

ENCAPSULAMIENTO

Programación orientada a objetos, niveles de acceso, diseño de clases y creación de objetos.

CAPACIDAD DE PROCESO:

Diseña clases con niveles de acceso adecuados y crea objetos desde una Interface gráfica de usuario (GUI).

Programación orientada a objetos

La programación orientada a objetos es un nuevo enfoque de programación ya aceptado universalmente por la comunidad de programadores. Esta aceptación se sustenta en los múltiples beneficios que trae y los resume en sus siguientes características principales:

Abstracción, a través de la cual definimos y establecemos el contenido de una clase.

Encapsulamiento, a través de la cual establecemos los niveles de acceso a los componentes de una clase. El acceso a los atributos debe ser privado/protegido.

Herencia, a través de la cual se puede extender y/o agregar y/o modificar y/o reutilizar la funcionalidad de una clase existente, llamada padre, en otra clase llamada hija para evitar la redundancia en el código ya desarrollado. Esta característica resume la frase “aprovecha lo que ya está hecho”.

Polimorfismo, a través de la cual se puede programar lo genérico de algún proceso sin saber el detalle de cómo se va a llevar a cabo. Por ejemplo, podemos programar la secuencia genérica de un proceso de cálculo sin saber aún la forma del cálculo que se hará. Esta característica se orienta a la construcción de plantillas de uso general para que, a través de la herencia esas plantillas se puedan particularizar.

Niveles de acceso

Existen 3 niveles de acceso a los componentes de una clase: **private**, **protected**, **public**. El acceso **private** permite la implementación de la característica del

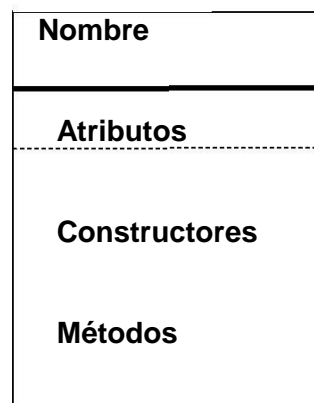
encapsulamiento, el acceso **protected** facilita la implementación de la característica de la herencia y el acceso **public** permite el acceso a la funcionalidad de la clase desde cualquier objeto. Los atributos de una clase deben ser de acceso **private** o **protected**, nunca **public**. Un constructor siempre debe ser de acceso **public**. Los métodos pueden ser de cualquier nivel de acceso, dependiendo de la apertura a la funcionalidad de la clase.

En la siguiente tabla se muestra la posibilidad de acceso a los componentes de una clase dependiendo de su nivel establecido.

Nivel de acceso	Desde la clase	Desde el objeto	Desde la clase hija
Private	SI	NO	NO
Protected	SI	NO	SI
public	SI	SI	SI

Diseño de clases

Una clase, en la práctica es un nuevo tipo de dato particularizado que se compone de: nombre, atributos, constructores y métodos. Los atributos representan los datos y/o características de la clase, los constructores permiten crear objetos de la clase y los métodos representan la funcionalidad de la clase que será utilizada por los objetos.



El nombre de una clase debe empezar con letra mayúscula. Los atributos deben ser de acceso **privado**/protegido y los métodos pueden ser de acceso **público** y/o protegido y/o **privado**.

Sintaxis general:

```
public class Nombre{  
  
    // atributos privados o protegidos  
  
    // constructores públicos  
  
    // métodos públicos y/o protegidos y/o privados  
  
}
```

Constructores

Un constructor es un componente de una clase que tiene el mismo nombre de la clase, de acceso público, a través del cual se puede inicializar los atributos de un objeto creado. Cuando una clase no tiene constructor explícito, se considera un constructor implícito por defecto, sin parámetros y con un bloque de código vacío.

Una clase puede tener varios constructores. Sin embargo, deben diferenciarse por la cantidad y/o tipo de parámetros (sobrecarga). Cuando se crea un objeto, se invoca automáticamente al constructor que corresponda a los parámetros dados. Si no encuentra un constructor adecuado, se produce un error.

Métodos

Los métodos de una clase expresan la funcionalidad con la que dispone una clase para el accionar de sus objetos. Por lo general son de acceso público.

Los métodos necesarios para el acceso de los atributos son conocidos como **get/set** y los demás métodos son conocidos como métodos adicionales que permiten realizar procesos de cálculo, búsqueda, clasificación, ordenamiento, etc.

Los métodos **get** son aquellos que permiten obtener el valor de los atributos. Los métodos **set** permiten modificar el valor de los atributos.

Estos métodos se han estandarizado, por eso se les conoce como los métodos **get/set**. Incluso, los entornos de programación modernos, como **NetBeans**, generan la escritura

del código correspondiente.

Palabra reservada **this**

La palabra reservada **this** se utiliza para diferenciar el nombre del atributo con el nombre del parámetro cuando éstos coinciden.

Creación de objetos

Un objeto, en la práctica es una variable cuyo tipo de dato es una clase. Cuando el objeto es creado, obtiene una copia de los atributos de la clase para uso exclusivo de dicho objeto. Por ésta razón, todos los objetos de una clase tendrán los mismos atributos pero cada uno con valores diferentes.

Sintaxis general:

```
// declara 2 objetos
Clase objeto1, objeto2;

// crea el objeto1, invoca al constructor sin parámetros
objeto1 = new Clase();

// crea el objeto2, invoca al constructor con parámetros
objeto2 = new Clase(valor1, valor2, ...);
```

Ejemplo 1

Diseña una clase, de nombre **Producto** con los siguientes atributos privados: código (cadena), descripción (cadena), precio (real) y los métodos públicos get/set. Considere un constructor implícito.

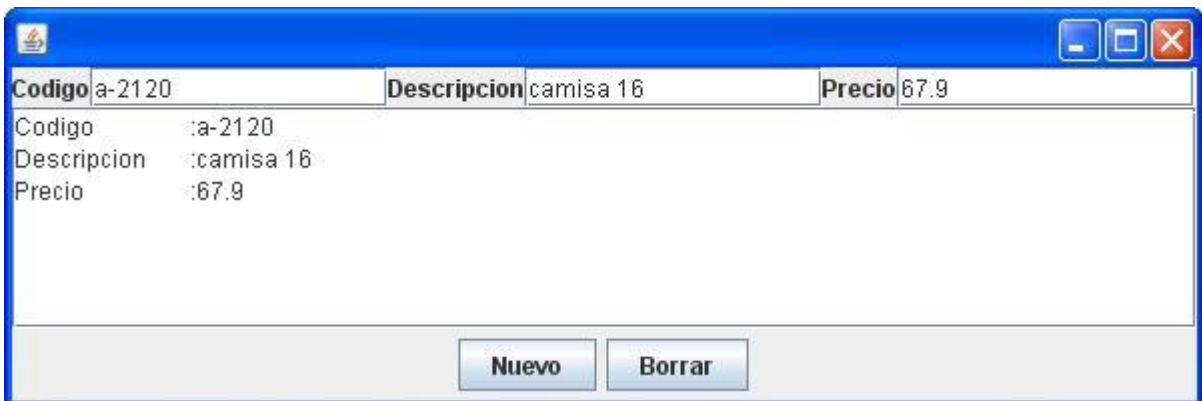
```
public class Producto
{
    // atributos
    privados
    private String codigo,
    descripcion; private double
    precio;

    // métodos get
```

```

    public String getCodigo(){return codigo;}
    public String getDescripcion(){return
    descripcion;}
    public double getPrecio(){ return precio;}
    // métodos set
    public void setCodigo(String
        c){ codigo =c;
    }
    public void setDescripcion(String
        d){ descripcion=d;
    }
    public void setPrecio(double
        p){ precio=p;
    }
}
    
```

Considere el siguiente diseño de una GUI y programe la acción del botón Nuevo



Codigo	Descripcion	Precio
a-2120	camisa 16	67.9

Codigo :a-2120
 Descripcion :camisa 16
 Precio :67.9

Nuevo Borrar

```

private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    // crea un objeto nuevo de la clase Producto
    Producto nuevo = new Producto();

    // lee los valores de la GUI y los coloca como atributos
    del objeto nuevo.setCodigo(leeCodigo());
    nuevo.setDescripcion(leeDescripcion());
    nuevo.setPrecio(leePrecio());

    // muestra la información del nuevo objeto
    lista(nuevo);
}
    
```

```
}  
  
// métodos que leen los datos de la GUI  
private String leeCodigo(){return txtCodigo.getText();}  
private String leeDescripcion(){return txtDescripcion.getText();}  
private double leePrecio(){return  
Double.parseDouble(txtPrecio.getText());}  
  
// método que muestra la información de un objeto de la clase Producto  
private void lista(Producto p){  
    imprime("Codigo\t:"+p.getCodigo());  
    imprime("Descripcion\t:"+p.getDescripcion()  
n());  
    imprime("Precio\t:"+p.getPrecio());  
}  
  
// método que muestra una cadena en el objeto de salida de  
la GUI private void imprime(String s){  
    txtSalida.append(s+"\n");  
}
```

Ejemplo 2

Diseña una clase de nombre **Persona** con los siguientes atributos privados: nombres (cadena), apellidos (cadena), edad(entero), peso(real) y los métodos get/set. Considere un constructor explícito con parámetros.

```
public class Persona {  
    // atributos  
    privados  
    private String nombres,  
    apellidos;  
    private int edad;  
    private double peso;  
  
    // constructor explícito con parámetros  
    public Persona(String nombres, String apellidos, int edad, double  
peso){  
        this.nombres = nombres;  
        this.apellidos=apellidos;
```

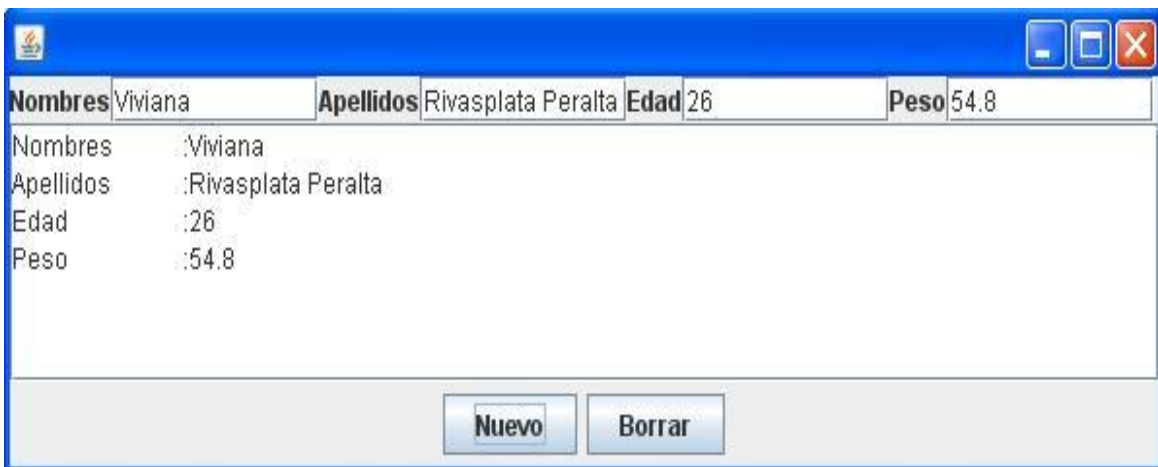
```

        this.edad = edad;
        this.peso = peso;
    }

    // métodos get
    public String getNombres(){return nombres;}
    public String getApellidos(){return
    apellidos;}
    public int getEdad(){return edad;}
    public double getPeso(){ return peso;}

    // métodos set
    public void setNombres(String nombres){
        this.nombres = nombres;
    }
    public void setApellidos(String apellidos){
        this.apellidos = apellidos;
    }
    public void setEdad(int edad){
        this.edad = edad;
    }
    public void setPeso(double
        peso){ this.peso = peso;
    }
}
    
```

Considere el siguiente diseño de una GUI y programe la acción del botón Nuevo



```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
```

```
// crea un objeto nuevo de la clase Persona
Persona nuevo = new Persona(leeNombres(), leeApellidos(),
                             leeEdad(), leePeso());

// muestra la información del nuevo objeto
lista(nuevo);
}

// métodos que leen los datos de la GUI
private String leeNombres(){return txtNombres.getText();}
private String leeApellidos(){return txtApellidos.getText();}
private int leeEdad(){return Integer.parseInt(txtEdad.getText());}
private double leePeso(){return
Double.parseDouble(txtPeso.getText());}

// método que muestra la información de un objeto de la clase Producto
private void lista(Persona p){
    imprime("Nombres\t:"+p.getNombres());
    imprime("Apellidos\t:"+p.getApellidos());
    ;
    imprime("Edad\t:"+p.getEdad());
    imprime("Peso\t:"+p.getPeso());
}

// método que muestra una cadena en el objeto de salida de
la GUI private void imprime(String s){
    txtSalida.append(s+"\n");
}
```

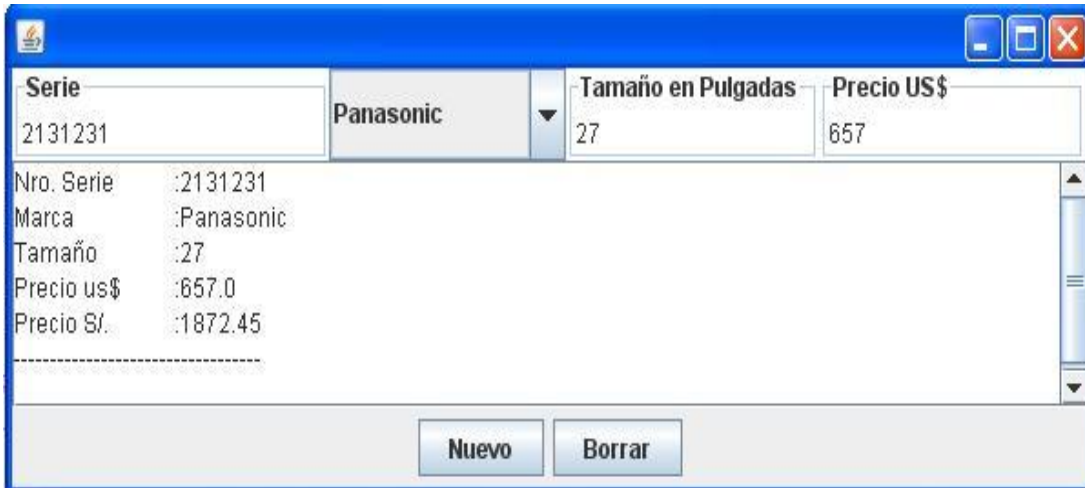
- Observe la utilidad y el funcionamiento del constructor explícito
- Una clase siempre va a considerar un constructor implícito?
- Puede una clase considerar varios constructores?
- Observe cuando se usa la palabra reservada this
- Observe cómo se genera el código para los métodos get/set

Ejemplo 3

Diseñe una clase de nombre **TV** con los siguientes atributos privados: serie (cadena), marca (entero), tamaño en pulgadas(entero), precio (real) y con los métodos get/set,

con un método adicional que devuelve el precio en soles dado el tipo de cambio como parámetro, y con un método adicional que retorne el nombre de la marca. Considere las siguientes marcas: Sony, LG, Samsung, Panasonic, otro. Considere un constructor implícito.

Considere el siguiente diseño de una GUI y programe la acción del botón Nuevo



Serie	Marca	Tamaño en Pulgadas	Precio US\$
2131231	Panasonic	27	657

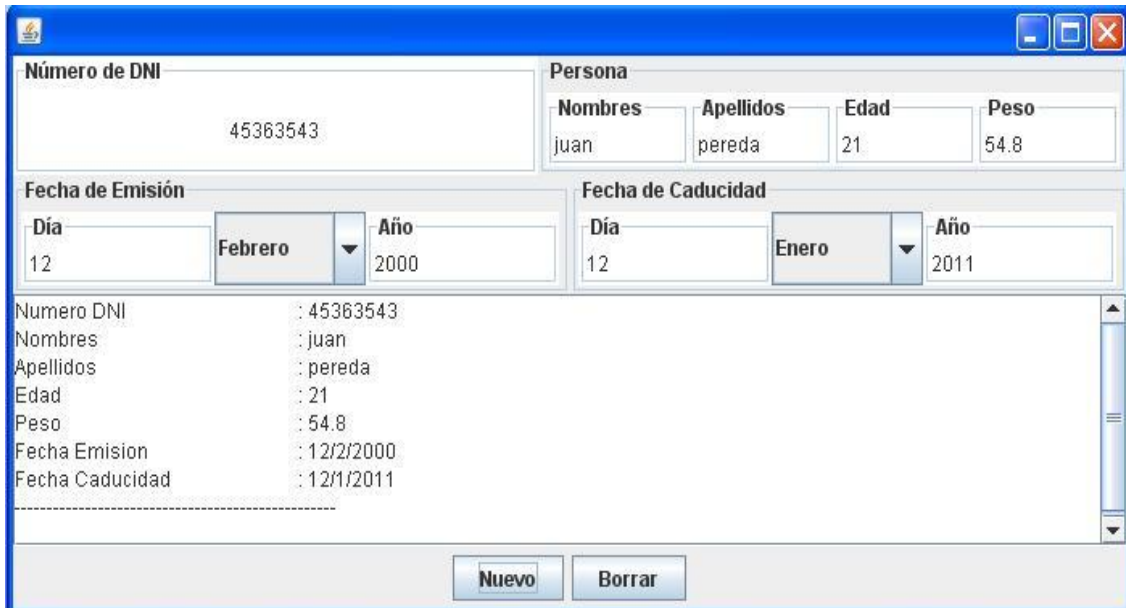
Nro. Serie	:2131231
Marca	:Panasonic
Tamaño	:27
Precio us\$:657.0
Precio S/.	:1872.45

Nuevo
Borrar

Ejemplo 4

Diseñe una clase de nombre **Fecha** con los siguientes atributos privados: dia(entero), mes(entero), año(entero) y con los métodos get/set. Considere un constructor explícito. Diseñe otra clase de nombre **DNI** con los siguientes atributos privados: numero(cadena), dueño(Persona), fecha de emisión (Fecha), fecha de caducidad(Fecha) y con los métodos get/set. Considere un constructor explícito.

Considere el siguiente diseño de una GUI y programe la acción del botón Nuevo.



Número de DNI

45363543

Persona

Nombres : juan **Apellidos** : pereda **Edad** : 21 **Peso** : 54.8

Fecha de Emisión

Día : 12 Mes : Febrero Año : 2000

Fecha de Caducidad

Día : 12 Mes : Enero Año : 2011

Numero DNI : 45363543
 Nombres : juan
 Apellidos : pereda
 Edad : 21
 Peso : 54.8
 Fecha Emision : 12/2/2000
 Fecha Caducidad : 12/1/2011

Nuevo **Borrar**

Ejemplo 5

Diseñe una clase de nombre **Punto** con los siguientes atributos privados: x (entero), y (entero), con un constructor explícito y con los métodos get/set.

Diseñe una clase de nombre **Triangulo** con los siguientes atributos privados: a(**Punto**), b(**Punto**), c(**Punto**), con un constructor implícito y con los métodos get/set además de los métodos de cálculo para los lados del triángulo, para el perímetro del triángulo y para el área del triángulo.

Considere el diseño de una GUI para programar la acción del botón nuevo.

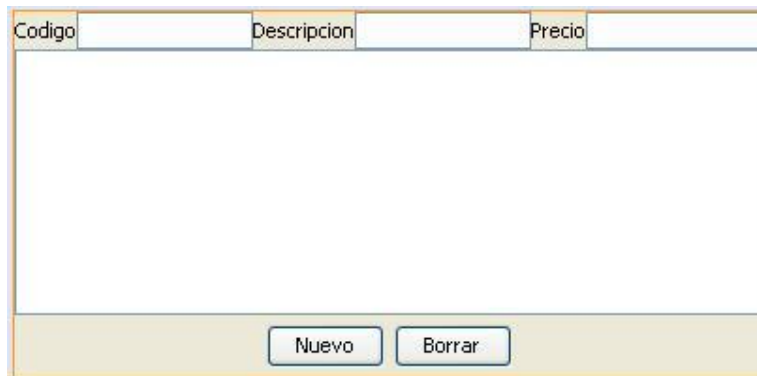
SEMANA 01

GUIA DE LABORATORIO 1

Encapsulamiento, diseño de clases y objetos

Ejercicio 1

Cree un proyecto nuevo de nombre **P01E01**. Cree un paquete nuevo de nombre **modelo** y diseñe una clase de nombre **Producto** con los siguientes atributos privados: código (cadena), descripción (cadena), precio (real) y con los métodos públicos get/set. Cree un nuevo paquete de nombre vista y diseñe la GUI en un Panel de nombre **PanelPrincipal** para el ingreso de los datos con el siguiente diseño:



Cada vez que hace clic en el botón **Nuevo** debe crear un objeto de la clase **Producto** y mostrar la información de sus atributos. Cada vez que hace clic en el botón **Borrar** debe borrar la información del área de texto, de las cajas de texto y enfocar el ingreso en la caja de texto del código.

Damos doble clic en el botón **Nuevo** para programar su acción:

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    Producto nuevo = new Producto();
    nuevo.setCodigo(leeCodigo());
    nuevo.setDescripcion(leeDescripcion());
    nuevo.setPrecio(leePrecio());
    lista(nuevo);
}

private String leeCodigo(){return txtCodigo.getText();}
private String leeDescripcion(){return txtDescripcion.getText();}
private double leePrecio(){return
Double.parseDouble(txtPrecio.getText());}
```

```
private void lista(Producto p){
    imprime("Codigo\t:"+p.getCodigo());
    imprime("Descripcion\t:"+p.getDescripcion(
    )); imprime("Precio\t:"+p.getPrecio());
}
private void imprime(String s){
    txtSalida.append(s+"\n");
}
```

Regresamos al diseño y hacemos doble clic en el botón **Borrar** para programar su acción:

```
private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt)
{
    txtSalida.setText("");
    txtCodigo.setText("");
    txtDescripcion.setText("");
    txtPrecio.setText("");
    txtCodigo.requestFocus();
}
```

En el paquete **vista** desarrolle la clase de aplicación de nombre **Principal** en un **JFrame**. Coloque distribución **BorderLayout**, vaya a la ficha **Source** y escriba lo que está en negrita:

```
public class Principal extends javax.swing.JFrame {
    public Principal() {
        initComponents();
        add(new PanelPrincipal());
        setSize(600,200);
    }
}
```

Ejecute su aplicación.

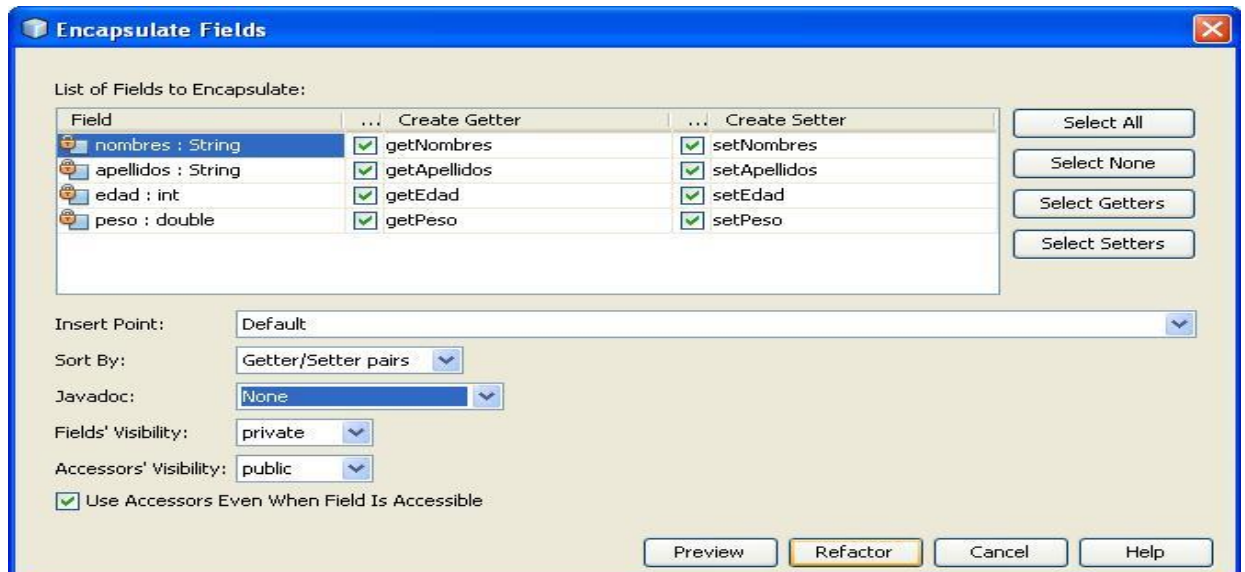
Ejercicio 2

Cree un proyecto nuevo de nombre **P01E02**. Cree un paquete nuevo de nombre **modelo** y diseñe una clase de nombre **Persona** con los siguientes atributos privados: nombres (cadena), apellidos (cadena), edad(entero), peso(real); con un constructor explícito y con los métodos get/set.

Clic derecho en el paquete creado, elegimos **New Java Class** y escribimos el nombre **Persona** y escribimos lo siguiente:

```
public class Persona {  
    // atributos privados  
    private String nombres, apellidos;  
    private int edad;  
    private double peso;  
  
    // constructor  
    public Persona(String nombres, String apellidos, int edad, double peso){  
        this.nombres=nombres;  
        this.apellidos=apellidos;  
        this.edad=edad;  
        this.peso=peso;  
    }  
}
```

Ahora, vamos a utilizar NetBeans para que genere, automáticamente, los métodos get/set. Clic derecho en la clase creada, elegimos **Refactor**, **Encapsulate Fields** y nos aparece la siguiente ventana donde seleccionamos todo (**Select All**).

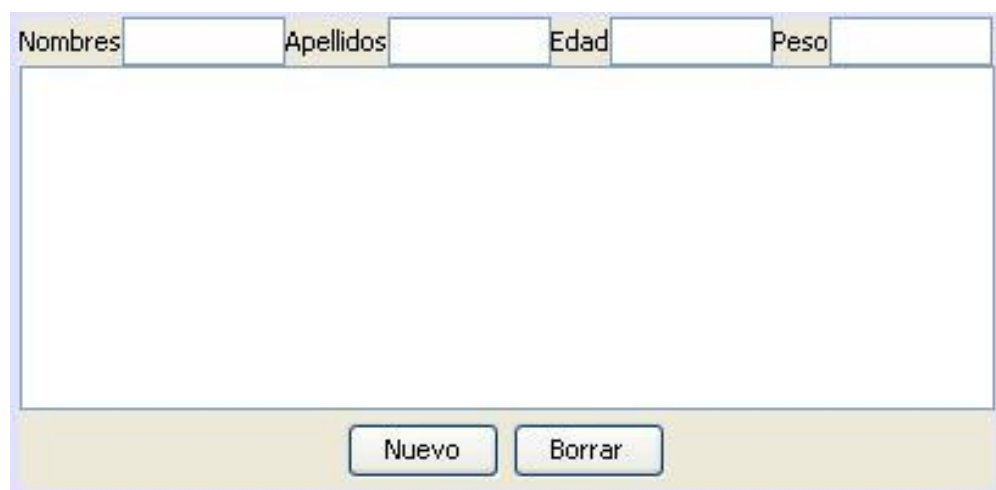


En **Javadoc** elegimos **none** para evitar los comentarios que genera NetBeans y por último hacemos clic en el botón **Refactor**. Ver los métodos get-set generados

Observe y analice el desarrollo del **constructor** y el uso de la palabra reservada **this**.

También puede generar el código del constructor y los métodos get-set a través de la opción **Insert code...** de netbeans.

Cree un nuevo paquete **vista** y diseñe la GUI en un Panel de nombre **PanelPrincipal** para el ingreso de los datos con el siguiente diseño:

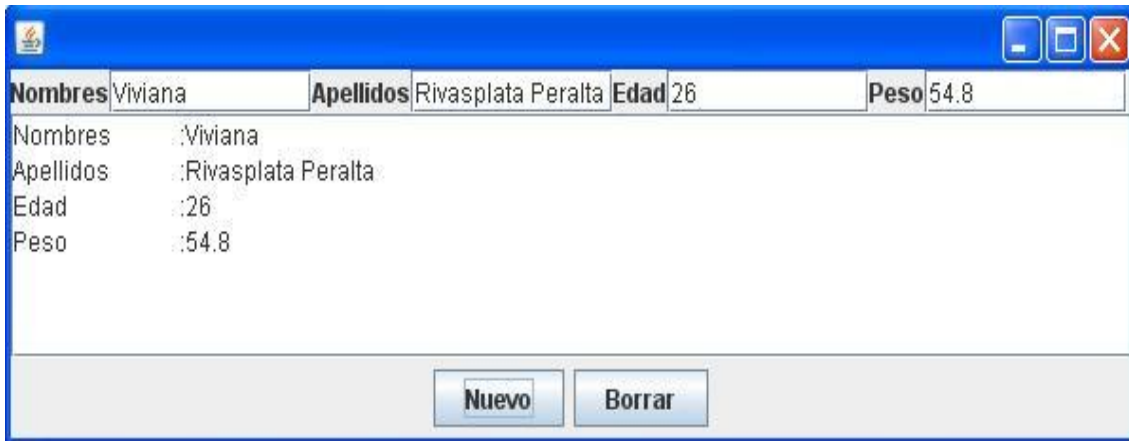


Cada vez que hace clic en el botón **Nuevo** debe crear un objeto de la clase **Persona** y

mostrar la información de sus atributos. Cada vez que hace clic en el botón **Borrar** debe borrar la información del área de texto, de las cajas de texto y enfocar el ingreso en la caja de texto de nombres.

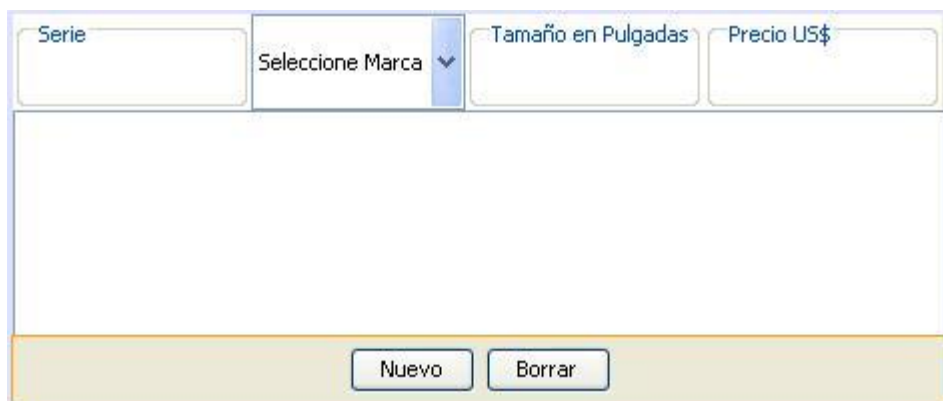
En el paquete **vista** desarrolle la clase de aplicación de nombre **Principal** en un **JFrame**

Ejecute su aplicación.



Ejercicio 3

Cree un proyecto nuevo de nombre **P01E03**. Cree un paquete nuevo de nombre **modelo** y diseñe una clase de nombre **TV** con los siguientes atributos privados: serie (cadena), marca (entero), tamaño en pulgadas(entero), precio (real) y con los métodos get/set, con un método adicional que devuelve el precio en soles dado el tipo de cambio como parámetro, y con un método adicional que retorne el nombre de la marca. Considere las siguientes marcas: Sony, LG, Samsung, Panasonic, otro. Cree un nuevo paquete de nombre **vista** y diseñe la GUI en un Panel de nombre **PanelPrincipal** para el ingreso de los datos con el siguiente diseño:



Cada vez que hace clic en el botón **Nuevo** debe crear un objeto de la clase **TV** y mostrar la información de sus atributos además de su precio en soles considerando un tipo de cambio actual. Cada vez que hace clic en el botón **Borrar** debe borrar la información del área de texto, de las cajas de texto y enfocar el ingreso en la caja de texto de serie.

Doble clic en el botón **Nuevo** para programar su acción:

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {  
    TV nuevo = new TV();  
    nuevo.setSerie(leeSerie());  
    nuevo.setMarca(leeMarca());  
    nuevo.setTamaño(leeTamaño());  
    nuevo.setPrecio(leePrecio());  
    lista(nuevo);  
}  
private String leeSerie(){return txtSerie.getText();}  
private int leeMarca(){return  
    cboMarca.getSelectedIndex();}  
private int leeTamaño(){return  
    Integer.parseInt(txtTamaño.getText());}  
private double leePrecio(){return  
    Double.parseDouble(txtPrecio.getText());}  
private void lista(TV t){  
    imprime("Nro. Serie\t:"+t.getSerie());  
    imprime("Marca\t:"+t.nombreMarca());  
    imprime("Tamaño\t:"+t.getTamaño());  
    imprime("Precio us$\t:"+t.getPrecio());  
    imprime("Precio  
    S/.\t:"+t.precioSoles(2.85));  
    imprime("-----");  
}  
private void imprime(String s){  
    txtSalida.append(s+"\n");  
}
```

Regrese al diseño y doble clic en el botón **Borrar** para programar su acción:


```
private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt)
{
    txtSalida.setText("");
    txtSerie.setText("");
    cboMarca.setSelectedIndex(0);
    txtTamaño.setText("");
    txtPrecio.setText("");
    txtSerie.requestFocus();
}
```

En el paquete **vista** desarrolle la clase de aplicación de nombre **Principal** en un JFrame.

Ejecute su aplicación.

Ejercicio 4

Cree un proyecto nuevo de nombre **P01E04**. Cree un paquete nuevo de nombre **modelo** y diseñe una clase de nombre **Fecha** con los siguientes atributos privados: dia(entero), mes(entero), año(entero) y con los métodos get/set. En el mismo paquete diseñe una clase de nombre **DNI** con los siguientes atributos privados: numero(cadena), dueño(Persona), fecha de emisión (Fecha), fecha de caducidad(Fecha) y con los métodos get/set.

Cree un nuevo paquete de nombre **vista** y diseñe la GUI en un Panel de nombre **PanelPrincipal** para el ingreso de los datos de un DNI. Cada vez que hace clic en el botón Nuevo debe crear un objeto de la clase **DNI** y mostrar la información de sus atributos.

En el mismo paquete **vista** desarrolle la clase de aplicación de nombre **Principal** en un JFrame.

Ejecute su aplicación