UNIDAD DE APRENDIZAJE
4

SEMANA
9

Métodos con valor de retorno

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al finalizar la unidad, el alumno, mediante el uso de métodos con o sin retorno, diseña programas de manera modular.

TEMARIO

1. Métodos con valor de retorno

ACTIVIDADES

Los alumnos desarrollan programas usando métodos con valor de retorno y tipo void.

1. Métodos con valor de retorno

Un método con valor de retorno es un módulo de programa que puede recibir datos de entrada a través de variables locales denominadas parámetros y que retorna un resultado al punto donde es invocado. Este tipo de método se utiliza para efectuar cualquier tipo de proceso que produzca un resultado

Estos métodos pueden dividirse a su vez en dos tipos:

- Métodos con valor de retorno sin parámetros
- Métodos con valor de retorno con parámetros

1.1 Métodos con valor de retorno sin parámetros

Este tipo de método no recibe datos de entrada através de parámetros; pero retorna un valor al punto donde es invocado.

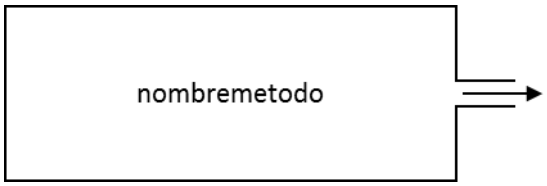


Figura 1 Método con valor de retorno sin parámetros

Definición

Este tipo de método se define de la siguiente manera:

```
tiporetorno nombremetodo ( ) {
        Declaración de variables locales
        Cuerpo del método
        return valor;
}
```

Donde:

`nombremetodo`	: Es el nombre del método.
`tiporetorno`	: Es el tipo del valor de retorno.
`valor`	: Es el valor de retorno.

Llamada

Este tipo de método se invoca de la siguiente manera:

```
variable = nombremetodo ( );
```

Donde:

`nombremetodo`	: Es el nombre del método invocado.
`variable`	: Es la variable que recibe el valor de retorno.

1.2 Métodos con valor de retorno con parámetros

Este tipo de método recibe datos de entrada através de parámetros y retorna un resultado al punto donde es invocado. En la Figura 2, p1 y p2 son parámetros.

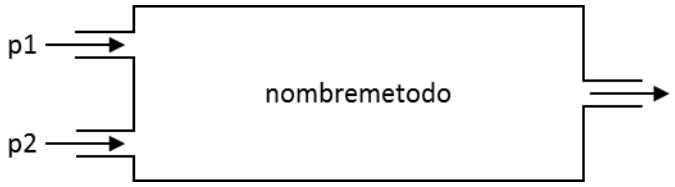


Figura 2 Método con valor de retorno y con parámetros

Definición

Este tipo de método se define de la siguiente manera:

```
tiporetorno nombremetodo (tipo1 p1, tipo2 p2, tipo3 p3, . . .) {
        Declaración de variables locales
        Cuerpo del método
        return valor;
}
```

Donde:

`nombremetodo`	: Es el nombre del método.
`tiporetorno`	: Es el tipo del valor de retorno.
`p1, p2, p3, ...`	: Son los nombres de los parámetros.
`tipo1, tipo2, tipo3, ...`	: Son los tipos de los parámetros.
`valor`	: Es el valor de retorno.

Llamada

Este tipo de método se invoca de la siguiente manera:

```
variable = nombremetodo (v1, v2, v3, ...);
```

Donde:

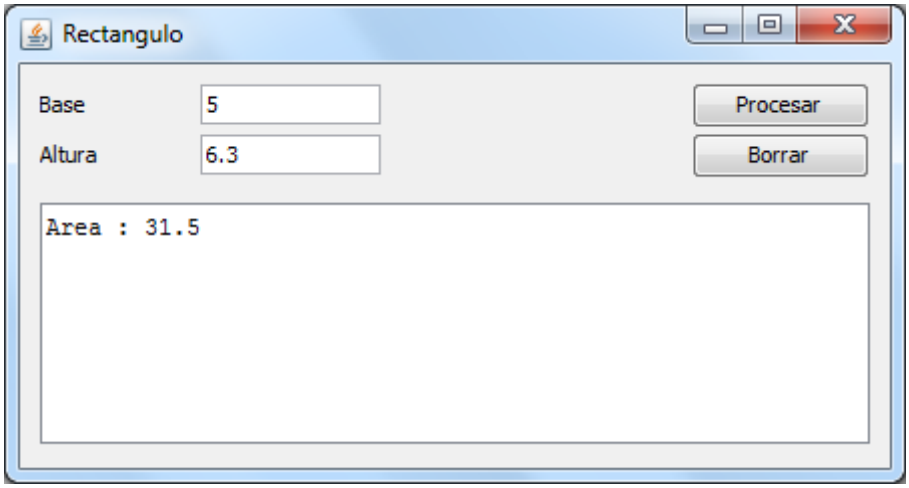
`nombremetodo`	: Es el nombre del método invocado.
`variable`	: Es la variable que recibe el valor de retorno.
`v1, v2, v3, ...`	: Son los valores dados a los parámetros.

2. Problemas resueltos

Problema 1

Diseñe un programa que determine el área de un rectángulo. Declare todas las variables como locales y use métodos para la entrada de datos, para el cálculo del área y para la salida de resultados.

Solucion 1




```
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 Rectangulo extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblBase;
    private JLabel lblAltura;
    private JTextField txtBase;
    private JTextField txtAltura;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // 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 {
                    Rectangulo frame = new Rectangulo();
                    frame.setVisible(true);
                }
                catch (Exception e) {
```

```
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Rectangulo() {
        setTitle("Rectangulo");
        setBounds(100, 100, 450, 239);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

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

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

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

        txtAltura = new JTextField();
        txtAltura.setBounds(90, 35, 90, 20);
        getContentPane().add(txtAltura);
        txtAltura.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, 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) {
    // Declaración de variables locales
    double b, h, arearec;

    // LLama a getBase
    // getBase envía como retorno la base del rectángulo
    // El valor retornado es almacenado en b
    b = getBase();

    // LLama a getAltura
    // getAltura envía como retorno la altura del rectángulo
    // El valor retornado es almacenado en h
    h = getAltura();

    // Llama a calcularArea y le envía los valores de b y h
    // El valor de b es recibido por el parámetro bas de calcularArea
    // El valor de h es recibido por el parámetro alt de calcularArea
    // calcularArea retorna el área del rectángulo
    // El valor retornado es almacenado en arearec
    arearec = calcularArea(b, h);

    // Llama a mostrarArea y le envía el valor de arearec
    // El valor de arearec es recibido por el parámetro are
    // mostrarArea retorna sin ningún resultado
    // Un método que no retorna un resultado es de tipo void
    mostrarArea(arearec);
}

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

// Lee y retorna la base
// Dado que la base es de tipo double, el tipo de retorno es double
double getBase() {
    return Double.parseDouble(txtBase.getText());
}

// Lee y retorna la altura
// Dado que la altura es de tipo double, el tipo de retorno es double
double getAltura() {
    return Double.parseDouble(txtAltura.getText());
}

// Calcula y retorna el área
// El parámetro bas recibirá el valor de la base
// El parámetro alt recibirá el valor de la altura
// Dado que el área es de tipo double, el tipo de retorno es double
// Los valores son enviados desde actionPerformedBtnProcesar
double calcularArea(double bas, double alt) {
    return bas * alt;
}
```

```
    // Muestra el área
    // El parámetro are recibirá el valor del área
    // El valor del área es enviado desde actionPerformedBtnProcesar
    // El método no retorna ningún resultado; por lo que, es de tipo void
    void mostrarArea(double are) {
        txtS.setText("Area : " + are);
    }
}
```

Solucion 2

```
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 Rectangulo extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblBase;
    private JLabel lblAltura;
    private JTextField txtBase;
    private JTextField txtAltura;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // 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 {
                    Rectangulo frame = new Rectangulo();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

```
    // Crea la GUI
    public Rectangulo() {
        setTitle("Rectangulo");
        setBounds(100, 100, 450, 239);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

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

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

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

        txtAltura = new JTextField();
        txtAltura.setBounds(90, 35, 90, 20);
        getContentPane().add(txtAltura);
        txtAltura.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, 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) {
        // Declaración de variables locales
        double b, h, areарec;
```

```
        // LLama a getBase
        // getBase envía como retorno la base del rectángulo
        // El valor retornado es almacenado en b
        b = getBase();

        // LLama a getAltura
        // getAltura envía como retorno la altura del rectángulo
        // El valor retornado es almacenado en h
        h = getAltura();

        // Llama a calcularArea y le envía los valores de b y h
        // El valor de b es recibido por el parámetro bas de calcularArea
        // El valor de h es recibido por el parámetro alt de calcularArea
        // calcularArea retorna el área del rectángulo
        // Dado que el área es de tipo double, el tipo de retorno es double

        // El valor retornado es almacenado en arearec
        arearec = calcularArea(b, h);

        // Llama a mostrarArea y le envía el valor de arearec
        // El valor de arearec es recibido por el parámetro are
        // mostrarArea retorna sin ningún resultado
        // Un método que no retorna un resultado es de tipo void
        mostrarArea(arearec);
    }

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

    // Lee y retorna la base
    // Dado que la base es de tipo double, el tipo de retorno es double
    // Declara una variable local xb para guardar la base temporalmente
    // Almacena la base en xb
    // Retorna la base almacenada en xb
    double getBase() {
        double xb;
        xb = Double.parseDouble(txtBase.getText());
        return xb;
    }

    // Lee y retorna la altura
    // Dado que la altura es de tipo double, el tipo de retorno es double
    // Declara una variable local xh para guardar la altura temporalmente
    // Almacena la altura en xh
    // Retorna la altura almacenada en xh
    double getAltura() {
        double xh;
        xh = Double.parseDouble(txtAltura.getText());
        return xh;
    }

    // Calcula y retorna el área
    // El parámetro bas recibirá el valor de la base
```

```
    // El parámetro alt recibirá el valor de la altura
    // Los valores son enviados desde actionPerformedBtnProcesar
    // Declara una variable local xa para guardar el área temporalmente
    // Almacena el área en xa
    // Retorna el área almacenada en xa
    double calcularArea(double bas, double alt) {
        double xa;
        xa = bas * alt;
        return xa;
    }

    // Muestra el área
    // El parámetro are recibirá el valor del área
    // El valor del área es enviado desde actionPerformedBtnProcesar
    // El método no retorna ningún resultado; por lo que, es de tipo void
    void mostrarArea(double are) {
        txtS.setText("Area : " + are);
    }
}
```

Problema 2

Una dulcería vende chocolates a los precios dados en la siguiente tabla:

Tipo de chocolate	Precio unitario
Primor	S/. 8.5
Dulzura	S/. 10.0
Tentación	S/. 7.0
Explosión	S/. 12.5

Como oferta, la tienda aplica un porcentaje de descuento sobre el importe de la compra, basándose en la cantidad de chocolates adquiridos, de acuerdo con la siguiente tabla:

Cantidad de chocolates	Descuento
< 5	4.0%
≥ 5 y < 10	6.5%
≥ 10 y < 15	9.0%
≥ 15	11.5%

Adicionalmente, si el importe a pagar es no menor de S/. 250, la tienda obsequia 3 caramelos por cada chocolate; en caso contrario, obsequia 2 caramelos por cada chocolate.

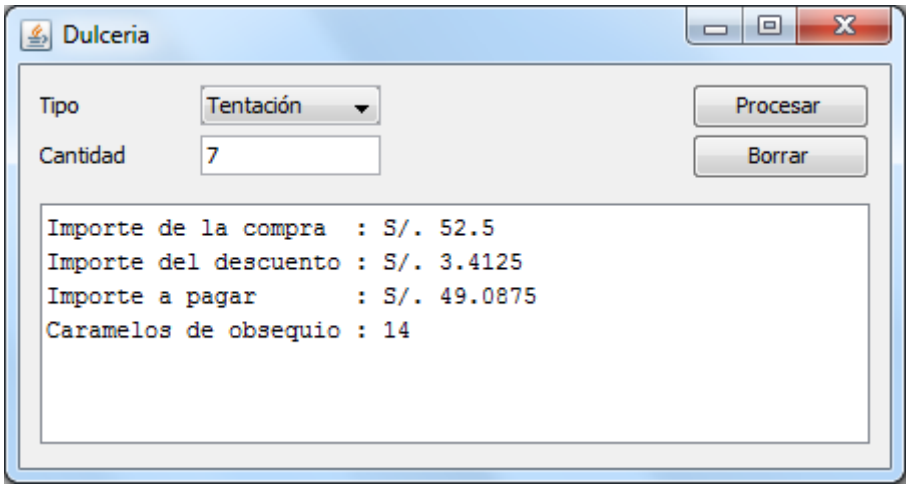
Dado el tipo de chocolate y la cantidad de unidades adquiridas, diseñe un programa que determine el importe de la compra, el importe del descuento, el importe a pagar y la cantidad de caramelos de obsequio.

Solución 1

```
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 Dulceria 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;

    // 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 {
                    Dulceria frame = new Dulceria();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
```

```
        });
    }

    // Crea la GUI
    public Dulceria() {
        setTitle("Dulceria");
        setBounds(100, 100, 450, 239);
        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[] {
                "Primor", "Dulzura", "Tentación", "Explosión" }));
        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, 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) {
    // Declaración de variables locales
    int tipo, cantidad, caramelos;
    double impcom, impdes, imppag;

    // Entrada de datos
    tipo = getTipo();
    cantidad = getCantidad();

    // Proceso de cálculo
    impcom = calcularImporteCompra(tipo, cantidad);
    impdes = calcularImporteDescuento(cantidad, impcom);
    imppag = calcularImportePagar(impcom,  impdes);
    caramelos = calcularCaramelos(cantidad, imppag);

    // Salida de resultados
    mostrarResultados(impcom, impdes, imppag, caramelos);
}

// Procesa la pulsación del botón Borrar
protected void actionPerformedBtnBorrar(ActionEvent arg0) {
    cboTipo.setSelectedIndex(0);
    txtCantidad.setText("");
    txtS.setText("");
    txtCantidad.requestFocus();
}

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

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

// Calcula y retorna el importe de la compra
double calcularImporteCompra(int tip, int can) {
    switch (tip) {
        case 0: return 8.5 * can;
        case 1: return 10.0 * can;
        case 2: return 7.5 * can;
        default: return 12.5 * can;
    }
}

// Calcula y retorna el importe del descuento
double calcularImporteDescuento(int can, double ic) {
    if (can < 5)
        return  0.04 * ic;
    else if (can < 10)
        return 0.065 * ic;
    else if (can < 15)
        return 0.09 * ic;
    else
        return 0.115 * ic;
}
```

```
    // Calcula y retorna el importe a pagar
    double calcularImportePagar(double ic, double id) {
        return ic - id;
    }

    // Calcula y retorna los caramelos de obsequio
    int calcularCaramelos(int can, double ip) {
        if (ip < 250)
            return 2 * can;
        else
            return 3 * can;
    }

    // Muestra los resultados obtenidos
    void mostrarResultados(double ic, double id, double ip, int car) {
        txtS.setText("");
        imprimir("Importe de la compra  : S/. " + ic);
        imprimir("Importe del descuento : S/. " + id);
        imprimir("Importe a pagar       : S/. " + ip);
        imprimir("Caramelos de obsequio : " + car);
    }

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

Solución 2

```
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 Dulceria 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;
```

```
    // 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 {
                    Dulceria frame = new Dulceria();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Dulceria() {
        setTitle("Dulceria");
        setBounds(100, 100, 450, 239);
        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[] {
                "Primor", "Dulzura", "Tentación", "Explosión" }));
        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, 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) {
        // Declaración de variables locales
        int tipo, cantidad, caramelos;
        double impcom, impdes, imppag;

        // Entrada de datos
        tipo = getTipo();
        cantidad = getCantidad();

        // Proceso de cálculo
        impcom = calcularImporteCompra(tipo, cantidad);
        impdes = calcularImporteDescuento(cantidad, impcom);
        imppag = calcularImportePagar(impcom,  impdes);
        caramelos = calcularCaramelos(cantidad, imppag);

        // Salida de resultados
        mostrarResultados(impcom, impdes, imppag, caramelos);
    }

    // Procesa la pulsación del botón Borrar
    protected void actionPerformedBtnBorrar(ActionEvent arg0) {
        cboTipo.setSelectedIndex(0);
        txtCantidad.setText("");
        txtS.setText("");
        txtCantidad.requestFocus();
    }

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

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

```
// Calcula y retorna el importe de la compra
double calcularImporteCompra(int tip, int can) {
    double xic;
    switch (tip) {
        case 0:
                xic = 8.5 * can;
                break;
        case 1:
                xic = 10.0 * can;
                break;
        case 2:
                xic = 7.5 * can;
                break;
        default:
                xic = 12.5 * can;
    }
    return xic;
}

// Calcula y retorna el importe del descuento
double calcularImporteDescuento(int can, double ic) {
    double xid;
    if (can < 5)
        xid =  0.04 * ic;
    else if (can < 10)
        xid = 0.065 * ic;
    else if (can < 15)
        xid = 0.09 * ic;
    else
        xid = 0.115 * ic;
    return xid;
}

// Calcula y retorna el importe a pagar
double calcularImportePagar(double ic, double id) {
    double xip;
    xip = ic - id;
    return xip;
}

// Calcula y retorna los caramelos de obsequio
int calcularCaramelos(int can, double ip) {
    int xcar;
    if (ip < 250)
        xcar = 2 * can;
    else
        xcar = 3 * can;
    return xcar;
}

// Muestra los resultados obtenidos
void mostrarResultados(double ic, double id, double ip, int car) {
    txtS.setText("");
    imprimir("Importe de la compra  : S/. " + ic);
    imprimir("Importe del descuento : S/. " + id);
    imprimir("Importe a pagar       : S/. " + ip);
    imprimir("Caramelos de obsequio : " + car);
}
```

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

Problema 3

En una universidad, los alumnos están clasificados en cuatro categorías. A cada categoría le corresponde una pensión mensual distinta dada en la siguiente tabla:

Categoría	Pensión
A	S/. 550
B	S/. 500
C	S/. 460
D	S/. 400

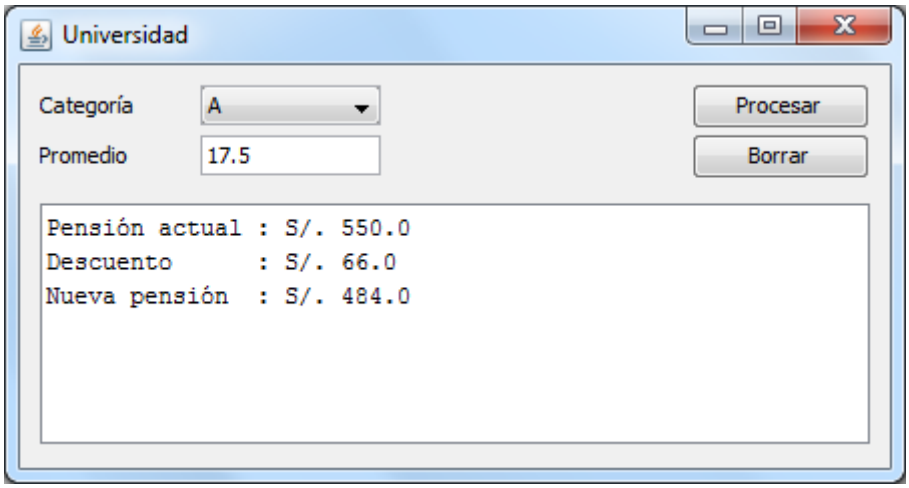
Semestralmente, la universidad efectúa rebajas en las pensiones de sus estudiantes a partir del segundo ciclo basándose en el promedio ponderado del ciclo anterior en porcentajes dados en la tabla siguiente:

Promedio	Descuento
0 a 13.99	No hay descuento
14.00 a 15.99	10 %
16.00 a 17.99	12 %
18.00 a 20.00	15 %

Dado el promedio ponderado y la categoría de un estudiante, diseñe un programa que determine cuánto de rebaja recibirá sobre su pensión actual y a cuánto asciende su nueva pensión.

Declare todas las variables como globales y use métodos tipo void.

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 Universidad extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

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

    // 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 {
                    Universidad frame = new Universidad();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Universidad() {
        setTitle("Universidad");
        setBounds(100, 100, 450, 239);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

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

```
        lblPromedio = new JLabel("Promedio");
        lblPromedio.setBounds(10, 38, 80, 14);
        getContentPane().add(lblPromedio);

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

        txtPromedio = new JTextField();
        txtPromedio.setBounds(90, 35, 90, 20);
        getContentPane().add(txtPromedio);
        txtPromedio.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, 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) {
        // Declaración de variables globales
        int categoria;
        double actualpen, nuevapen, descuento, promedio;

        // Entrada de datos
        categoria = getCategoria();
        promedio = getPromedio();

        // Proceso de cálculo
        actualpen = calcularPensionActual(categoria);
        descuento = calcularDescuento(promedio, actualpen);
        nuevapen = calcularNuevaPension(actualpen, descuento);
```

```
        // Salida deresultados
        mostrarResultados(actualpen, descuento, nuevapen);
    }

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

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

    // Lee y retorna el promedio ponderado
    double getPromedio() {
        return Double.parseDouble(txtPromedio.getText());
    }

    // Calcula y retorna la pensión actual
    double calcularPensionActual(int cat) {
        switch(cat){
            case  0: return 550.0;
            case  1: return 500.0;
            case  2: return 460.0;
            default: return 400.0;

        }
    }

    // Calcula y retorna el descuento
    double calcularDescuento(double pro, double apen) {
        if (pro <= 13.99)
            return 0;
        else if (pro <= 15.99)
            return 0.10 * apen;
        else if (pro <= 17.99)
            return 0.12 * apen;
        else
            return 0.15 * apen;
    }

    // Calcula y retorna la nueva pensión
    double calcularNuevaPension(double apen, double des) {
        return apen - des;
    }

    // Muestra los resultados obtenidos
    void mostrarResultados(double apen, double des, double npen) {
        txtS.setText("");
        imprimir("Pensión actual : S/. " + apen);
        imprimir("Descuento      : S/. " + des);
        imprimir("Nueva pensión  : S/. " + npen);
    }

    // Imprime una línea de texto incluyendo un salto de línea
```

```
    void imprimir(String cad) {
        txtS.append(cad + "\n");
    }
}
```

Problema 4

Una empresa evalúa a sus empleados bajo dos criterios: puntualidad y rendimiento. En cada caso el empleado recibe un puntaje que va de 1 a 10, de acuerdo con los siguientes criterios:

Puntaje por puntualidad:- está en función de los minutos de tardanza de acuerdo con la siguiente tabla:

Minutos de tardanza	Puntaje
0	10
1 a 2	8
3 a 5	6
6 a 9	4
Más de 9	0

Puntaje por rendimiento:- está en función de la cantidad de observaciones efectuadas al empleado por no cumplir sus obligaciones de acuerdo con la siguiente tabla:

Observaciones efectuadas	Puntaje
0	10
1	8
2	5
3	1
Más de 3	0

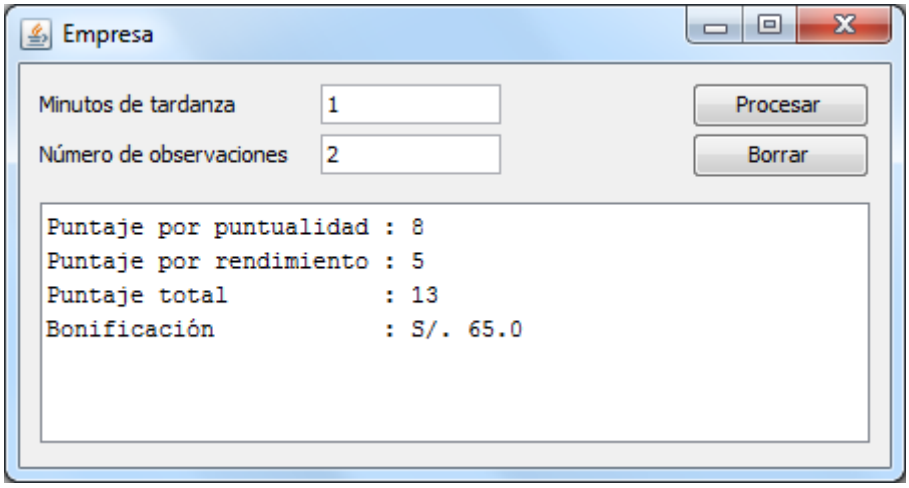
El puntaje total del empleado es la suma del puntaje por puntualidad más el puntaje por rendimiento. Basándose en el puntaje total, el empleado recibe una bonificación anual de acuerdo con la siguiente tabla:

Puntaje total	Bonificación
Menos de 11	S/. 2.5 por punto
11 a 13	S/. 5.0 por punto
14 a 16	S/. 7.5 por punto
17 a 19	S/. 10.0 por punto
20	S/. 12.5 por punto

Dados los minutos de tardanza y el número de observacciones de un empleado, diseñe un programa que determine el puntaje por puntualidad, el puntaje por rendimiento, el puntaje total y la bonificación que le corresponden.

Declare todas las variables como globales y use métodos tipo void.

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 Empresa extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblMinutosTardanza;
    private JLabel lblNumeroObservaciones;
    private JTextField txtMinutosTardanza;
    private JTextField txtNumeroObservaciones;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // 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 {
                    Empresa frame = new Empresa();
                    frame.setVisible(true);
```

```
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    // Crea la GUI
    public Empresa() {
        setTitle("Empresa");
        setBounds(100, 100, 450, 239);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);

        lblNumeroObservaciones = new JLabel("N\u00FAmero de observaciones");
        lblNumeroObservaciones.setBounds(10, 38, 140, 14);
        getContentPane().add(lblNumeroObservaciones);

        lblMinutosTardanza = new JLabel("Minutos de tardanza");
        lblMinutosTardanza.setBounds(10, 13, 140, 14);
        getContentPane().add(lblMinutosTardanza);

        txtMinutosTardanza = new JTextField();
        txtMinutosTardanza.setBounds(150, 10, 90, 20);
        getContentPane().add(txtMinutosTardanza);
        txtMinutosTardanza.setColumns(10);

        txtNumeroObservaciones = new JTextField();
        txtNumeroObservaciones.setBounds(150, 35, 90, 20);
        getContentPane().add(txtNumeroObservaciones);
        txtNumeroObservaciones.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, 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) {
        // Declaración de variables globales
        int minutosTar, numeroObs, puntajePun, puntajeRen, puntajeTot;
        double bonificacion;

        // Entrada de datos
        minutosTar = getMinutosTardanza();
        numeroObs = getNumeroObservaciones();

        // Proceso de cálculo
        puntajePun = determinarPuntajePuntualidad(minutosTar);
        puntajeRen = determinarPuntajeRendimiento(numeroObs);
        puntajeTot = determinarPuntajeTotal(puntajePun, puntajeRen);
        bonificacion = determinarBonificacion(puntajeTot);

        // Salida de resultados
        mostrarResultados(puntajePun, puntajeRen, puntajeTot, bonificacion);
    }

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

    // Lee y retorna los minutos de tardanza
    int getMinutosTardanza() {
        return Integer.parseInt(txtMinutosTardanza.getText());
    }

    // Lee y retorna el número de observaciones
    int getNumeroObservaciones() {
        return Integer.parseInt(txtNumeroObservaciones.getText());
    }

    // Determina y retorna el puntaje por puntualidad
    int determinarPuntajePuntualidad(int min) {
        if (min == 0)
            return 10;
        else if (min <= 2)
            return 8;
        else if (min <= 5)
            return 6;
        else if (min <= 9)
            return 4;
        else
            return 0;
    }

    // Determina y retorna el puntaje por rendimiento
    int determinarPuntajeRendimiento(int nob) {
        if (nob == 0)
            return 10;
```

```
        else if (nob == 1)
            return 8;
        else if (nob == 2)
            return 5;
        else if (nob == 3)
            return 1;
        else
            return 0;
    }

    // Determina y retorna el puntaje total
    int determinarPuntajeTotal(int pp, int pr) {
        return pp + pr;
    }

    // Determina y retorna la bonificación
    double determinarBonificacion(int pt) {
        if (pt < 11)
            return 2.5 * pt;
        else if (pt <= 13)
            return 5.0 * pt;
        else if (pt <= 16)
            return 7.5 * pt;
        else if (pt <= 19)
            return 10.0 * pt;
        else
            return 12.5 * pt;
    }

    // Muestra los resultados
    void mostrarResultados(int pp, int pr, int pt, double bo) {
        txtS.setText("");
        imprimir("Puntaje por puntualidad : " + pp);
        imprimir("Puntaje por rendimiento : " + pr);
        imprimir("Puntaje total           : " + pt);
        imprimir("Bonificación            : S/. " + bo);
    }

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

Problema 5

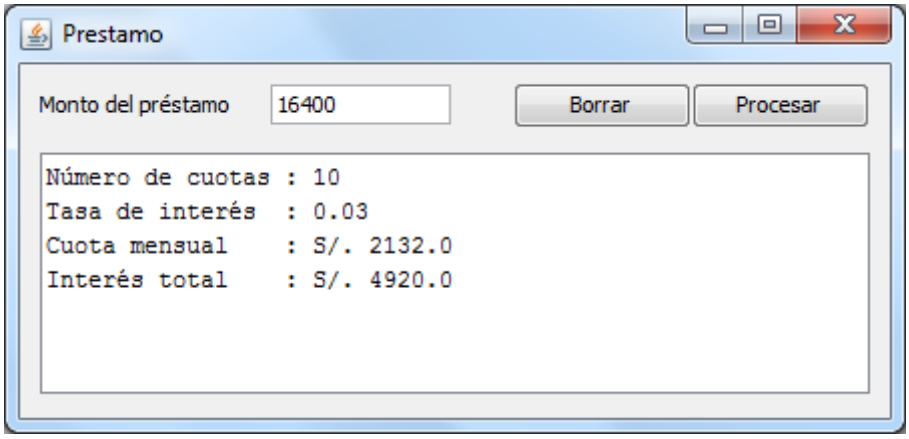
Una empresa de préstamos tiene el siguiente esquema de cobros:

Monto del préstamo (S/.)	Número de cuotas
Hasta 5000	2
Más de 5000 hasta 10000	4
Más de 10000 hasta 15000	6
Más de 15000	10

Si el monto del préstamo es mayor de S/. 10000, la empresa cobra 3% de interés mensual; en caso contrario, cobra 5% de interés mensual.

Dado el monto del préstamo de un cliente, diseñe un programa que determine el número de cuotas, el monto de la cuota mensual y el monto del interés total entre todas las cuotas.

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 Prestamo extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    // Declaración de variables
    private JLabel lblMontoPrestamo;
    private JTextField txtMontoPrestamo;
    private JButton btnProcesar;
    private JButton btnBorrar;
    private JScrollPane scpScroll;
    private JTextArea txtS;

    // 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 {
```

```
                    Prestamo frame = new Prestamo();
                    frame.setVisible(true);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

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

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

        lblMontoPrestamo = new JLabel("Monto del préstamo");
        lblMontoPrestamo.setBounds(10, 13, 115, 14);
        getContentPane().add(lblMontoPrestamo);

        txtMontoPrestamo = new JTextField();
        txtMontoPrestamo.setBounds(125, 10, 90, 20);
        getContentPane().add(txtMontoPrestamo);
        txtMontoPrestamo.setColumns(10);

        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) {
        // Declaración de variables globales
        double montopres, montointe, tasainte, montocuo;
```

```
        int cuotas;

        // Entrada de datos
        montopres = getMontoPrestamo();

        // Proceso de cálculo
        cuotas = obtenerNumeroCuotas(montopres);
        tasainte = obtenerTasaInteres(montopres);
        montointe = calcularMontoInteresTotal(tasainte, montopres, cuotas);
        montocuo = calcularMontoCuota(montopres, montointe, cuotas);

        // Salida deresultados
        mostrarResultados(cuotas, tasainte, montointe, montocuo);
    }

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

    // Lee y retorna el monto del préstamo
    double getMontoPrestamo() {
        return Double.parseDouble(txtMontoPrestamo.getText());
    }

    // Obtiene y retorna el número de cuotas
    int obtenerNumeroCuotas(double mpre) {
        if (mpre <= 5000)
            return 2;
        else if (mpre <= 10000)
            return 4;
        else if (mpre <= 15000)
            return 6;
        else
            return 10;
    }

    // Obtiene y retorna la tasa de interés
    double obtenerTasaInteres(double mpre) {
        if (mpre > 10000)
            return 0.03;
        else
            return 0.05;
    }

    // Calcula y retorna el monto del interés total
    double calcularMontoInteresTotal(double tint, double mpre, int ncuo) {
        return tint * mpre * ncuo;
    }

    // Calcula y retorna el monto de la cuota
    double calcularMontoCuota(double mpre, double mint, int ncuo) {
        return (mpre + mint) / ncuo;
    }

    // Muestra los resultados obtenidos
    void mostrarResultados(int ncuo, double tint, double mint, double mcuo) {
```

```
        txtS.setText("");
        imprimir("Número de cuotas : " + ncuo);
        imprimir("Tasa de interés  : " + tint);
        imprimir("Cuota mensual    : S/. " + mcuo);
        imprimir("Interés total    : S/. " + mint);
    }

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