cantidad de notas ingresadas
- La cantidad de notas desaprobatorias
- La cantidad de notas desaprobatorias
- La cantidad de ventas efectuadas
- Etc.

Una instrucción de conteo tiene la siguiente forma:

```
contador = contador + 1;
```

Que puede escribirse también como:

```
contador++;
```

Ejemplo 3

```
// Incrementa la cantidad de alumnos aprobados de una sección
cantidadAprobados++;

// Incrementa la cantidad de ventas efectuadas en un día
cantidadVentasDia++;
```

4. Acumuladores

Un ***acumulador*** es una variable que se utiliza para acumular o totalizar cantidades de una misma especie: sueldos, edades, pesos, etc. El proceso de acumulación consiste en incrementar la variable que sirve de acumulador en la cantidad que se pretende acumular. Antes de iniciar el proceso de acumulación, el acumulador debe ser inicializado en 0.

Por ejemplo, se necesita un **acumulador** para determinar:

- El sueldo total de los empleados de una empresa
- La suma total de las notas de un alumno
- La suma total de los pesos de un grupo de personas
- La suma total de las edades de un grupo de personas
- La cantidad total de unidades vendidas de un producto
- Etc.

Una instrucción de acumulación tiene la siguiente forma:

```
acumulador = acumulador + cantidad;
```

Qué puede escribirse también como:

```
acumulador +=  cantidad;
```

Ejemplo 4

```
// Incrementa el monto total vendido
montoTotalVendido += montoVenta;

// Incrementa el sueldo total de los empleados de una empresa
sueldoTotalEmpresa += sueldoEmpleado;

// Incrementa la cantidad total de unidades vendidas de un producto
cantidadUnidadesVendidas += cantidad;

// Incrementa la suma total de notas de un alumno
sumaNotas += nota;
```

5. Problemas resueltos

Problema 1

Una empresa desarrolladora de software ha puesto a la venta licencias de su programa de edición de video Video Edit 2.0 a los siguientes costos unitarios:

Licencia	Costo
Cobre	$ 510
Bronze	$ 1500
Silver	$ 3100
Gold	$ 4500

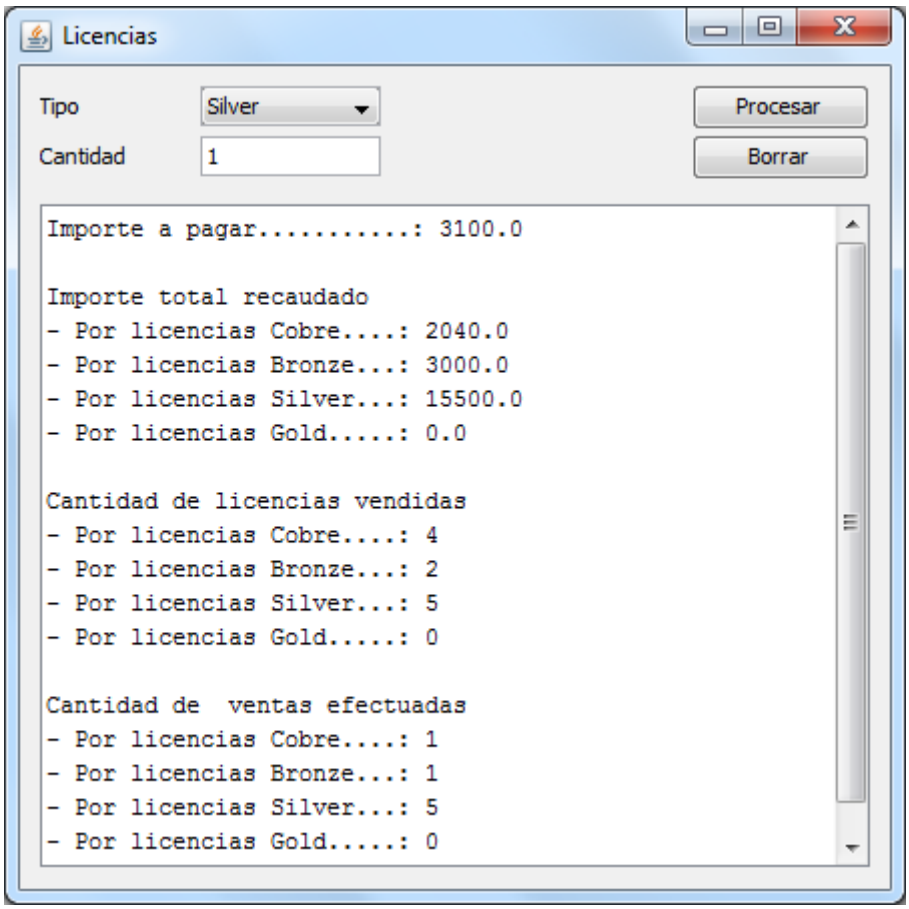
Diseñe un programa que permita ingresar, por cada venta, el tipo de licencia y la cantidad de licencias, y muestre luego de cada venta:

- El importe a pagar para la venta efectuada.
- El importe total recaudado de cada tipo de licencia.
- La cantidad de licencias vendidas de cada tipo de licencia.
- La cantidad de ventas efectuadas de cada tipo de licencia.

Solución

```
package cibertec;

import java.awt.EventQueue;
import java.awt.Font;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.UIManager;
```

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
```




```
public class Licencias extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblTipo;
    private JLabel lblCantidad;
    private JComboBox<String> cboTipo;
    private JTextField txtCantidad;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // Declaración de variables globales para el algoritmo
    double imptot0, imptot1, imptot2, imptot3;
    int canlic0, canlic1, canlic2, canlic3;
    int canven0, canven1, canven2, canven3;

    // Lanza la aplicación
    public static void main(String[] args) {
```

```
        try {
            UIManager.setLookAndFeel(
                "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
        catch (Throwable e) {
            e.printStackTrace();
        }
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Licencias frame = new Licencias();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Licencias() {
        setTitle("Licencias");
        setBounds(100, 100, 450, 449);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        lblTipo = new JLabel("Tipo");
        lblTipo.setBounds(10, 13, 80, 14);
        getContentPane().add(lblTipo);

        lblCantidad = new JLabel("Cantidad");
        lblCantidad.setBounds(10, 38, 80, 14);
        getContentPane().add(lblCantidad);

        cboTipo = new JComboBox<String>();
        cboTipo.setModel(new DefaultComboBoxModel<String>(new String[] {
                "Cobre", "Bronze", "Silver", "Gold" }));
        cboTipo.setBounds(90, 10, 90, 20);
        getContentPane().add(cboTipo);

        txtCantidad = new JTextField();
        txtCantidad.setBounds(90, 35, 90, 20);
        getContentPane().add(txtCantidad);
        txtCantidad.setColumns(10);

        btnProcesar = new JButton("Procesar");
        btnProcesar.addActionListener(this);
        btnProcesar.setBounds(335, 9, 89, 23);
        getContentPane().add(btnProcesar);

        btnBorrar = new JButton("Borrar");
        btnBorrar.addActionListener(this);
        btnBorrar.setBounds(335, 34, 89, 23);
        getContentPane().add(btnBorrar);

        scpScroll = new JScrollPane();
        scpScroll.setBounds(10, 69, 414, 331);
        getContentPane().add(scpScroll);
```

```
        txtS = new JTextArea();
        txtS.setFont(new Font("Monospaced", Font.PLAIN, 12));
        scpScroll.setViewportView(txtS);
    }

    // Direcciona eventos de tipo ActionEvent
    public void actionPerformed(ActionEvent arg0) {
        if (arg0.getSource() == btnProcesar) {
            actionPerformedBtnProcesar(arg0);
        }
        if (arg0.getSource() == btnBorrar) {
            actionPerformedBtnBorrar(arg0);
        }
    }

    // Procesa la pulsación del botón Procesar
    protected void actionPerformedBtnProcesar(ActionEvent arg0) {
        int tipo, cantidad;
        double imppag;
        tipo = getTipo();
        cantidad = getCantidad();
        imppag = calcularImportePagar(tipo, cantidad);
        efectuarIncrementos(tipo, cantidad, imppag);
        mostrarResultados(imppag);
    }

    // Procesa la pulsación del botón Borrar
    protected void actionPerformedBtnBorrar(ActionEvent arg0) {
        txtCantidad.setText("");
        txtS.setText("");
        txtCantidad.requestFocus();
    }

    // Lee y retorna el tipo de licencia
    int getTipo() {
        return cboTipo.getSelectedIndex();
    }

    // Lee y retorna la cantidad de licencias
    int getCantidad() {
        return Integer.parseInt(txtCantidad.getText());
    }

    // Calcula y retorna el importe a pagar
    double calcularImportePagar(int tip, int can) {
        switch (tip) {
            case 0:
                return 510 * can;
            case 1:
                return 1500 * can;
            case 2:
                return 3100 * can;
            default:
                return 4500 * can;
        }
    }

    // Efectúa los incrementos necesarios
```

```
    void efectuarIncrementos(int tip, int can, double ip) {
        switch (tip) {
            case 0:
                imptot0 += ip;
                canlic0 += can;
                canven0++;
                break;
            case 1:
                imptot1 += ip;
                canlic1 += can;
                canven1++;
                break;
            case 2:
                imptot2 += ip;
                canlic2 += can;
                canven2++;
                break;
            default:
                imptot3 += ip;
                canlic3 += can;
                canven3++;
        }
    }

    // Muestra el reporte solicitado
    void mostrarResultados(double ip) {
        txtS.setText("");
        imprimir("Importe a pagar...........: " + ip);
        imprimir("");
        imprimir("Importe total recaudado");
        imprimir("- Por licencias Cobre....: " + imptot0);
        imprimir("- Por licencias Bronze...: " + imptot1);
        imprimir("- Por licencias Silver...: " + imptot2);
        imprimir("- Por licencias Gold.....: " + imptot3);
        imprimir("");
        imprimir("Cantidad de licencias vendidas");
        imprimir("- Por licencias Cobre....: " + canlic0);
        imprimir("- Por licencias Bronze...: " + canlic1);
        imprimir("- Por licencias Silver...: " + canlic2);
        imprimir("- Por licencias Gold.....: " + canlic3);
        imprimir("");
        imprimir("Cantidad de  ventas efectuadas");
        imprimir("- Por licencias Cobre....: " + canven0);
        imprimir("- Por licencias Bronze...: " + canven1);
        imprimir("- Por licencias Silver...: " + canven2);
        imprimir("- Por licencias Gold.....: " + canven3);
    }

    // Imprime una cadena con un salto de línea al final
    void imprimir(String cad) {
        txtS.append(cad + "\n");
    }
}
```

Problema 2

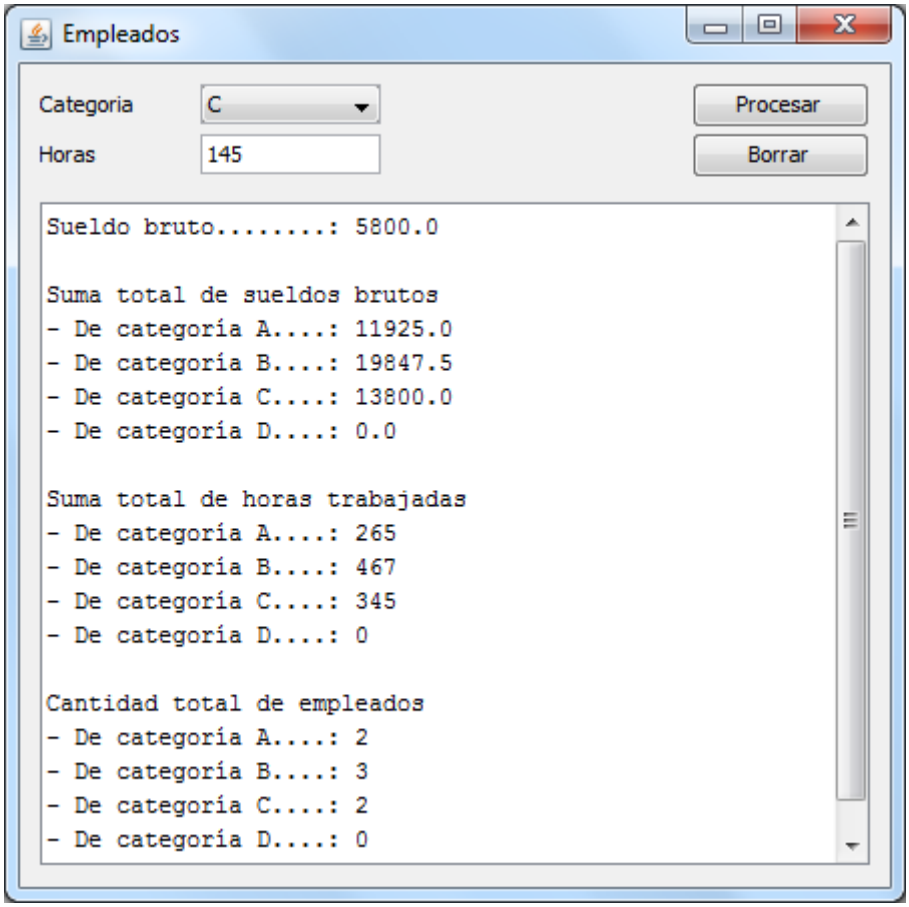
El sueldo bruto de los empleados de una empresa se calcula multiplicando las horas trabajadas por una tarifa horaria que depende de la categoría del empleado de acuerdo con la siguiente tabla:

Categoría	Tarifa
A	45.0
B	42.5
C	40.0
D	37.5

Diseñe un programa que permita ingresar, por cada empleado, la categoría y la cantidad de horas trabajadas, y muestre, luego de cada ingreso:

- El sueldo bruto del empleado
- La suma total de sueldos brutos de cada categoría
- La suma total de horas trabajadas de cada categoría
- La cantidad total de empleados de cada categoría

Solución




```
package cibertec;

import java.awt.EventQueue;
import java.awt.Font;
```

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.UIManager;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;

public class Empleados extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblCategoria;
    private JLabel lblHoras;
    private JComboBox<String> cboCategoria;
    private JTextField txtHoras;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // Declaración de variables globales para el algoritmo
    double sbrutot0, sbrutot1, sbrutot2, sbrutot3;
    int tothor0, tothor1, tothor2, tothor3;
    int canemp0, canemp1, canemp2, canemp3;

    // Lanza la aplicación
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(
                    "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
        catch (Throwable e) {
            e.printStackTrace();
        }
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Empleados frame = new Empleados();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Empleados() {
        setTitle("Empleados");
        setBounds(100, 100, 450, 449);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);
```

```
        lblCategoria = new JLabel("Categoria");
        lblCategoria.setBounds(10, 13, 80, 14);
        getContentPane().add(lblCategoria);

        lblHoras = new JLabel("Horas");
        lblHoras.setBounds(10, 38, 80, 14);
        getContentPane().add(lblHoras);

        cboCategoria = new JComboBox<String>();
        cboCategoria.setModel(new DefaultComboBoxModel<String>(
            new String[] { "A", "B", "C", "D" }));
        cboCategoria.setBounds(90, 10, 90, 20);
        getContentPane().add(cboCategoria);

        txtHoras = new JTextField();
        txtHoras.setBounds(90, 35, 90, 20);
        getContentPane().add(txtHoras);
        txtHoras.setColumns(10);

        btnProcesar = new JButton("Procesar");
        btnProcesar.addActionListener(this);
        btnProcesar.setBounds(335, 9, 89, 23);
        getContentPane().add(btnProcesar);

        btnBorrar = new JButton("Borrar");
        btnBorrar.addActionListener(this);
        btnBorrar.setBounds(335, 34, 89, 23);
        getContentPane().add(btnBorrar);

        scpScroll = new JScrollPane();
        scpScroll.setBounds(10, 69, 414, 331);
        getContentPane().add(scpScroll);

        txtS = new JTextArea();
        txtS.setFont(new Font("Monospaced", Font.PLAIN, 12));
        scpScroll.setViewportView(txtS);
    }

    // Direcciona eventos de tipo ActionEvent
    public void actionPerformed(ActionEvent arg0) {
        if (arg0.getSource() == btnProcesar) {
            actionPerformedBtnProcesar(arg0);
        }
        if (arg0.getSource() == btnBorrar) {
            actionPerformedBtnBorrar(arg0);
        }
    }

    // Procesa la pulsación del botón Procesar
    protected void actionPerformedBtnProcesar(ActionEvent arg0) {
        int categoria, horas;
        double suelbru;
        categoria = getCategoria();
        horas = getHoras();
        suelbru = calcularSueldoBruto(categoria, horas);
        efectuarIncrementos(categoria, horas, suelbru);
        mostrarResultados(suelbru);
    }
```

```
    // Procesa la pulsación del botón Borrar
    protected void actionPerformedBtnBorrar(ActionEvent arg0) {
        txtHoras.setText("");
        txtS.setText("");
        txtHoras.requestFocus();
    }

    // Lee y retorna la categoría
    int getCategoria() {
        return cboCategoria.getSelectedIndex();
    }

    // Lee y retorna las cantidad de horas trabajadas
    int getHoras() {
        return Integer.parseInt(txtHoras.getText());
    }

    // Calcula y retorna el sueldo bruto
    double calcularSueldoBruto(int cat, int hor) {
        switch (cat) {
            case 0:
                return 45.0 * hor;
            case 1:
                return 42.5 * hor;
            case 2:
                return 40.0 * hor;
            default:
                return 37.5 * hor;
        }
    }

    // Efectúa los incrementos necesarios
    void efectuarIncrementos(int cat, int hor, double sb) {
        switch (cat) {
            case 0:
                sbrutot0 += sb;
                tothor0 += hor;
                canemp0++;
                break;
            case 1:
                sbrutot1 += sb;
                tothor1 += hor;
                canemp1++;
                break;
            case 2:
                sbrutot2 += sb;
                tothor2 += hor;
                canemp2++;
                break;
            default:
                sbrutot3 += sb;
                tothor3 += hor;
                canemp3++;
        }
    }

    // Muestra el reporte solicitado
    void mostrarResultados(double sb) {
        txtS.setText("");
```

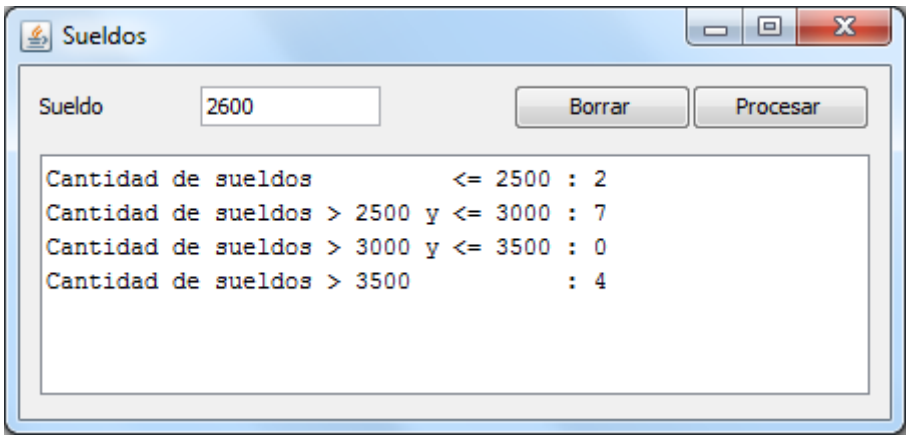
```
        imprimir("Sueldo bruto.....: " + sb);
        imprimir("");
        imprimir("Suma total de sueldos brutos");
        imprimir("- De categoría A....: " + sbrutot0);
        imprimir("- De categoría B....: " + sbrutot1);
        imprimir("- De categoría C....: " + sbrutot2);
        imprimir("- De categoría D....: " + sbrutot3);
        imprimir("");
        imprimir("Suma total de horas trabajadas");
        imprimir("- De categoría A....: " + tothor0);
        imprimir("- De categoría B....: " + tothor1);
        imprimir("- De categoría C....: " + tothor2);
        imprimir("- De categoría D....: " + tothor3);
        imprimir("");
        imprimir("Cantidad total de empleados");
        imprimir("- De categoría A....: " + canemp0);
        imprimir("- De categoría B....: " + canemp1);
        imprimir("- De categoría C....: " + canemp2);
        imprimir("- De categoría D....: " + canemp3);
    }

    // Imprime una cadena con un salto de línea al final
    void imprimir(String cad) {
        txtS.append(cad + "\n");
    }
}
```

Programa 3

Diseñe un programa que permita ingresar los sueldos de un conjunto de empleados. Luego de cada ingreso, muestre un reporte actualizado indicando:

- La cantidad de sueldos menores o iguales que 2500
- La cantidad de sueldos mayores de 2500 pero menores o iguales a 3000
- La cantidad de sueldos mayores de 3000 pero menores o iguales a 3500
- La cantidad de sueldos mayores de 3500

Solución




```
package cibertec;

import java.awt.EventQueue;
import java.awt.Font;
```

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.UIManager;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class Sueldos extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblSueldo;
    private JTextField txtSueldo;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // Declaración de variables globales para el algoritmo
    int conta1, conta2, conta3, conta4;

    // Lanza la aplicación
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(
                    "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
        catch (Throwable e) {
            e.printStackTrace();
        }
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Sueldos frame = new Sueldos();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Sueldos() {
        setTitle("Sueldos");
        setBounds(100, 100, 450, 214);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        lblSueldo = new JLabel("Dato1");
        lblSueldo.setBounds(10, 13, 80, 14);
        getContentPane().add(lblSueldo);

        txtSueldo = new JTextField();
        txtSueldo.setBounds(90, 10, 90, 20);
```

```
        getContentPane().add(txtSueldo);
        txtSueldo.setColumns(10);

        btnProcesar = new JButton("Procesar");
        btnProcesar.addActionListener(this);
        btnProcesar.setBounds(335, 9, 89, 23);
        getContentPane().add(btnProcesar);

        btnBorrar = new JButton("Borrar");
        btnBorrar.addActionListener(this);
        btnBorrar.setBounds(246, 9, 89, 23);
        getContentPane().add(btnBorrar);

        scpScroll = new JScrollPane();
        scpScroll.setBounds(10, 44, 414, 120);
        getContentPane().add(scpScroll);

        txtS = new JTextArea();
        txtS.setFont(new Font("Monospaced", Font.PLAIN, 12));
        scpScroll.setViewportView(txtS);
    }

    // Direcciona eventos de tipo ActionEvent
    public void actionPerformed(ActionEvent arg0) {
        if (arg0.getSource() == btnProcesar) {
            actionPerformedBtnProcesar(arg0);
        }
        if (arg0.getSource() == btnBorrar) {
            actionPerformedBtnBorrar(arg0);
        }
    }

    // Procesa la pulsación del botón Procesar
    protected void actionPerformedBtnProcesar(ActionEvent arg0) {
        double sueldo;
        sueldo = getSueldo();
        efectuarIncrementos(sueldo);
        mostrarResultados();
    }

    // Procesa la pulsación del botón Borrar
    protected void actionPerformedBtnBorrar(ActionEvent arg0) {
        txtSueldo.setText("");
        txtS.setText("");
        txtSueldo.requestFocus();
    }

    // Lee y retorna el sueldo
    double getSueldo() {
        return Double.parseDouble(txtSueldo.getText());
    }

    // Efectúa los incrementos necesarios
    void efectuarIncrementos(double suel) {
        if (suel <= 2500)        conta1++;
        else if (suel <= 3000)   conta2++;
        else if (suel <= 3500)   conta3++;
        else                     conta4++;
    }
```

```
    // Muestra los resultados solicitados
    void mostrarResultados() {
        txtS.setText("");
        imprimir("Cantidad de sueldos          <= 2500 : " + conta1);
        imprimir("Cantidad de sueldos > 2500 y <= 3000 : " + conta2);
        imprimir("Cantidad de sueldos > 3000 y <= 3500 : " + conta3);
        imprimir("Cantidad de sueldos > 3500           : " + conta4);
    }

    // Imprime una cadena con un salto de línea al final
    void imprimir(String cad) {
        txtS.append(cad + "\n");
    }
}
```

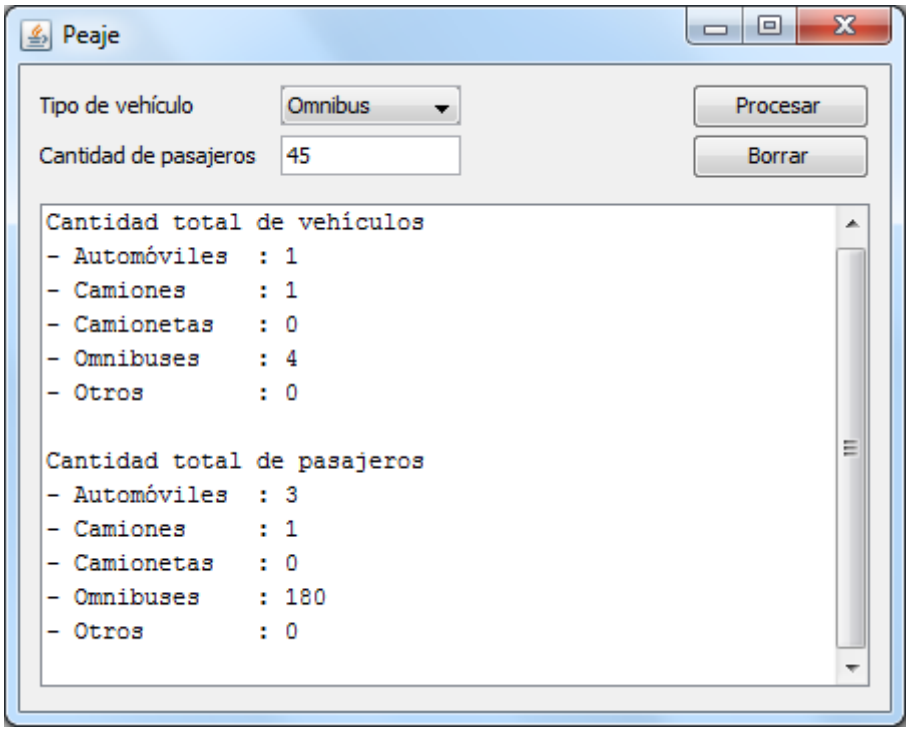
Problema 4

Diseñe un programa que lea, por cada vehículo, que pasa por un peaje, el tipo y la cantidad de pasajeros que transporta. Muestre, a continuación, un reporte indicando:

- La cantidad total de vehículos de cada tipo que pasaron por el peaje
- La cantidad total de pasajeros por cada tipo de vehículo

Los tipos de vehículos por considerar son: automóvil, camión, camioneta, omnibus y otros.

Solución




```
package cibertec;

import java.awt.EventQueue;
import java.awt.Font;
import javax.swing.JFrame;
```

```
import javax.swing.JButton;
import javax.swing.UIManager;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;

public class Peaje extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblTipoVehiculo;
    private JLabel lblCantidadPasajeros;
    private JComboBox<String> cboTipoVehiculo;
    private JTextField txtCantidadPasajeros;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // Declaración de variables globales para el algoritmo
    int canveh0, canveh1, canveh2, canveh3, canveh4;
    int totpas0, totpas1, totpas2, totpas3, totpas4;

    // Lanza la aplicación
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(
                    "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
        catch (Throwable e) {
            e.printStackTrace();
        }
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Peaje frame = new Peaje();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Peaje() {
        setTitle("Peaje");
        setBounds(100, 100, 450, 360);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        lblTipoVehiculo = new JLabel("Tipo de vehículo");
        lblTipoVehiculo.setBounds(10, 13, 120, 14);
```

```
        getContentPane().add(lblTipoVehiculo);

        lblCantidadPasajeros = new JLabel("Cantidad de pasajeros");
        lblCantidadPasajeros.setBounds(10, 38, 120, 14);
        getContentPane().add(lblCantidadPasajeros);

        cboTipoVehiculo = new JComboBox<String>();
        cboTipoVehiculo.setModel(new DefaultComboBoxModel<String>(new
                String[] {"Automóvil", "Camión", "Camioneta",
                "Omnibus", "Otro" }));
        cboTipoVehiculo.setBounds(130, 10, 90, 20);
        getContentPane().add(cboTipoVehiculo);

        txtCantidadPasajeros = new JTextField();
        txtCantidadPasajeros.setBounds(130, 35, 90, 20);
        getContentPane().add(txtCantidadPasajeros);
        txtCantidadPasajeros.setColumns(10);

        btnProcesar = new JButton("Procesar");
        btnProcesar.addActionListener(this);
        btnProcesar.setBounds(335, 9, 89, 23);
        getContentPane().add(btnProcesar);

        btnBorrar = new JButton("Borrar");
        btnBorrar.addActionListener(this);
        btnBorrar.setBounds(335, 34, 89, 23);
        getContentPane().add(btnBorrar);

        scpScroll = new JScrollPane();
        scpScroll.setBounds(10, 69, 414, 241);
        getContentPane().add(scpScroll);

        txtS = new JTextArea();
        txtS.setFont(new Font("Monospaced", Font.PLAIN, 12));
        scpScroll.setViewportView(txtS);
    }

    // Direcciona eventos de tipo ActionEvent
    public void actionPerformed(ActionEvent arg0) {
        if (arg0.getSource() == btnProcesar) {
            actionPerformedBtnProcesar(arg0);
        }
        if (arg0.getSource() == btnBorrar) {
            actionPerformedBtnBorrar(arg0);
        }
    }

    // Procesa la pulsación del botón Procesar
    protected void actionPerformedBtnProcesar(ActionEvent arg0) {
        int tipoveh, cantpas;
        tipoveh = getVehiculo();
        cantpas = getPasajeros();
        efectuarIncrementos(tipoveh, cantpas);
        mostrarReporte();
    }

    // Procesa la pulsación del botón Borrar
    protected void actionPerformedBtnBorrar(ActionEvent arg0) {
        txtCantidadPasajeros.setText("");
```

```
        txtS.setText("");
        txtCantidadPasajeros.requestFocus();
    }

    // Lee y retorna el tipo de vehículo
    int getVehiculo() {
        return cboTipoVehiculo.getSelectedIndex();
    }

    // Lee y retorna la cantidad de pasajeros
    int getPasajeros() {
        return Integer.parseInt(txtCantidadPasajeros.getText());
    }

    // Efectúa los incrementos necesarios
    void efectuarIncrementos(int tipv, int canp) {
        switch (tipv) {
            case 0:
                canveh0++;
                totpas0 += canp;
                break;
            case 1:
                canveh1++;
                totpas1 += canp;
                break;
            case 2:
                canveh2++;
                totpas2 += canp;
                break;
            case 3:
                canveh3++;
                totpas3 += canp;
                break;
            default:
                canveh4++;
                totpas4 += canp;
        }
    }

    // Muestra el reporte solicitado
    void mostrarReporte() {
        txtS.setText("");
        imprimir("Cantidad total de vehículos");
        imprimir("- Automóviles  : " + canveh0);
        imprimir("- Camiones     : " + canveh1);
        imprimir("- Camionetas   : " + canveh2);
        imprimir("- Omnibuses    : " + canveh3);
        imprimir("- Otros        : " + canveh4);
        imprimir("");
        imprimir("Cantidad total de pasajeros");
        imprimir("- Automóviles  : " + totpas0);
        imprimir("- Camiones     : " + totpas1);
        imprimir("- Camionetas   : " + totpas2);
        imprimir("- Omnibuses    : " + totpas3);
        imprimir("- Otros        : " + totpas4);
    }

    // Imprime una cadena con un salto de línea al final
    void imprimir(String cad) {
```

```
            txtS.append(cad + "\n");
        }
}
```