**SEMANA 10**

# POLIMORFISMO

## Coleccion de Objetos Polimórficos, instanceof

**CAPACIDAD EN PROCESO**:
Construye un arreglo de objetos polimórficos diferenciando los objetos particulares.

### Colección de objetos polimórficos

Estas colecciones son clases administradoras que se caracterizan porque soportan cualquier objeto de las clases descendientes de una jerarquía de clases, teniendo como clase **Padre** a una clase **abstracta**.

### Instanceof

Es una palabra reservada que permite identificar a que clase hija pertenece un objeto cuyo tipo de dato es una clase abstracta.

### Ejemplo 1

Considere la existencia de la clase abstracta de nombre **Figura,** las clases hijas **Cuadrado** y **Circulo.**

Diseñe una clase administradora de nombre **ColeccionFiguras** que permita la administración de objetos de tipo Cuadrado y/o Circulo a la vez, utilizando un objeto de la clase **ArrayList**.

```
public class ColeccionFiguras{
      protected ArrayList <Figura> coleccion;

      public ArregloFiguras(){
            coleccion = new ArrayList<Figura>();
      }
      // métodos de
      administración public
      void agrega(Figura f){
            coleccion.add(f);
```

```
        }
        public Figura obtiene(int i){  return coleccion.get(i); }
        public int tamaño(){ return coleccion.size();}
        public void elimina(int i){ coleccion.remove(i); }
        //... complete métodos adicionales
}
```

Diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de nombre **cf** de la clase **ColeccionFiguras**, como atributo de la clase.

```
Programación del botón Nuevo Circulo:{
        Figura a = new
        Circulo(leeRadio()); lista(a);
        cf.agrega(a); // agrega un objeto Circulo a la colección
}
Programación del botón Nuevo Cuadrado:{
     Figura b = new Cuadrado(leeLado());
        lista(b);
        cf.agrega(b); // agrega un objeto Cuadrado a la coleccion
}
public void lista(Figura
        f){
        imprime(f.info());
}
Programación del botón Lista:{
        for(int i=0; i<cf.tamaño();
        i++){
                Figura
                f=cf.obtiene(i);
                if (f instanceof
                Circulo)
                        imprime("Circulo: "+f.info());
                else
                        imprime("Cuadrado: "+f.info());
        }
}
```

**Ejemplo 2**

Considere la existencia de la clase abstracta de nombre **Empleado,** las clases hijas **EmpleadoVendedor** y **EmpleadoPermanente.**

Diseñe una clase administradora de nombre **ColeccionEmpleados** que permita la administración de objetos de tipo **EmpleadoVendedor** y/o **EmpleadoPermanente** a la vez, utilizando un objeto de la clase **ArrayList**.

Diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionEmpleados**, como atributo de la clase.

**Ejemplo 3**

Considere la existencia de la clase abstracta de nombre **Celular,** las clases hijas **CelularSmart** y **Celular4G.**

Diseñe una clase administradora de nombre **ColeccionCelulares** que permita la administración de objetos de tipo **CelularSmart** y/o **Celular4G** a la vez, utilizando un objeto de la clase **LinkedList**.

Diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionCelulares**, como atributo de la clase.

**Ejemplo 4:**

Considere la existencia de la clase abstracta de nombre **Vehiculo,** las clases hijas **Automovil** y **Camion.**

Diseñe una clase administradora de nombre **ColeccionVehiculos** que permita la administración de objetos de tipo **Automovil** y/o **Camion** a la vez, utilizando un objeto de la clase **LinkedList**.

Diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionVehiculos**, como atributo de la clase.

# GUIA DE LABORATORIO 10

## Polimorfismo, colecciones polimorficas

### Ejercicio 1

Cree un proyecto nuevo de nombre **P10E01**. Cree un paquete nuevo de nombre **modelo**. En esta paquete considere la existencia de la clase abstracta de nombre **Figura,** las clases hijas **Cuadrado** y **Circulo.**

Cree un nuevo paquete de nombre **controlador** y diseñe una clase administradora de nombre **ColeccionFiguras** que permita la administración de objetos de tipo Cuadrado y/o Circulo a la vez, utilizando un objeto de la clase **ArrayList**.

```
public class ColeccionFiguras{
      protected ArrayList <Figura> coleccion;

      public ArregloFiguras(){
            coleccion = new ArrayList<Figura>();
      }
      // métodos de
      administración public
      void agrega(Figura f){
            coleccion.add(f);
      }
      public Figura obtiene(int
            i){
            return
            coleccion.get(i);
      }

      public int tamaño(){ return coleccion.size();}

      public void elimina(int i){
            coleccion.remove(i);
      }
      //... complete métodos adicionales
```

```
}
```

Cree un nuevo paquete de nombre **vista** y diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionFiguras**, como atributo de la clase y programe la acción de los botones:

```
protected ColeccionFiguras cf = new ColeccionFiguras();

Programación del botón Nuevo Circulo:{
      Figura a = new
      Circulo(leeRadio()); lista(a);
      cf.agrega(a); // agrega un objeto Circulo a la colección
}
Programación del botón Nuevo Cuadrado:{
     Figura b = new Cuadrado(leeLado());
      lista(b);
      cf.agrega(b); // agrega un objeto Cuadrado a la coleccion
}
public void lista(Figura
      f){
      imprime(f.info());
}
Programación del botón Lista:{
      for(int i=0; i<cf.tamaño();
      i++){
            Figura
            f=cf.obtiene(i);
            if (f instanceof
            Circulo)
                  imprime("Circulo: "+f.info());
            else
                  imprime("Cuadrado: "+f.info());
      }
}
```

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

**Ejercicio 2**

Cree un proyecto nuevo de nombre **P10E02**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de la clase abstracta de nombre **Empleado,** las clases hijas **EmpleadoVendedor** y **EmpleadoPermanente.**

Cree un nuevo paquete de nombre **controlador** y diseñe una clase administradora de nombre **ColeccionEmpleados** que permita la administración de objetos de tipo **EmpleadoVendedor** y/o **EmpleadoPermanente** a la vez, utilizando un objeto de la clase **ArrayList**.

Cree un paquete nuevo de nombre **vista** y diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionEmpleados**, como atributo de la clase 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 **P10E03**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de la clase abstracta de nombre **Celular,** las clases hijas **CelularSmart** y **Celular4G.**

Cree un nuevo paquete de nombre **controlador** y diseñe una clase administradora de nombre **ColeccionCelulares** que permita la administración de objetos de tipo **CelularSmart** y/o **Celular4G** a la vez, utilizando un objeto de la clase **LinkedList**.

Cree un paquete nuevo de nombre **vista** y diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionCelulares**, como atributo de la clase y programe la acción de los botone.

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

**Ejercicio 4:**

Cree un proyecto nuevo de nombre **P10E04**. Cree un paquete nuevo de nombre **modelo**. En este paquete considere la existencia de la clase abstracta de nombre **Vehiculo,** las clases hijas **Automovil** y **Camion.**

Cree un nuevo paquete de nombre **controlador** y diseñe una clase administradora de nombre **ColeccionVehiculos** que permita la administración de objetos de tipo **Automovil** y/o **Camion** a la vez, utilizando un objeto de la clase **LinkedList**.

Cree un paquete nuevo de nombre **vista** y diseñe una clase de interfaz de nombre **PanelPrincipal** donde cree un objeto de la clase **ColeccionVehiculos**, como atributo de la clase y programe la acción de los botones.

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