POLIMORFISMO

Persistencia con archivos de texto

CAPACIDAD EN PROCESO:
Utiliza archivos de texto para la persistencia de información de los arreglos polimórficos

Persistencia con archivos de texto

La persistencia de datos se refiere a la forma de conservar los datos en el disco. Hay dos maneras de hacerlo: en archivos de texto y en bases de datos.

En esta parte utilizaremos archivos de texto para guardar los datos de los objetos polimórficos que se encuentren en las colecciones.

Ejemplo 1

Considere la existencia de la clase **ColeccionFiguras** desarrollada anteriormente y aplique herencia desarrollando la clase **ArchivoFiguras** que permita conservar la información de las figuras en un archivo de texto.

```
public class ArchivoFiguras extends ColeccionFiguras{
      protected String nombre;

      public ArchivoFiguras(String
            nombre){
            this.nombre=nombre;
            lee();
      }
      public void
      graba(){
      try{
            FileWriter fw = new
            FileWriter(nombre); PrintWriter pw
            = new FilePrinter(fw);
            for(Figura f : coleccion){
                  if(f instanceof
```

```
                Circulo)
                      pw.println("1/"+
                              ((Circulo)(f)).getRadio() );      //
                              molde
                else
                      pw.println("2/"+
                              ((Cuadrado)(f)).getLado() ); // molde
          }
          pw.close();
    }catch(Exception ex){   }
    }
    public void
    lee(){ try{
          FileReader fr = new
          FileReader(nombre); BufferedReader br
          = new BufferedReader(fr); String linea
          = br.readLine();
          while(linea!=null){
                StringTokenizer st =new
                StringTokenizer(linea,"/");
                int tipo=Integer.parseInt(st.nextToken());
                if(tipo==1){ // circulo
                      double
                      radio=Double.parseDouble(st.nextToken());
                      Figura a=new Circulo(radio);
                      agrega(a);
                }else{// cuadrado

                      double
                      lado=Double.parseDouble(st.nextToken());
                      Figura b=new Cuadrado(lado);
                      agrega(b);
                }
                linea=br.readLine();
          }
          br.close();
    }catch(Exception
    ex){}
```

```
        }
}
```

Utilizando la interface que se diseñó para administrar una colección de figuras polimórficas, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoFiguras** en lugar de la clase ColeccionFiguras y programe la acción de los botones, así:

```
// atributo de la clase interfaz
protected ArchivoFiguras af = new ArchivoFiguras();

Programación del botón Nuevo Circulo:{
        Figura a = new
        Circulo(leeRadio()); lista(a);
        af.agrega(a); // agrega un objeto Circulo
        af.graba(); // graba la información en el archivo de texto
}

Programación del botón Nuevo Cuadrado:{
     Figura b = new Cuadrado(leeLado());
     lista(b);
     af.agrega(b); // agrega un objeto Cuadrado
     af.graba();   // graba la información en el archivo de texto
}
```

Ejemplo 2

Considere la existencia de la clase **ColeccionEmpleados** desarrollada anteriormente y aplique herencia desarrollando la clase **ArchivoEmpleados** que permita conservar la información de los empleados en un archivo de texto.

Utilizando la interface que se diseñó para administrar una colección de empleados polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoEmpleados** en lugar de la clase ColeccionEmpleados y programe la acción de los botones.

Ejemplo 3

Considere la existencia de la clase **ColeccionCelulares** desarrollada anteriormente y aplique herencia desarrollando la clase **ArchivoCelulares** que permita conservar la información de los celulares en un archivo de texto.

Utilizando la interface que se diseñó para administrar una colección de celulares polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoCelulares** en lugar de la clase ColeccionCelularess y programe la acción de los botones.

Ejemplo 4

Considere la existencia de la clase **ColeccionVehiculos** desarrollada anteriormente y aplique herencia desarrollando la clase **ArchivoVehiculos** que permita conservar la información de los vehiclos en un archivo de texto.

Utilizando la interface que se diseñó para administrar una colección de vehiculos polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoVehiculos** en lugar de la clase ColeccionVehiculos y programe la acción de los botones.

GUIA DE LABORATORIO 12

Polimorfismo y persistencia con archivos de texto

Ejercicio 1

Cree un proyecto nuevo de nombre **P12E01**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de las clases Figura, Circulo y Cuadrado desarrolladas anteriormente. Cree un paquete nuevo de nombre **controlador** donde considere la existencia de la clase **ColeccionFiguras** desarrollada anteriormente. Aplique herencia desarrollando, en este mismo paquete, la clase **ArchivoFiguras** que permita conservar la información de las figuras en un archivo de texto.

```
public class ArchivoFiguras extends ColeccionFiguras{
	protected String nombre;

	public ArchivoFiguras(String
		nombre){
		this.nombre=nombre;
		lee();
	}
	public void graba(){
	try{
		FileWriter fw = new FileWriter(nombre);
		PrintWriter pw = new FilePrinter(fw);
		for(Figura f : coleccion){
			if(f instanceof Circulo)
				pw.println("1/"+
					((Circulo)(f)).getRadio() );// molde
			else
				pw.println("2/"+
					((Cuadrado)(f)).getLado() ); // molde
		}
		pw.close();
	}catch(Exception ex){   }
	}
```

```
	public void lee(){ try{
		FileReader fr = new FileReader(nombre);
		BufferedReader br = new BufferedReader(fr);
		String linea = br.readLine();
		while(linea!=null){
			StringTokenizer st =new
			StringTokenizer(linea,"/");
			int tipo=Integer.parseInt(st.nextToken());
			if(tipo==1){ // circulo
				double
				radio=Double.parseDouble(st.nextToken());
				Figura a=new Circulo(radio);
				agrega(a);
			}else{// cuadrado

				double
				lado=Double.parseDouble(st.nextToken());
				Figura b=new Cuadrado(lado);
				agrega(b);
			}
			linea=br.readLine();
		}
		br.close();
	}catch(Exception
	ex){}

	}
}
```

Cree un nuevo paquete de nombre **vista** y utilizando la interface que se diseñó para administrar una colección de figuras polimórficas, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoFiguras** en lugar de la clase ColeccionFiguras y programe la acción de los botones, así:

```
// atributo de la clase interfaz
protected ArchivoFiguras af = new ArchivoFiguras();
```

Programación del botón Nuevo Circulo:{

```
        Figura a = new
        Circulo(leeRadio()); lista(a);
        af.agrega(a); // agrega un objeto Circulo
        af.graba(); // graba la información en el archivo de texto
}

Programación del botón Nuevo Cuadrado:{
     Figura b = new Cuadrado(leeLado());
     lista(b);
     af.agrega(b); // agrega un objeto Cuadrado
     af.graba();   // graba la información en el archivo de texto
}
```

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

Ejercicio 2

Cree un proyecto nuevo de nombre **P12E02**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de las clases Empleado, EmpleadoPermanente y EmpleadoVendedor desarrolladas anteriormente. Cree un paquete nuevo de nombre **controlador** donde considere la existencia de la clase **ColeccionEmpleados** desarrollada anteriormente. Aplique herencia desarrollando, en este mismo paquete, la clase **ArchivoEmpleados** que permita conservar la información de los empleados en un archivo de texto.

Cree un nuevo paquete de nombre **vista** y utilizando la interface que se diseñó para administrar una colección de empleados polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoEmpleados** en lugar de la clase ColeccionEmpleados y programe la acción de los botones.

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

Ejercicio 3

Cree un proyecto nuevo de nombre **P12E03**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de las clases Celular, CelularSmart y Celular4G desarrolladas anteriormente. Cree un paquete nuevo de nombre **controlador** donde considere la existencia de la clase **ColeccionCelulares** desarrollada

anteriormente. Aplique herencia desarrollando, en este mismo paquete, la clase **ArchivoCelulares** que permita conservar la información de los celulares en un archivo de texto.

Cree un nuevo paquete de nombre **vista** y utilizando la interface que se diseñó para administrar una colección de celulares polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoCelulares** en lugar de la clase ColeccionCelularess y programe la acción de los botones.

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

Ejercicio 4

Cree un proyecto nuevo de nombre **P12E04**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de las clases Vehiculo, Automovil y Camion desarrolladas anteriormente. Cree un paquete nuevo de nombre **controlador** donde considere la existencia de la clase **ColeccionVehiculos** desarrollada anteriormente. Aplique herencia desarrollando, en este mismo paquete, la clase **ArchivoVehiculos** que permita conservar la información de los vehiclos en un archivo de texto.

Cree un nuevo paquete de nombre **vista** y ytilizando la interface que se diseñó para administrar una colección de vehiculos polimórficos, en un nuevo proyecto haga las modificaciones necesarias para utilizar la clase **ArchivoVehiculos** en lugar de la clase ColeccionVehiculos y programe la acción de los botones.

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