Ejemplos de Interfaces

package Interface\_Persona;

public interface **InterfacePersona** {

public double calculaRiqueza();

public String getNombre();

}

public class **Persona implements InterfacePersona**

{

double dineroEnEfectivo;

double fondoPension;

String nombre;

public Persona(double dineroEnEfectivo, double fondoPension, String nombre) {

this.dineroEnEfectivo = dineroEnEfectivo;

this.fondoPension = fondoPension;

this.nombre = nombre;

}

@Override

public double calculaRiqueza(){

return this.dineroEnEfectivo+this.fondoPension;

}

@Override

public String getNombre() {return this.nombre;}

}

public class **Interfaces\_Lab\_2** {

public static void main(String[] args) {

Persona p1=new Persona(1000,2000,"Pepe");

InterfacePersona interfacePersona=p1;

System.out.println("p1 "+p1.getNombre());

System.out.println("interfacePersona "+interfacePersona.getNombre());

System.out.println("riqueza de p1 "+p1.calculaRiqueza());

System.out.println("riqueza de interfacePersona "+interfacePersona.calculaRiqueza());

boolean iguales=(p1==interfacePersona);

System.out.println("¿p1 e interfacePersona apuntan al mismo objeto? "+iguales);

InterfacePersona in2=new Persona(3000,5000,"Margarita");

System.out.println("in2 "+in2.getNombre());

System.out.println("riqueza de in2 "+in2.calculaRiqueza());

}

}

package Interface\_Relacion;

public interface **Relacion** {

public boolean esMayor(Object o1, Object o2);

public boolean esMenor(Object o1, Object o2);

public boolean esIgual(Object o1, Object o2);

}

public class **Linea implements Relacion{**

int x1;

int x2;

int y1;

int y2;

public Linea(int x1, int x2, int y1, int y2) {

this.x1 = x1;

this.x2 = x2;

this.y1 = y1;

this.y2 = y2;

}

public double distancia(){

double d=Math.sqrt((x2-x1)\*(x2-x1) + (y2-y1)\*(y2-y1));

return d;

}

public boolean esMayor( Object a, Object b){

double aLen = ((Linea)a).distancia();

double bLen = ((Linea)b).distancia();

return (aLen > bLen);

}

public boolean esMenor( Object a, Object b){

double aLen = ((Linea)a).distancia();

double bLen = ((Linea)b).distancia();

return (aLen < bLen);

}

public boolean esIgual( Object a, Object b){

double aLen = ((Linea)a).distancia();

double bLen = ((Linea)b).distancia();

return (aLen == bLen);

}

}

public class **Principal** {

public static void main(String[] args) {

Linea linea1 = new Linea(1, 2, 1, 2);

Linea linea2 = new Linea(2, 3, 2, 3);

boolean b1 = linea1.esMayor(linea1, linea2);

System.out.println("linea1 es mayor que linea2: " + b1);

boolean b2 = linea1.esIgual(linea1, linea2);

System.out.println("linea1 es igual que linea2: " + b2);

// La linea3 es una instancia del tipo Linea. Como el tipo Linea es también un tipo de Relacion

// La variable linea3 puede ser declarada como tipo Relacion. Este concepto es muy importante para entender.

Relacion linea3 = new Linea(1, 5, 1, 5);

boolean b3 = linea3.esIgual(linea1, linea3);

System.out.println("linea1 es igual que linea3: " + b3);

System.out.println("Longitud de linea1 es " + linea1.distancia());

System.out.println("Longitud de linea2 es " + linea2.distancia());

// La siguiente línea de código generará un error de compilación como linea3 está declarada como un tipo de interface Relacion

// y el método distancia() no es uno de los métodos definidos en el interface Relacion.

// System.out.println("Longitud de linea3 es " + linea3.distancia());

}

}

Implementando Interfaces múltiples

package Interface\_Persona;

public interface **InterfacePersona** {

public double calculaRiqueza();

public String getNombre();

}

public interface **OtroInterface** {

public int medirInteligencia(String nombre);

}

public interface **LongitudNombre** {

public int longitudNombre(String nombre);

}

public class **Persona implements InterfacePersona, OtroInterface, LongitudNombre**

{

double dineroEnEfectivo;

double fondoPension;

String nombre;

public Persona(double dineroEnEfectivo, double fondoPension, String nombre) {

this.dineroEnEfectivo = dineroEnEfectivo;

this.fondoPension = fondoPension;

this.nombre = nombre;

}

@Override

public double calculaRiqueza(){

return this.dineroEnEfectivo+this.fondoPension;

}

@Override

public String getNombre() {return this.nombre;}

@Override

public int medirInteligencia(String nombre){

if (nombre.startsWith("Isabel")){

return 140;

}

else{

return 100;

}

}

@Override

public int longitudNombre(String nombre){

return nombre.length();

}

}

public class **Interfaces\_Lab\_2** {

public static void main(String[] args) {

Persona p1=new Persona(1000,2000,"Pepe");

InterfacePersona interfacePersona=p1;

System.out.println("p1 "+p1.getNombre());

System.out.println("interfacePersona "+interfacePersona.getNombre());

System.out.println("riqueza de p1 "+p1.calculaRiqueza());

System.out.println("riqueza de interfacePersona "+interfacePersona.calculaRiqueza());

System.out.println("inteligencia de p1 "+p1.medirInteligencia("Isabel"));

boolean iguales=(p1==interfacePersona);

System.out.println("¿p1 e interfacePersona apuntan al mismo objeto? "+iguales);

InterfacePersona in2=new Persona(3000,5000,"Margarita");

System.out.println("in2 "+in2.getNombre());

System.out.println("riqueza de in2 "+in2.calculaRiqueza());

// Error de compilación que ocurrirá en la siguiente línea de código

// interfacePersona.medirInteligencia(interfacePersona.getNombre());

// Se puede asignar el objeto p1 al tipo OtroInterface

OtroInterface otroInterface = p1;

// Comprobamos si el objeto instanciado referenciado por interfacePersona

// y el referenciado por otroInterface es el mismo. (p1)

boolean b1 = (interfacePersona == otroInterface);

System.out.println("¿interfacePersona y otroInterface apuntan al mismo objeto instanciado? " + b1);

System.out.println("La longitud del nombre de p1 es "+p1.longitudNombre(p1.getNombre()));

}

}

Ejemplo de Herencia y Polimorfismo

package paquete\_Productos;

public interface **Interface\_Producto** {

public double getPrecio();

public void setPrecio(double precio);

public double precioVenta();

}

public **abstract class Producto implements Interface\_Producto** {

private double precio;

public Producto(double precio) {

this.precio = precio;

}

@Override

public double getPrecio() {

return precio;

}

@Override

public void setPrecio(double precio) {

this.precio = precio;

}

@Override

public String toString() {

return "Producto{" + "precio=" + precio + '}';

}

@Override

public double precioVenta(){

return 0;

}

}

public class **Book extends Producto** {

private String editorial;

private int anioEdicion;

public Book(String editorial, int anioEdicion, double precio) {

super(precio);

this.editorial = editorial;

this.anioEdicion = anioEdicion;

}

public String getEditorial() {

return editorial;

}

public int getAnioEdicion() {

return anioEdicion;

}

public void setEditorial(String editorial) {

this.editorial = editorial;

}

public void setAnioEdicion(int anioEdicion) {

this.anioEdicion = anioEdicion;

}

@Override

public String toString() {

return "Book{" +super.toString()+ "editorial=" + editorial + ", anioEdicion=" + anioEdicion + '}';

}

@Override

public double precioVenta() {

return super.getPrecio()\*0.5;

}

}

public interface **Interface\_Electronics extends Interface\_Producto** {

public String getFabricante();

}

public **abstract class Electronics extends Producto implements Interface\_Electronics**{

private String fabricante;

public Electronics(String fabricante, double precio) {

super(precio);

this.fabricante = fabricante;

}

@Override

public String getFabricante() {

return fabricante;

}

public void setFabricante(String fabricante) {

this.fabricante = fabricante;

}

@Override

public String toString() {

return "Electronics{" + super.toString()+ "fabricante=" + fabricante + '}';

}

}

public class **MP3Player extends Electronics**{

private String color;

public MP3Player(String color, String fabricante, double precio) {

super(fabricante, precio);

this.color = color;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

@Override

public String toString() {

return "MP3Player{" +super.toString()+ "color=" + color + '}';

}

@Override

public double precioVenta(){

return super.getPrecio()\*0.9;

}

}

public class **TV extends Electronics**{

private int tamanio;

public TV(int tamanio, String fabricante, double precio) {

super(fabricante, precio);

this.tamanio = tamanio;

}

public int getTamanio() {

return tamanio;

}

public void setTamanio(int tamanio) {

this.tamanio = tamanio;

}

@Override

public String toString() {

return "TV{" + super.toString()+ "tamanio=" + tamanio + '}';

}

@Override

public double precioVenta(){

return super.getPrecio()\*0.8;

}

}

public **class Ppal** {

public static void main(String[] args) {

Producto pa[]=new Producto[5];

pa[0]=new TV(30,"Samsung",1000);

pa[1]=new TV(50,"Sony",2000);

pa[2]=new MP3Player("Apple","Blue",250);

pa[3]=new Book("Sun press",1992,34);

pa[4]=new Book("Korea press",1986,15);

double totalPrecio=0.0;

double totalPrecioVenta=0.0;

for (int i=0; i<pa.length; i++){

// Llama al método getPrecio de la super clase para obtener el precio

totalPrecio += pa[i].getPrecio();

// El precio de venta es diferente dependiendo del tipo de producto

// utilizará el método sobreescrito -overriding-

// del objeto perteneciente a la subclase llamada.

// Es una rutina polimórfica

totalPrecioVenta += pa[i].precioVenta();

System.out.println("Elemento número " + i +

": Tipo = " + pa[i].getClass().getName() +

", Precio = " + pa[i].getPrecio() +

", Precio de Venta = " + pa[i].precioVenta());

}

System.out.println("total Precio = " + totalPrecio);

System.out.println("total Precio Venta = " + totalPrecioVenta);

}

}

Herencia entre Interfaces

package herencia\_entre\_interfaces;

public interface **InterfacePersona** {

public double calculaRiqueza();

public String getNombre();

}

public interface **InterfaceEstudiante extends InterfacePersona** {

public String estudiaEnIES();

}

public class **Estudiante implements InterfaceEstudiante** {

double dineroEnEfectivo;

double fondoPension;

String nombre;

String IES;

public Estudiante(double dineroEnEfectivo, double fondoPension, String nombre, String IES) {

this.dineroEnEfectivo = dineroEnEfectivo;

this.fondoPension = fondoPension;

this.nombre = nombre;

this.IES = IES;

}

@Override

public double calculaRiqueza(){

return this.dineroEnEfectivo + this.fondoPension;

}

@Override

public String getNombre(){

return this.nombre;

}

@Override

public String estudiaEnIES(){

return IES;

}

@Override

public String toString() {

return "Estudiante{" + "dineroEnEfectivo=" + dineroEnEfectivo + ", fondoPension=" + fondoPension + ", nombre=" + nombre + ", IES=" + IES + '}';

}

}

public class **Herencia\_Entre\_Interfaces** {

public static void main(String[] args) {

// Crea una instancia de objeto de la clase Estudiante

Estudiante student1 = new Estudiante(10000, 20000, "Shin Chang","IES Jose Planes");

// Se puede asignar a una interfaz un objeto de la clase Estudiante.

InterfaceEstudiante in1 = student1;

System.out.println("Nombre del estudiante = " + student1.getNombre() + "," +

" riqueza del estudiante = " + student1.calculaRiqueza()+ "," +

" IES del estudiante = " + student1.estudiaEnIES());

System.out.println("Datos de in1 = " + in1.toString());

}

}

package reescribir\_interface;

public interface **InterfacePersona** {

public double calculaRiqueza();

public String getNombre();

}

public class **Persona implements InterfacePersona**

{

double dineroEnEfectivo;

double fondoPension;

String nombre;

public Persona(double dineroEnEfectivo, double fondoPension, String nombre) {

this.dineroEnEfectivo = dineroEnEfectivo;

this.fondoPension = fondoPension;

this.nombre = nombre;

}

@Override

public double calculaRiqueza(){

return this.dineroEnEfectivo+this.fondoPension;

}

@Override

public String getNombre() {return this.nombre;}

}

public interface **OtroInterface** {

public void nuevoMetodo();

}

public class **OtraPersona implements InterfacePersona,OtroInterface**

{

double dineroEnEfectivo;

double fondoPension;

String nombre;

public OtraPersona(double dineroEnEfectivo, double fondoPension, String nombre) {

this.dineroEnEfectivo = dineroEnEfectivo;

this.fondoPension = fondoPension;

this.nombre = nombre;

}

@Override

public double calculaRiqueza(){

return this.dineroEnEfectivo+this.fondoPension;

}

@Override

public String getNombre() {return this.nombre;}

@Override

public void nuevoMetodo(){

System.out.println("Método de la interfaz OtroInterface");

}

}

public class **Mujer extends Persona implements OtroInterface**{

public Mujer(double dineroEnEfectivo, double fondoPension, String nombre) {

super(dineroEnEfectivo, fondoPension, nombre);

}

@Override

public void nuevoMetodo(){

System.out.println("Método nuevo de OtroInterface en Mujer");

}

}

public class **Reescribir\_Interface** {

public static void main(String[] args) {

Persona p=new Persona(100,1000,"Pepe");

System.out.println("Riqueza de p "+p.calculaRiqueza());

System.out.println("Nombre de p "+p.getNombre());

InterfacePersona intP=p;

System.out.println("Riqueza de intP "+ intP.calculaRiqueza());

System.out.println("Nombre de intP "+ p.getNombre());

/\*----------------------------------------------------\*/

OtraPersona op=new OtraPersona(8,900,"Angel");

System.out.println("Riqueza de op "+op.calculaRiqueza());

System.out.println("Nombre de op "+op.getNombre());

op.nuevoMetodo();

//solo puede acceder a los métodos calculaRiqueza y getNombre

InterfacePersona intOp=op;

System.out.println("Riqueza de intOp "+ intP.calculaRiqueza());

System.out.println("Nombre de intOp "+ p.getNombre());

//solo puede acceder al método nuevoMetodo

OtroInterface otherIntOp=op;

otherIntOp.nuevoMetodo();

/\*------------------------------------------------------\*/

Mujer mujer=new Mujer(200,2000,"Laura");

//Los métodos calculaRiqueza los hereda de Persona que a su vez implementa la interface InterfacePersona

System.out.println("Riqueza de mujer "+mujer.calculaRiqueza());

System.out.println("Nombre de mujer "+mujer.getNombre());

// La subclase Mujer implementa el método nuevoMetodo de OtroInterface

mujer.nuevoMetodo();

//solo puede acceder a los métodos calculaRiqueza y getNombre, heredados de Persona que implementa la interface InterfacePersona

InterfacePersona intMujer=mujer;

System.out.println("Riqueza de intMujer "+ intMujer.calculaRiqueza());

System.out.println("Nombre de intMujer "+ intMujer.getNombre());

//solo puede acceder al método nuevoMetodo

OtroInterface otherIntMujer=mujer;

otherIntMujer.nuevoMetodo();

}}