

HERENCIA

Herencia. Jerarquía de Clases, extends, super

CAPACIDAD EN PROCESO:

- Aplica el mecanismo de herencia construyendo una jerarquía de clases.
- Desarrolla métodos de procesos específicos con información de una lista dinámica de objetos.

Herencia

La Herencia es una de las características más importantes y beneficiosas para el enfoque de programación orientado a objetos. En resumen se trata de “utilizar lo que ya está hecho” para agregarle y/o cambiarle funcionalidad, nunca quitarle. Esto en la vida real tiene mucho sentido. Por ejemplo, Existiendo el televisor en blanco y negro (clase Padre) fabricaron el televisor a color (clase Hija) aprovechando todo lo bueno del televisor en blanco y negro ya implementado como cambio de canales, administración del volumen, mecanismo de encendido/apagado. Al televisor a color le agregaron más funcionalidad como: soportar audio, video, cable. Más herencia: Existiendo el televisor a color (clase Padre) fabricaron el televisor digital (clase Hija) al que le cambiaron y agregaron funcionalidad para el soporte de la tecnología digital. Más herencia: Existiendo el televisor digital (clase Padre) fabricaron el televisor Smart (clase Hija) al que le cambiaron y agregaron funcionalidad para el soporte de internet.

En programación sucede lo mismo, por ejemplo: Existiendo una clase Fecha (clase Padre) cuya funcionalidad soporta un formato de presentación DD/MM/AA, se puede construir una nueva clase FechaM (clase Hija) cuya funcionalidad soporte más formatos de presentación como DD/NOMBRE DEL MES/AAAA, AA/MM/DD, etc.

Jerarquía de Clases

La jerarquía de clases se va construyendo con la aplicación del concepto de herencia donde una clase padre extiende una clase hija y ésta a su vez extiende otra clase hija convirtiéndose en clase padre y así sucesivamente.

Como parte de la terminología de aplicación de herencia, en programación orientada a objetos, a la clase ya existente se le conoce como clase Padre ó clase Base ó clase Ascendiente o Super clase; en cambio, a la nueva clase que hereda se le conoce como clase Hija ó clase Derivada ó clase Descendiente o Subclase.

Cuando uno va a desarrollar por primera vez una clase y piensa que con el tiempo se podría utilizar como clase Padre, entonces el nivel de acceso a sus atributos deja de ser `prívate` y se debe considerar como `protected` ya que sigue permitiendo el encapsulamiento pero además dispone su acceso directo de todas sus clases descendientes. En cambio, cuando la clase que se va a heredar ya existe con un nivel de acceso `prívate` para sus atributos, entonces la clase hija tiene que utilizar la funcionalidad pública de acceso a sus atributos.

Extends

Es una palabra reservada que se utiliza para implementar el concepto de herencia. Significa **extensión** de una clase existente hacia una nueva clase aprovechando todo su contenido.

Super

Es una palabra reservada que se utiliza para referirse al constructor de la clase padre desde el constructor de la clase hija o también para referirse a algún método de la clase padre que también exista con el mismo nombre en la clase hija.

Si la clase Padre no tiene un constructor explícito, entonces la clase Hija no está obligada a tenerlo. Sin embargo, cuando la clase Padre sí tiene un constructor explícito, entonces la clase Hija sí está obligada a tener su constructor desde donde debe invocar al constructor de la clase Padre utilizando la palabra reservada **super**.

Ejemplo 1

Considere la existencia de la clase **TV** desarrollada anteriormente y diseñe una nueva clase hija de nombre **TVH**, aplicando herencia, considerando los siguientes atributos protegidos adicionales: `origen(entero)`, `tecnología(entero)`. Considere para el campo `origen`: nacional, americano, japonés, coreano, chino, otro. Considere para el campo `tecnología`: Tradicional, LCD, Plasma, Digital. Considere métodos adicionales para que

retornen el nombre del origen y el nombre de la tecnología.

La nueva clase hija debe ser utilizada para el proceso de la siguiente GUI:



Serie	LG	Pulgadas	Precio US\$	Koreano	LCD
423432		27	1567.5		

Datos de la clase Padre-----

Nro. Serie :423432
 Marca :LG
 Tamaño :27
 Precio us\$:1567.5
 Precio S/. :4467.375

Datos de la clase Hija-----

Origen :Koreano
 Tecnología :LCD

Nuevo Borrar

Diseño de la clase hija, aplicando herencia:

```
public class TVH extends TV {
    // atributos protegidos
    protected int origen,
    tecnologia;

    // constructor explícito
    public TVH(String serie, int marca, int tamaño, double
        precio, int origen, int tecnologia){
        // invocación al constructor de la clase
        Padre super(serie, marca, tamaño,
            precio);

        // inicializa sus propios
        atributos this.origen = origen;
        this.tecnologia=tecnologia;
    }

    // métodos get-set
    public int
    getOrigen() {
```

```
        return origen;
    }

    public void setOrigen(int
        origen) { this.origen =
        origen;
    }

    public int getTecnologia() {
        return tecnologia;
    }

    public void setTecnologia(int
        tecnologia) { this.tecnologia =
        tecnologia;
    }

    // metodos adicionales
    public String nombreOrigen(){
        switch(origen){
            case 1: return
                "Nacional"; case 2:
                return "Americano"; case
                3: return "Japonés";
            case 4: return
                "Koreano"; case 5:
                return "Chino"; default:
                return "Otro";
        }
    }

    public String
        nombreTecnologia(){
        switch(tecnologia){
            case 1: return
                "Tradicional"; case 2:
                return "LCD";
            case 3: return
                "Plasma"; default:
                return "Digital";
        }
    }
}
```

```
// fin de la clase hija
```

- La nueva clase hija tendrá acceso a todo lo **public** y **protected** de la clase padre. Sin embargo, no tendrá acceso a ningún elemento **private**.
- La nueva clase hija tiene sus atributos con nivel de acceso **protected** porque estamos previniendo que ésta sea heredada por otras clases.
- Observe y analice la aplicación de Herencia: **extends**, **super**

Programamos la acción del botón **Nuevo**:

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {  
    TVH nuevo = new  
        TVH(leeSerie(), leeMarca(), leeTamaño(), lee  
        Precio(),  
            leeOrigen(), leeTecnologia());  
    lista(nuevo);  
}  
private void lista(TVH t){  
    imprime("Datos de la clase Padre-----");  
    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("Datos de la clase Hija-----");  
    imprime("Origen\t:" + t.nombreOrigen());  
    imprime("Tecnologia\t:" + t.nombreTecnologia());  
    imprime("-----");  
}
```

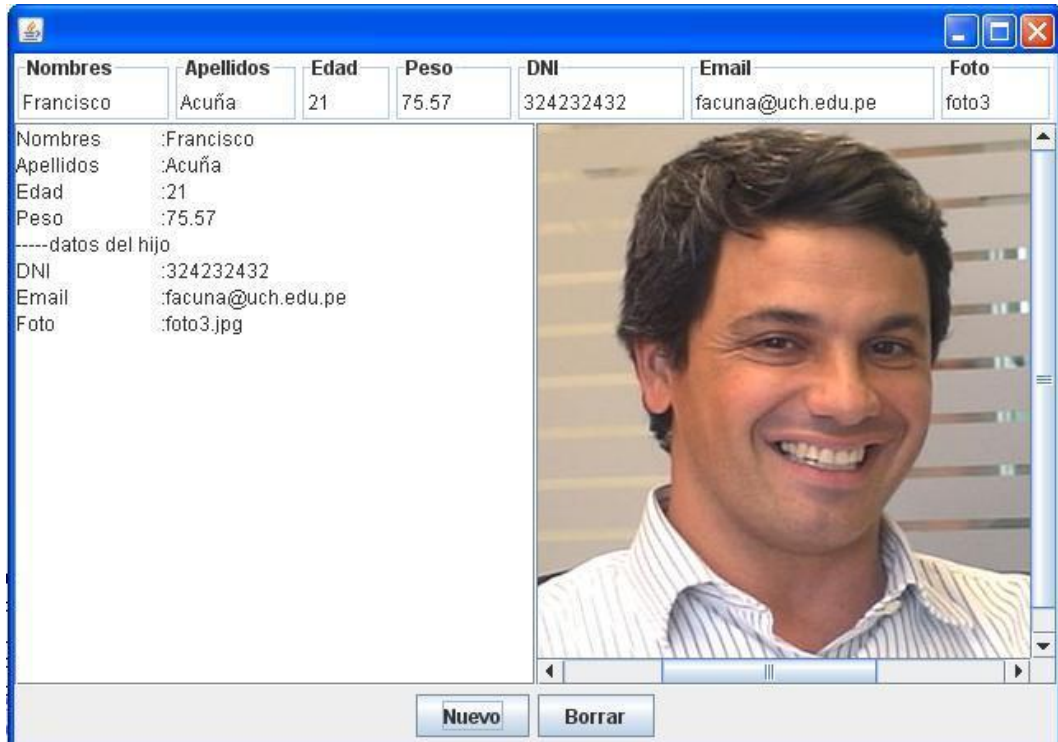
- Observe e identifique los datos que se utilizan al momento de crear el objeto.
- Observe y verifique la compatibilidad entre la invocación del método `lista()` y su desarrollo.
- Repase el desarrollo de los métodos de lectura de los datos de la GUI y también el método `imprime()`.

Ejemplo 2

Considere la existencia de una clase base de nombre **Persona** ya desarrollada

anteriormente y diseñe una nueva clase hija de nombre **PersonaH**, aplicando herencia, con los siguientes atributos protegidos adicionales: dni (cadena), email(cadena), foto(cadena).

La nueva clase hija debe ser utilizada para el proceso de la siguiente GUI:



Diseño de la clase hija aplicando herencia:

```

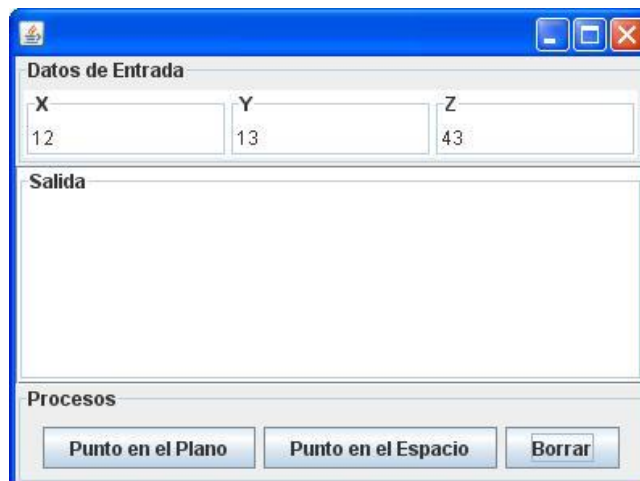
public class PersonaH extends Persona {
    // atributos protegidos
    protected String dni, email, foto;
    // constructor
    public PersonaH(String nombres, String apellidos, int edad, double
    peso,
                    String dni, String email, String
    foto){ super(nombres,apellidos,edad,peso);
    this.dni=dni;
    this.email
    =email;
    this.foto =foto;
    }
    
```

- Complete los métodos get-set.

- Observe y analice la aplicación de Herencia.
- Programe la acción del botón **Nuevo**.

Ejemplo 3

Diseñe una clase padre de nombre **PuntoP** cuyos atributos sean las coordenadas **x,y** de su ubicación en el plano. Considere un constructor con parámetros, la funcionalidad de acceso a sus atributos y un método adicional que retorne la distancia desde su ubicación hasta el punto de origen. Luego, diseñe una clase hija de nombre **PuntoE** que **herede** a la clase **PuntoP** y considere un atributo adicional para la coordenada **z** de su ubicación en el espacio. Considere un constructor con parámetros, la funcionalidad de acceso a su atributo y un método adicional que retorne la distancia desde su ubicación hasta el punto de origen. Considere una clase de GUI donde utilice objetos de ambas clases para mostrar su información.



Ejemplo 4

Diseñe la clase **ListaPilaTVH**, que administre una lista tipo pila de objetos tipo TVH y que considere los siguientes métodos adicionales:

- Incrementa el precio de todos los televisores en 8%
- Incrementa el precio de los televisores de una marca dada como parámetro en un porcentaje también dado como parámetro

- c) Disminuya el precio de los televisores cuyo tamaño actual esté en un rango dado como parámetro
- d) Retorne la cantidad de televisores de una marca y tamaño dado como parámetros.
- e) Retorne en un arreglo los televisores de una marca dada como parámetro.

Implemente, en la interfaz gráfica de usuario (GUI), lo necesario para incorporar a los métodos adicionales desarrollados

Ejemplo 5

Diseñe la clase **ListaColaPersonasH**, que administre una lista tipo cola de objetos tipo **PersonaH** y que considere los siguientes métodos adicionales:

- a) Incremente el peso de todas las personas en 5%
- b) Incremente el peso de las personas cuyo peso actual sea inferior a un valor dado como parámetro.
- c) Disminuya el peso de las personas cuyo peso actual esté en un rango dado como parámetro.
- d) Retorne la cantidad de personas menores de edad
- e) Retorne en un arreglo las personas cuya edad sea superior a un valor dado como parámetro.

Implemente, en la interfaz gráfica de usuario (GUI), lo necesario para incorporar a los métodos adicionales desarrollados.

GUIA DE LABORATORIO 5

Herencia

Ejercicio 1

Cree un proyecto nuevo de nombre **P05E01**. Cree un paquete nuevo de nombre **modelo**. Considere la existencia de la clase **TV** desarrollada anteriormente y diseñe una nueva clase hija de nombre **TVH**, aplicando herencia, considerando los siguientes atributos protegidos adicionales: origen(entero), tecnología(entero). Considere para el campo origen: nacional, americano, japonés, coreano, chino, otro. Considere para el campo tecnología: Tradicional, LCD, Plasma, Digital. Considere métodos adicionales para que retornen el nombre del origen y el nombre de la tecnología.

La nueva clase hija debe ser utilizada para el proceso de la siguiente GUI:



Diseño de la clase hija:

```

public class TVH extends TV {
    // atributos protegidos
    protected int origen, tecnologia;

    // constructor explícito
    public TVH(String serie, int marca, int tamaño, double precio,

```

```
        int origen, int tecnologia){
// invocación al constructor de la clase Padre
super(serie, marca, tamaño, precio);

// inicializa sus propios atributos
this.origen = origen;
this.tecnologia=tecnologia;
}

// métodos get-set
public int getOrigen() {
    return origen;
}
public void setOrigen(int origen) {
    this.origen = origen;
}
public int getTecnologia() {
    return tecnologia;
}
public void setTecnologia(int tecnologia) {
    this.tecnologia = tecnologia;
}

// metodos adicionales
public String nombreOrigen(){
    switch(origen){
        case 1: return "Nacional";
        case 2: return "Americano";
        case 3: return "Japonés";
        case 4: return "Koreano";
        case 5: return "Chino";
        default: return "Otro";
    }
}

public String
    nombreTecnologia(){
    switch(tecnologia){
        case 1: return "Tradicional";
        case 2: return "LCD";
```

```
        case 3: return "Plasma";  
        default: return "Digital";  
    }  
}  
} // fin de la clase hija
```

Cree un paquete nuevo de nombre **vista** y diseñe la GUI Principal donde considere como atributo un objeto de tipo TVH y programe la acción de los botones de acuerdo a lo siguiente:

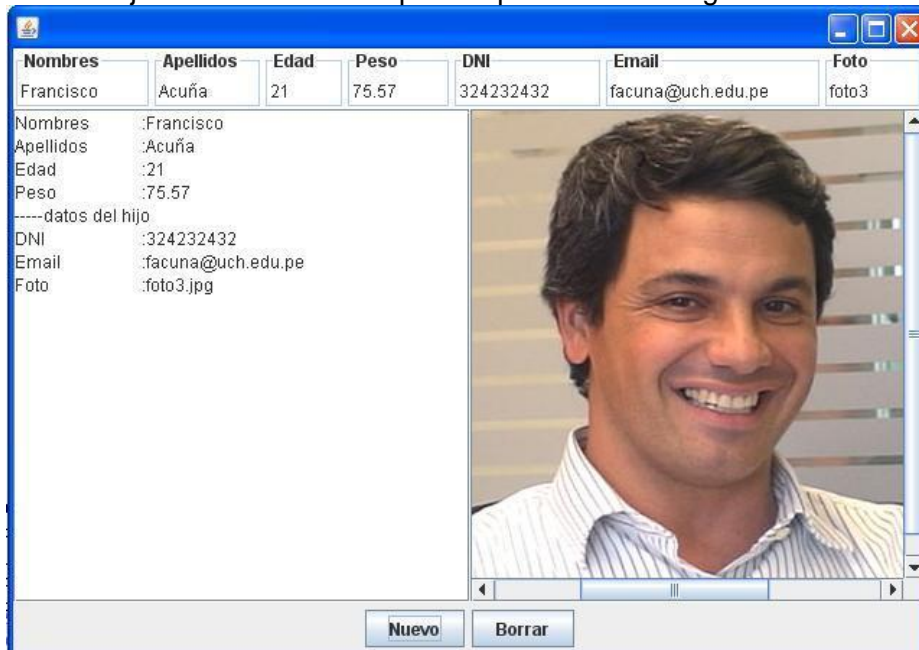
```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {  
    TVH nuevo = new  
        TVH(leeSerie(), leeMarca(), leeTamaño(), leePrecio(),  
            leeOrigen(), leeTecnologia());  
    lista(nuevo);  
}  
private void lista(TVH t){  
    imprime("Datos de la clase Padre-----");  
    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("Datos de la clase Hija-----");  
    imprime("Origen\t:" + t.nombreOrigen());  
    imprime("Tecnologia\t:" + t.nombreTecnologia());  
    imprime("-----");  
}
```

Diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

Ejercicio 2

Cree un proyecto nuevo de nombre **P05E02**. Cree un paquete nuevo de nombre **modelo**. Considere la existencia de una clase base de nombre **Persona** ya desarrollada anteriormente y diseñe una nueva clase hija de nombre **PersonaH**, aplicando herencia, con los siguientes atributos protegidos adicionales: dni (cadena), email(cadena), foto(cadena).

La nueva clase hija debe ser utilizada para el proceso de la siguiente GUI:



Diseño de la clase hija:

```

public class PersonaH extends Persona {
    // atributos protegidos
    protected String dni, email, foto;
    // constructor
    public PersonaH(String nombres, String apellidos, int edad, double
    peso,
                    String dni, String email, String
    foto){ super(nombres,apellidos,edad,peso);
    this.dni=dni;
    this.email
    =email;
    this.foto =foto;
    }
    
```

Cree un paquete nuevo de nombre **vista** y diseñe la GUI Principal donde considere como atributo un objeto de tipo PersonaH y programe la acción de los botones.

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

```

private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    PersonaH nuevo = new PersonaH(leeNombres(), leeApellidos(),
    leeEdad(),
    
```

```
        leePeso(), leeDNI(), leeEmail(),  
        leeFoto());  
  
    lista(nuevo);  
}  
  
private void lista(PersonaH p) {  
    imprime("Nombres\t:" + p.getNombres());  
    imprime("Apellidos\t:" + p.getApellidos());  
    imprime("Edad\t:" + p.getEdad());  
    imprime("Peso\t:" + p.getPeso());  
    imprime("-----datos del hijo");  
    imprime("DNI\t:" + p.getDni());  
    imprime("Email\t:" + p.getEmail());  
    imprime("Foto\t:" + p.getFoto());  
    lblImagen.setIcon(new  
    ImageIcon(getClass().getResource("/fotos/" + p.getFoto())));  
}
```

Doble clic en el botón **Borrar** para programar su acción.

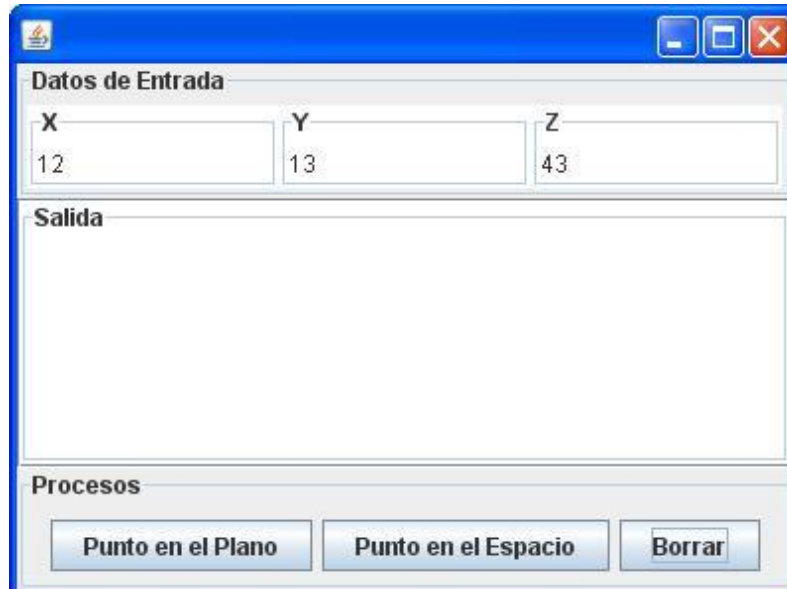
```
private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt)  
{  
    txtSalida.setText("");  
    txtNombres.setText("");  
    txtApellidos.setText("");  
    txtEdad.setText("");  
    txtPeso.setText("");  
    txtNombres.requestFocus();  
    lblImagen.setIcon(null);  
}
```

Diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

Ejercicio 3

Cree un proyecto nuevo de nombre **P05E03**. Cree un paquete nuevo de nombre **modelo** y diseñe una clase padre de nombre **PuntoP** cuyos atributos sean las coordenadas **x,y** de su ubicación en el plano. Considere un constructor con parámetros, la funcionalidad de acceso a sus atributos y un método adicional que retorne la distancia desde su ubicación hasta el punto de origen. Luego, diseñe una clase hija de nombre **PuntoE** que **herede** a la clase **PuntoP** y considere un atributo adicional para la coordenada **z** de su

ubicación en el espacio. Considere un constructor con parámetros, la funcionalidad de acceso a su atributo y un método adicional que retorne la distancia desde su ubicación hasta el punto de origen. Cree un paquete nuevo de nombre **vista** y considere una clase de GUI donde utilice objetos de ambas clases para mostrar su información.



Diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

Ejercicio 4

Cree un proyecto nuevo de nombre **P05E04**. Cree un paquete nuevo de nombre **controlador** y diseñe la clase **ListaPilaTVH**, que administre una lista tipo pila de objetos tipo TVH y que considere los siguientes métodos adicionales:

- Incremente el precio de todos los televisores en 8%
- Incremente el precio de los televisores de una marca dada como parámetro en un porcentaje también dado como parámetro
- Disminuya el precio de los televisores cuyo tamaño actual esté en un rango dado como parámetro
- Retorne la cantidad de televisores de una marca y tamaño dado como parámetros.
- Retorne en un arreglo los televisores de una marca dada como parámetro.

Cree un paquete nuevo de nombre **vista** e implemente, en la interfaz gráfica de usuario (GUI), lo necesario para incorporar a los métodos adicionales desarrollados.

En el paquete **vista** diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

Ejercicio 5

Cree un proyecto nuevo de nombre **P05E05**. Cree un paquete nuevo de nombre **controlador** y diseñe la clase **ListaColaPersonasH**, que administre una lista tipo cola de objetos tipo **PersonaH** y que considere los siguientes métodos adicionales:

- a) Incremente el peso de todas las personas en 5%
- b) Incremente el peso de las personas cuyo peso actual sea inferior a un valor dado como parámetro.
- c) Disminuya el peso de las personas cuyo peso actual esté en un rango dado como parámetro.
- d) Retorne la cantidad de personas menores de edad
- e) Retorne en un arreglo las personas cuya edad sea superior a un valor dado como parámetro.

Cree un paquete nuevo de nombre **vista** e implemente, en la interfaz gráfica de usuario (GUI), lo necesario para incorporar a los métodos adicionales desarrollados.

En el paquete **vista** diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.