UNIDAD DE APRENDIZAJE
4

SEMANA
8

Métodos tipo void

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. Programación modular
2. Variables locales y globales
3. Métodos tipo void

ACTIVIDADES

Los alumnos desarrollan programas mediante descomposición modular.

1. Programación modular

La programación modular es una metodología de programación que permite construir un programa grande descomponiéndolo en pequeños subprogramas o módulos. Para ello, se parte de un módulo principal que se descompone en varios submódulos que son controlados por el módulo principal. Si la tarea asignada a un módulo es demasiado compleja, este deberá descomponerse en otro módulos más pequeños hasta lograr módulos que hagan tareas relativamente sencillas. A este proceso de refinamiento sucesivo se conoce también como la técnica de “divide y vencerás”.

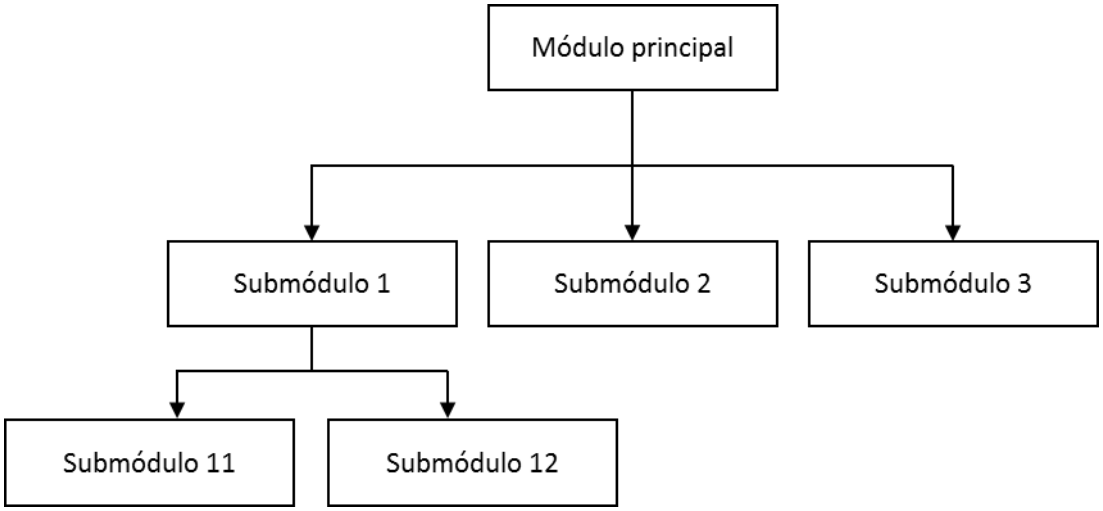


Figura 1 Descomposición modular de un programa

Las tareas asignadas a los subprogramas pueden ser de diversa índole: entrada, salida, cálculos, control de otros módulos, etc. Para que un subprograma pueda efectuar su tarea, tiene que ser llamado o invocado por el programa principal o por algún otro módulo que considere necesario el servicio del subprograma. Una vez que el subprograma termina su tarea, devuelve el control al punto donde se hizo la llamada. Un subprograma puede llamar, a su vez, a otros subprogramas.

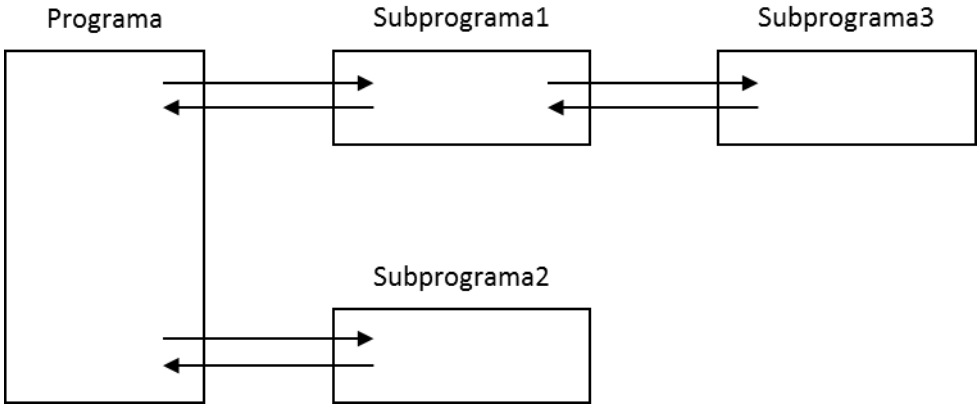


Figura 2 Un programa con diferentes niveles de subprograma.

En el lenguaje Java, a los módulos o subprogramas se denominan **métodos**, mientras que, en el lenguaje algorítmico, se denominan **subalgoritmos**.

2. Variables locales y globales

Los métodos pueden utilizar sus propias variables denominadas **variables locales** o variables de uso compartido, comunes a todos los métodos, denominadas **variables globales**.

2.1 Variables locales

Una *variable local* es una variable que se declara en el interior de un método por lo que su ámbito es el interior del método, es decir, sólo puede ser utilizada dentro del método donde fue declarada. Este tipo de variable se crea vacía al iniciar la ejecución del método y se destruye al finalizar la ejecución del método.

2.2 Variables globales

Una *variable global* es una variable que se declara dentro del programa, pero en el exterior de todos los métodos; por lo que, su ámbito es el interior de todo el programa, es decir, puede ser utilizada en cualquier parte del programa. Este tipo de variable se crea al iniciar la ejecución del programa y se destruye al finalizar. Por otro lado, una variable global se inicializa automáticamente: **0** si es de tipo **int**, **0.0** si es de tipo **double**, **false** si es de tipo **bolean**, **'\0'** si es de tipo **char** y **null** si es de tipo **String**.

3. Métodos tipo void

Un ***método tipo void*** es un módulo de programa que puede recibir datos de entrada a través de variables locales denominadas **parámetros**; pero que no retorna ningún resultado al punto donde es invocado, razón por el que se le conoce también como ***método sin valor de retorno***.

Los métodos tipo void pueden dividirse a su vez en dos tipos:

- Métodos tipo void sin parámetros
- Métodos tipo void con parámetros

3.1 Métodos tipo void sin parámetros

Estos métodos no pueden recibir datos de entrada ni retornar ningún resultado al punto de su invocación.

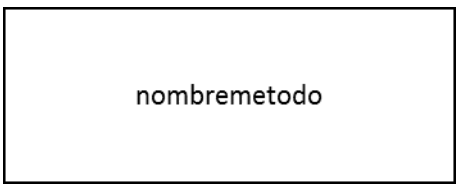


Figura 3 Método tipo void sin parámetros

Cuando se programa usando métodos, se siguen dos etapas. Primero, el método debe definirse. Esto consiste en crear el método ubicándolo en alguna parte del programa. Segundo, el método creado debe ser invocado en el lugar donde se requiera. Esto consiste en poner el método en ejecución.

Definición

Este tipo de método se define de la siguiente manera:

```
void nombremetodo() {
    Declaración de variables locales
    Cuerpo del método
}
```

Donde:

`nombremetodo` : Es el nombre del método.

Llamada

Este tipo de método se invoca de la siguiente manera:

```
nombremetodo();
```

Donde:

`nombremetodo` : Es el nombre del método invocado.

3.2 Métodos tipo void con parámetros

Estos métodos reciben datos de entrada a través de variables locales al método denominadas parámetros; pero, igual que en el caso anterior, no pueden retornar ningún resultado al punto de su invocación.

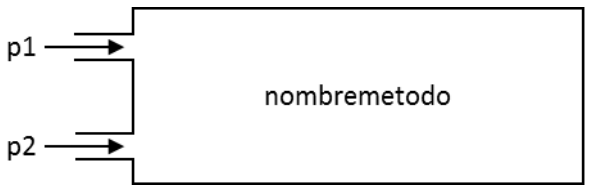


Figura 4 Método tipo void con parámetros

Donde **p1**, **p2**, etc son los parámetros del método. El número de parámetros es variable y depende de las necesidades del método.

Definición

Este tipo de método se define de la siguiente manera:

```
void nombremetodo(tipo1 p1, tipo2 p2, ...) {
    Declaración de variables locales
    Cuerpo del método
}
```

Donde:

`nombremetodo` : Es el nombre del método.
`p1, p2,...` : Son los nombres de los parámetros.

Llamada

Este tipo de método se invoca de la siguiente manera:

```
nombremetodo(v1, v2, ...);
```

Donde:

`nombremetodo` : Es el nombre del método invocado.
`v1, v2,...` : Son los valores pasados a los parámetros.

4. Problemas resueltos

Problema 1

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.

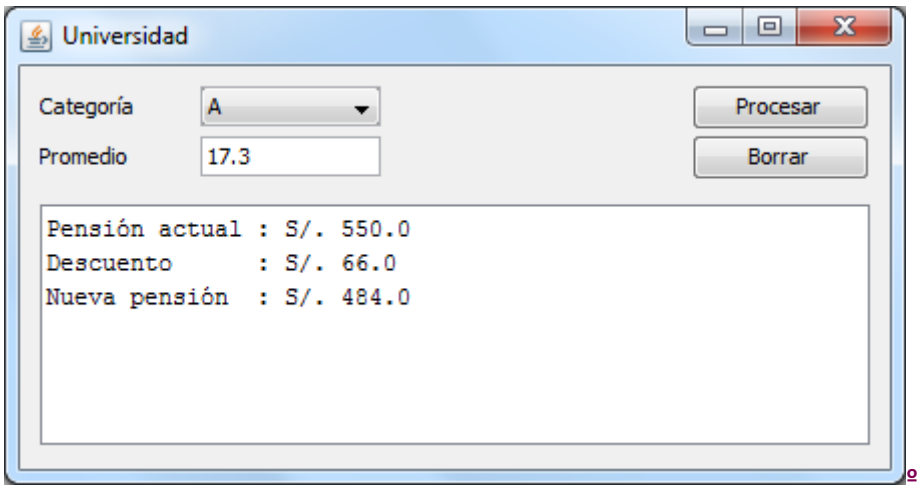
Declare todas las variables como globales y use métodos tipo void.

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;

    // Declaración de variables globales
    int categoria;
    double actualpen, nuevapen, descuento, promedio;

    // 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) {
    ingresarDatos();
    calcularPensionActual();
    calcularDescuento();
    calcularNuevaPension();
    mostrarResultados();
}

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

// Ingresa datos
void ingresarDatos() {
    categoria = cboCategoria.getSelectedIndex();
    promedio = Double.parseDouble(txtPromedio.getText());
}

// Calcula la pensión actual
void calcularPensionActual() {
    if (categoria == 0)
        actualpen = 550;
    else if (categoria == 1)
        actualpen = 500;
    else if (categoria == 2)
        actualpen = 460;
    else
        actualpen = 400;
}

// Calcula el descuento
void calcularDescuento() {
    if (promedio <= 13.99)
        descuento = 0;
    else if (promedio <= 15.99)
        descuento = 0.10 * actualpen;
    else if (promedio <= 17.99)
        descuento = 0.12 * actualpen;
    else
        descuento = 0.15 * actualpen;
}

// Calcula la nueva pensión
void calcularNuevaPension() {
    nuevapen = actualpen - descuento;
}

// Muestra resultados
void mostrarResultados() {
    txtS.setText("");
    imprimir("Pensión actual : S/. " + actualpen);
    imprimir("Descuento      : S/. " + descuento);
    imprimir("Nueva pensión  : S/. " + nuevapen);
}
```

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

Problema 2

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

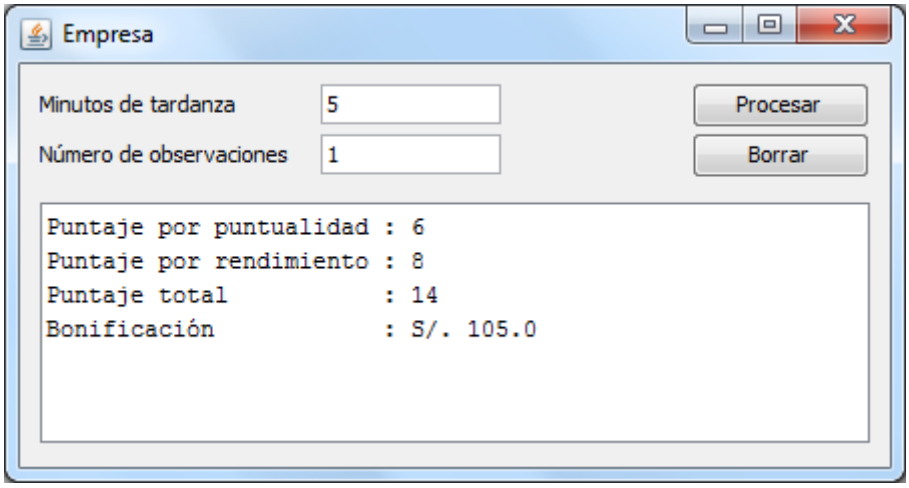
El puntaje total del empleado es la suma del puntaje por puntualidad más el puntaje por rendimiento. Basándose en el puntaje total, el empleado recibe una bonificación anual de acuerdo con la siguiente tabla:

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;

    // Declaración de variables globales
    int minutosTar, numeroObs, puntajePun, puntajeRen, puntajeTot;
    double bonificacion;

    // 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) {
        ingresarDatos();
        determinarPuntajePuntualidad();
        determinarPuntajeRendimiento();
        determinarPuntajeTotal();
        determinarBonificacion();
        mostrarResultados();
    }

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

    // Efectúa la entrada de datos
    void ingresarDatos() {
        minutosTar = Integer.parseInt(txtMinutosTardanza.getText());
        numeroObs = Integer.parseInt(txtNumeroObservaciones.getText());
    }

    // Determina el puntaje por puntualidad
    void determinarPuntajePuntualidad() {
        if (minutosTar == 0)
            puntajePun = 10;
        else if (minutosTar <= 2)
            puntajePun = 8;
        else if (minutosTar <= 5)
            puntajePun = 6;
        else if (minutosTar <= 9)
            puntajePun = 4;
        else
            puntajePun = 0;
    }

    // Determina el puntaje por rendimiento
    void determinarPuntajeRendimiento() {
        if (numeroObs == 0)
            puntajeRen = 10;
        else if (numeroObs == 1)
            puntajeRen = 8;
        else if (numeroObs == 2)
            puntajeRen = 5;
        else if (numeroObs == 3)
            puntajeRen = 1;
        else
            puntajeRen = 0;
    }
```

```
    // Determina el puntaje total
    void determinarPuntajeTotal() {
        puntajeTot = puntajePun + puntajeRen;
    }

    // Determina la bonificación
    void determinarBonificacion() {
        if (puntajeTot < 11)
            bonificacion = 2.5 * puntajeTot;
        else if (puntajeTot <= 13)
            bonificacion = 5.0 * puntajeTot;
        else if (puntajeTot <= 16)
            bonificacion = 7.5 * puntajeTot;
        else if (puntajeTot <= 19)
            bonificacion = 10.0 * puntajeTot;
        else
            bonificacion = 12.5 * puntajeTot;
    }

    // Muestra los resultados
    void mostrarResultados() {
        txtS.setText("");
        imprimir("Puntaje por puntualidad : " + puntajePun);
        imprimir("Puntaje por rendimiento : " + puntajeRen);
        imprimir("Puntaje total           : " + puntajeTot);
        imprimir("Bonificación            : S/. " + bonificacion);
    }

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

Problema 3

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:

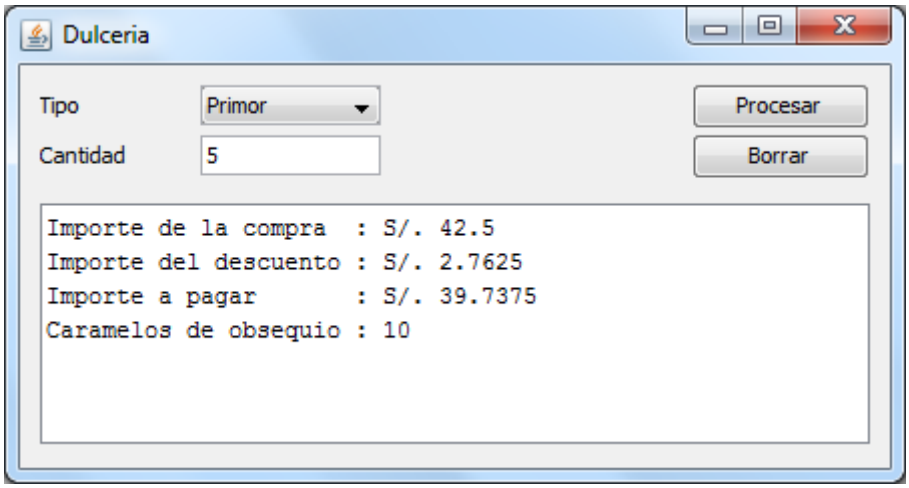
Cantidad de chocolates	Descuento
< 5	4.0%
≥ 5 y < 10	6.5%
≥ 10 y < 15	9.0%
≥ 15	11.5%

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.

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 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;

    // Declaración de variables globales
```

```
int tipo, cantidad, caramelos;
double impcom, impdes, imppag;

// 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) {
        ingresarDatos();
        calcularImporteCompra();
        calcularImporteDescuento();
        calcularImportePagar();
        calcularCaramelos();
        mostrarResultados();
    }

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

    // Efectúa la entrada de datos
    void ingresarDatos() {
        tipo = cboTipo.getSelectedIndex();
        cantidad = Integer.parseInt(txtCantidad.getText());
    }

    // Calcula el importe de la compra
    void calcularImporteCompra() {
        switch (tipo) {
            case 0:
                impcom = 8.5 * cantidad;
                break;
            case 1:
                impcom = 10.0 * cantidad;
                break;
            case 2:
                impcom = 7.0 * cantidad;
                break;
            default:
                impcom = 12.5 * cantidad;
```

```
        }
    }

    // Calcula el importe del descuento
    void calcularImporteDescuento() {
        if (cantidad < 5)
            impdes = 0.04 * impcom;
        else if (cantidad < 10)
            impdes = 0.065 * impcom;
        else if (cantidad < 15)
            impdes = 0.09 * impcom;
        else
            impdes = 0.115 * impcom;
    }

    // Calcula el importe a pagar
    void calcularImportePagar() {
        imppag = impcom - impdes;
    }

    // Calcula los caramelos de obsequio
    void calcularCaramelos() {
        if (imppag < 250)
            caramelos = 2 * cantidad;
        else
            caramelos = 3 * cantidad;
    }

    // Muestra los resultados obtenidos
    void mostrarResultados() {
        txtS.setText("");
        imprimir("Importe de la compra  : S/. " + impcom);
        imprimir("Importe del descuento : S/. " + impdes);
        imprimir("Importe a pagar       : S/. " + imppag);
        imprimir("Caramelos de obsequio : " + caramelos);
    }

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

}
```

Problema 4

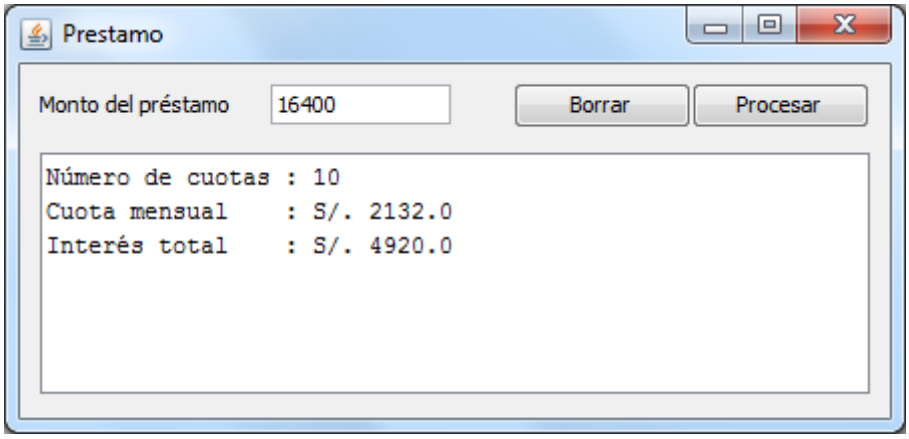
Una empresa de préstamos tiene el siguiente esquema de cobros:

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;

    // Declaración de variables globales
    double montoprestamo, montointeres, tasainteres, montocuota;
    int cuotas;

    // 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) {
    ingresarDatos();
    obtenerNumeroCuotas();
    obtenerTasaInteres();
    calcularMontoInteresTotal();
    calcularMontoCuota();
    mostrarResultados();
}

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

// Efectúa la entrada de datos
void ingresarDatos() {
    montoprestamo = Double.parseDouble(txtMontoPrestamo.getText());
}

// Obtiene el número de cuotas
void obtenerNumeroCuotas() {
    if (montoprestamo <= 5000)
        cuotas = 2;
    else if (montoprestamo <= 10000)
        cuotas = 4;
    else if (montoprestamo <= 15000)
        cuotas = 6;
    else
        cuotas = 10;
}

// Obtiene ka tasa de interés
void obtenerTasaInteres() {
    if (montoprestamo > 10000)
        tasainteres = 0.03;
    else
        tasainteres = 0.05;
}

// Calcula el monto del interés total
void calcularMontoInteresTotal() {
    montointeres = tasainteres * montoprestamo * cuotas;
}

// Calcula el monto de la cuota}
void calcularMontoCuota() {
    montocuota = (montoprestamo + montointeres) / cuotas;
}

// Muestra los resultados obtenidos
void mostrarResultados() {
    txtS.setText("");
    imprimir("Número de cuotas : " + cuotas);
    imprimir("Cuota mensual    : S/. " + montocuota);
    imprimir("Interés total    : S/. " + montointeres);
}
```

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