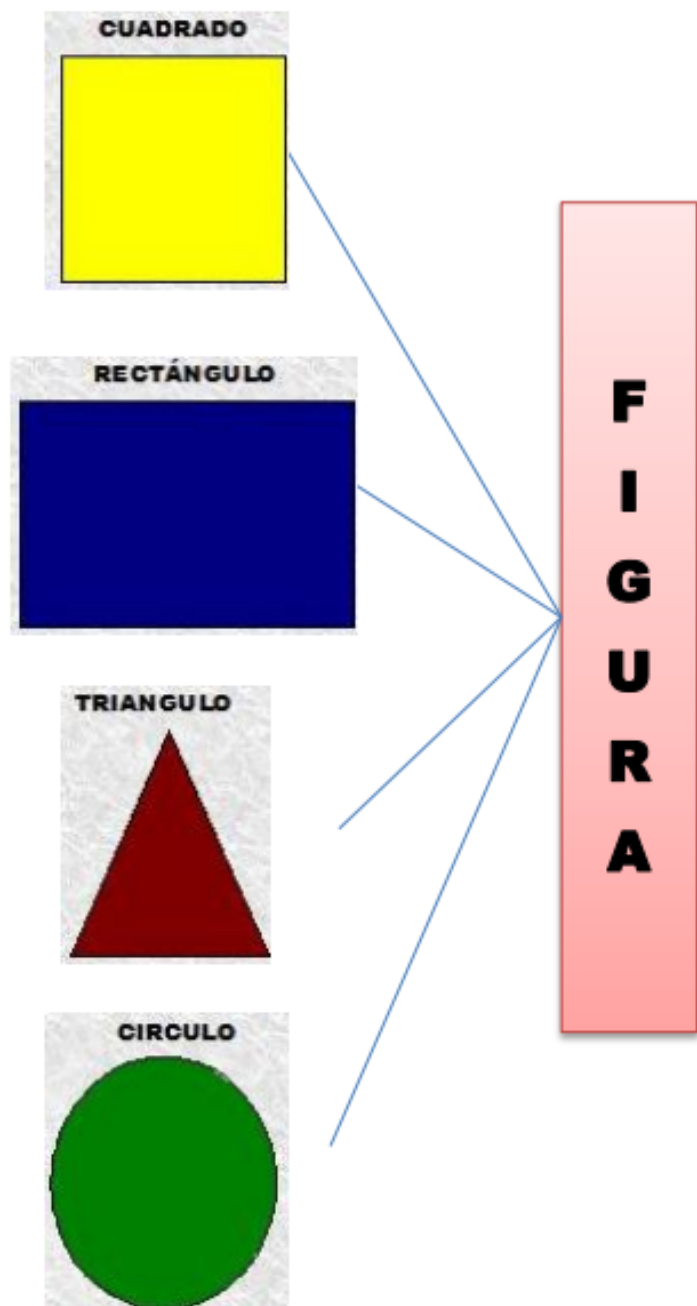


Polimorfismo e Interfaces

El polimorfismo ocurre cuando un programa invoca a un método a través de una variable de la superclase; en tiempo de ejecución, se hace una llamada a la versión correcta del método de la subclase, con base en el tipo de la referencia almacenada en la variable de la superclase

Con el polimorfismo podemos usar el mismo nombre y la misma firma del método para hacer que ocurran distintas acciones, dependiendo del tipo del objeto en el que se invoca el método. Esto proporciona al programador una enorme capacidad expresiva.

EJEMPLOS DE POLIMORFISMO



ArrayList

FIGURA	Método Area ()
Cuadrado(objeto)	$L * L$
Rectángulo (objeto)	$B * A$
Triangulo(objeto)	$(B * A) / 2$
Circulo (objeto)	$Pi * R * R$

El mismo
mensaje tiene
muchas formas

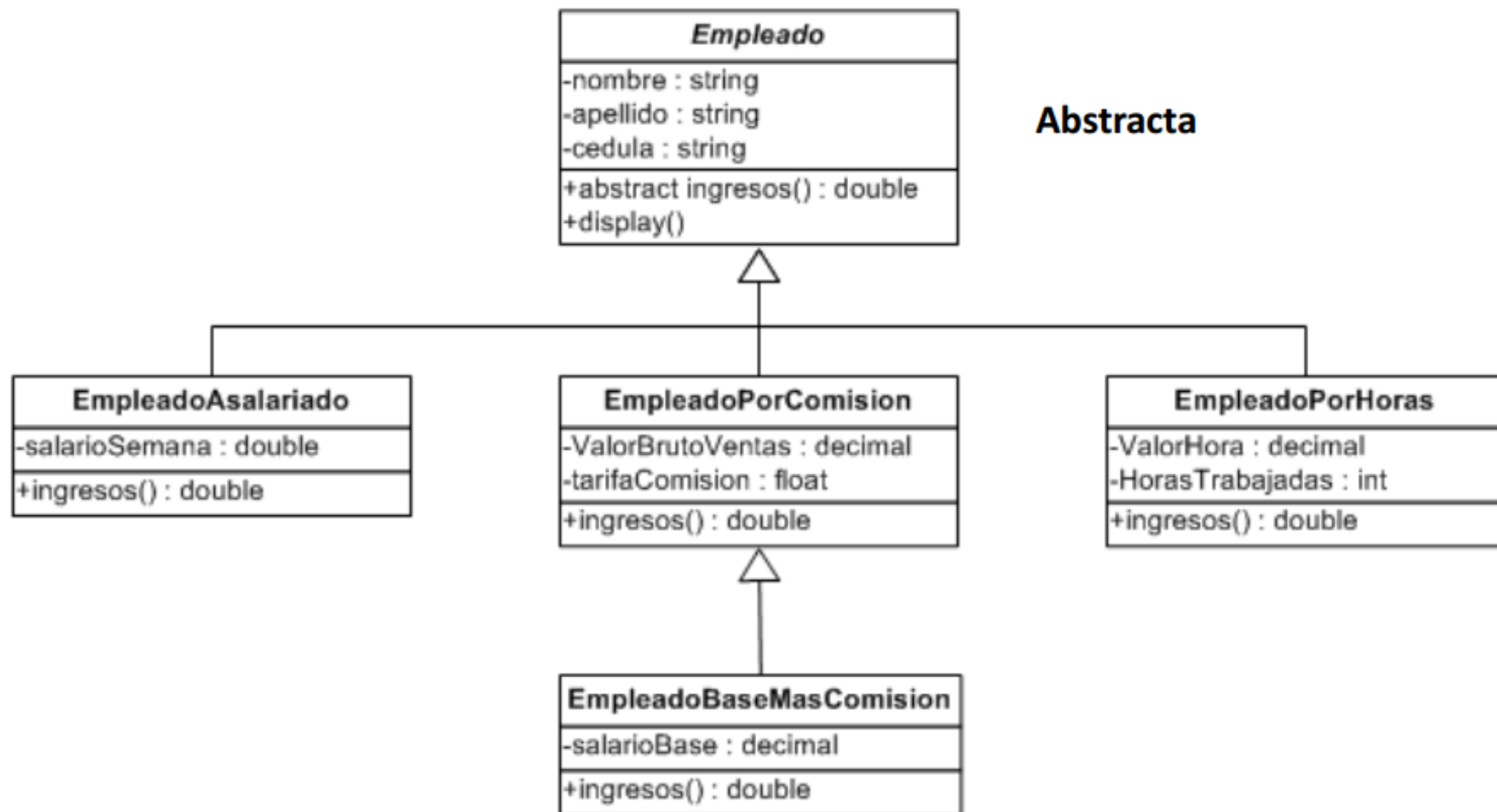
Cada objeto hace
lo correcto en la
llamada al mismo
método

Se invoca el
método a través
de una variable
de la superclase

El mismo nombre y
la misma firma del
método para hacer
distintas acciones

POLIMORFISMO – CASO DE ESTUDIO

Abstracta



CASO DE ESTUDIO - Clase Abstracta Empleado

```
1  /
2  * Creado Por Jairo Seoanes
3  *
4  */
5  package ejemplopolimorfismo;
6
7  public abstract class Empleado {
8      private String nombre;
9      private String apellido;
10     private String cedula;
11
12     public Empleado(String nombre, String apellido, String cedula) {
13         this.nombre = nombre;
14         this.apellido = apellido;
15         this.cedula = cedula;
16     }
17
18     public String getNombre() { return nombre; }
19     public void setNombre(String nombre) { this.nombre = nombre; }
20     public String getApellido() { return apellido; }
21     public void setApellido(String apellido) { this.apellido = apellido; }
22     public String getCedula() { return cedula; }
23     public void setCedula(String cedula) { this.cedula = cedula; }
24
25     public void display() {
26         System.out.println("Cedula: " + this.getCedula() + " - Nombre: " + this.getNombre()
27             + " - Apellido: " + this.getApellido());
28     }
29
30     public abstract double ingresos();
31 }
```


CASO DE ESTUDIO - Subclase EmpleadoAsalariado

```
package ejemplopolimorfismo;

public class EmpleadoAsalariado extends Empleado {
    private double ssalarioSemanal;

    public EmpleadoAsalariado(double ssalarioSemanal, String nombre, String apellido, String cedula) {
        super(nombre, apellido, cedula);
        this.ssalarioSemanal = ssalarioSemanal;
    }

    public double getSsalarioSemanal() { return ssalarioSemanal; }

    public void setSsalarioSemanal(double ssalarioSemanal) { this.ssalarioSemanal = ssalarioSemanal;}

    @Override
    public double ingresos(){
        return this.getSsalarioSemanal() * 4;
    }

    @Override
    public void display(){
        super.display();
        System.out.println("Salario Semanal: " + this.getSsalarioSemanal());
    }
}
```

CASO DE ESTUDIO - Subclase EmpleadoPorHoras

```
package ejemploPolimorfismo;

public class EmpleadoPorHoras extends Empleado {
    private double valorHora;
    private int horasTrabajadas;

    public EmpleadoPorHoras(double valorHora, int horasTrabajadas, String nombre, String apellido, String cedula) {
        super(nombre, apellido, cedula);
        this.valorHora = valorHora;
        this.horasTrabajadas = horasTrabajadas;
    }

    public double getValorHora() { return valorHora; }
    public void setValorHora(double valorHora) { this.valorHora = valorHora; }
    public int getHorasTrabajadas() { return horasTrabajadas; }
    public void setHorasTrabajadas(int horasTrabajadas) { this.horasTrabajadas = horasTrabajadas; }

    @Override
    public double ingresos() {
        if (this.getHorasTrabajadas() <= 40 ) {
            return this.getHorasTrabajadas() * this.getValorHora();
        }
        else {
            return (this.horasTrabajadas - 40 ) * (1.5) * this.getValorHora() + (40 * this.getValorHora());
        }
    }

    @Override
    public void display() {
        super.display();
        System.out.println("Horas Trabajadas : " + this.getHorasTrabajadas() + " - Valor de Hora: " + this.getValorHora());
    }
}
```

CASO DE ESTUDIO - Subclase EmpleadoPorComision

```
package ejemploPolimorfismo;

public class EmpleadoPorComision extends Empleado{
    private double ventasBrutas;
    private float valorComision;

    public EmpleadoPorComision(double ventasBrutas, float valorComision, String nombre, String apellido, String cedula) {
        super(nombre, apellido, cedula);
        this.ventasBrutas = ventasBrutas;
        this.valorComision = valorComision;
    }

    public double getVentasBrutas() { return ventasBrutas; }
    public void setVentasBrutas(double ventasBrutas) { this.ventasBrutas = ventasBrutas; }
    public float getValorComision() { return valorComision; }
    public void setValorComision(float valorComision) { this.valorComision = valorComision; }

    @Override
    public double ingresos(){
        return this.getVentasBrutas() + this.getVentasBrutas() * this.getValorComision()/100;
    }

    public void display(){
        super.display();
        System.out.println("Horas ventas brutas : " + this.getVentasBrutas()
            + " - Valor de Comision: " + this.getValorComision());
    }
}
```


CASO DE ESTUDIO - Subclase EmpleadoBaseMasComision

```
package ejemplopolimorfismo;

public class EmpleadoBaseMasComision extends EmpleadoPorComision {
    private double salarioBase;

    public EmpleadoBaseMasComision(double salarioBase, double ventasBrutas, float valorComision,
                                   String nombre, String apellido, String cedula) {
        super(ventasBrutas, valorComision, nombre, apellido, cedula);
        this.salarioBase = salarioBase;
    }

    public double getSalarioBase() {
        return salarioBase;
    }

    public void setSalarioBase(double salarioBase) {
        this.salarioBase = salarioBase;
    }

    @Override
    public double ingresos() {
        return super.ingresos() + this.getSalarioBase();
    }

    @Override
    public void display() {
        super.display();
        System.out.println("Salario Base : " + this.getSalarioBase());
    }
}
```

CASO DE ESTUDIO - Clase Principal

```
package ejemplopolimorfismo;

import java.util.ArrayList;

public class EjemploPolimorfismo {

    public static void main(String[] args) {
        // TODO code application logic here
        EmpleadoAsalariado empleadoAsalariado = new EmpleadoAsalariado(500.00, "John", "Smith", "111-11-1111" );
        EmpleadoPorHoras empleadoPorHoras = new EmpleadoPorHoras(100, 42, "Karen", "Price", "222-22-2222" );
        EmpleadoPorComision empleadoPorComision = new EmpleadoPorComision(10000, 6, "Sue", "Jones", "333-33-3333" );
        EmpleadoBaseMasComision empleadoBaseMasComision =
            new EmpleadoBaseMasComision(300, 300, 5, "Bob", "Lewis", "444-44-4444");

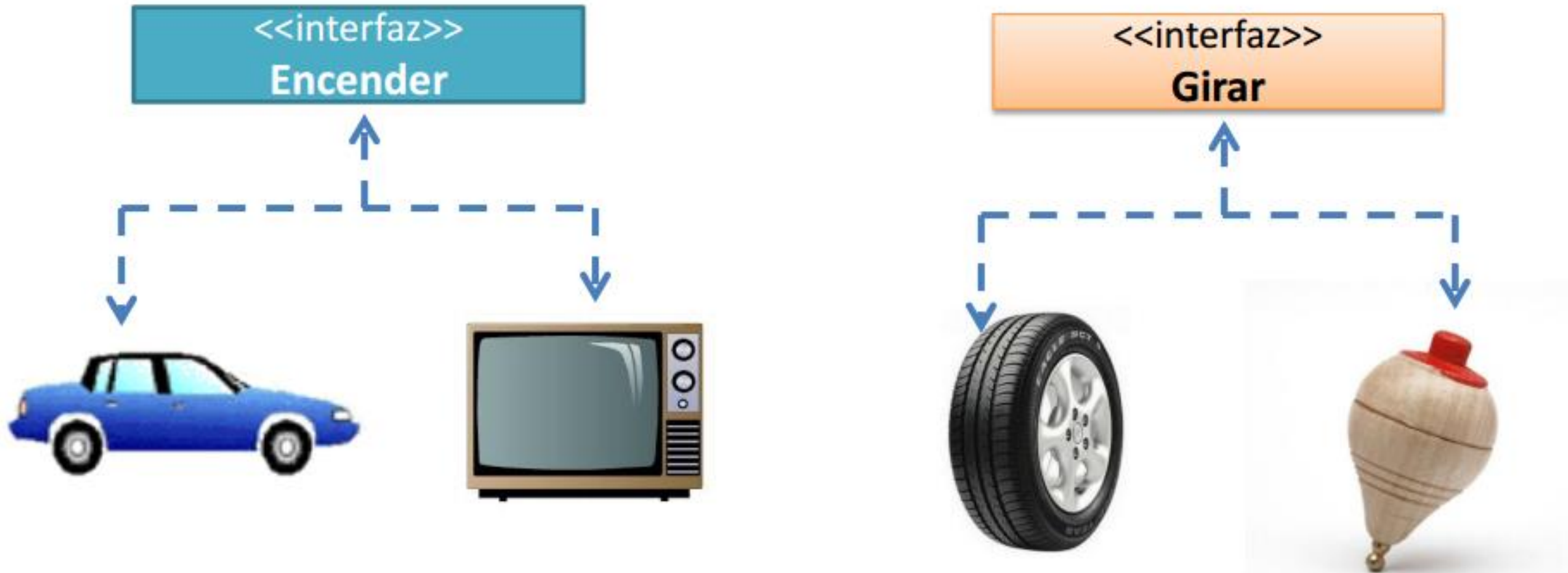
        System.out.println("LLamado del metodo ingresos de forma independiente: ");
        empleadoAsalariado.display();
        System.out.println("Ingresos: "+empleadoAsalariado.ingresos());

        System.out.println("\n LLamado con comportamiento polimorfico: ");
        ArrayList <Empleado> nomina = new ArrayList();
        nomina.add(empleadoAsalariado);
        nomina.add(empleadoPorHoras);
        nomina.add(empleadoPorComision);
        nomina.add(empleadoBaseMasComision);

        for(Empleado e: nomina)
        {
            e.display();
            System.out.println("Ingresos: " + e.ingresos());
            System.out.println("-----\n");
        }
    }
}
```

INTERFACES

Las **interfaces** son una herramienta de java que permite implementar de forma polimórfica un conjunto de métodos comunes para clases no relacionadas .



INTERFACES

Las interfaces definen y estandarizan las formas en que pueden interactuar las cosas entre sí, como las personas y los sistemas.

Por ejemplo, los controles en un radio sirven como una interfaz entre los usuarios del radio y sus componentes internos.

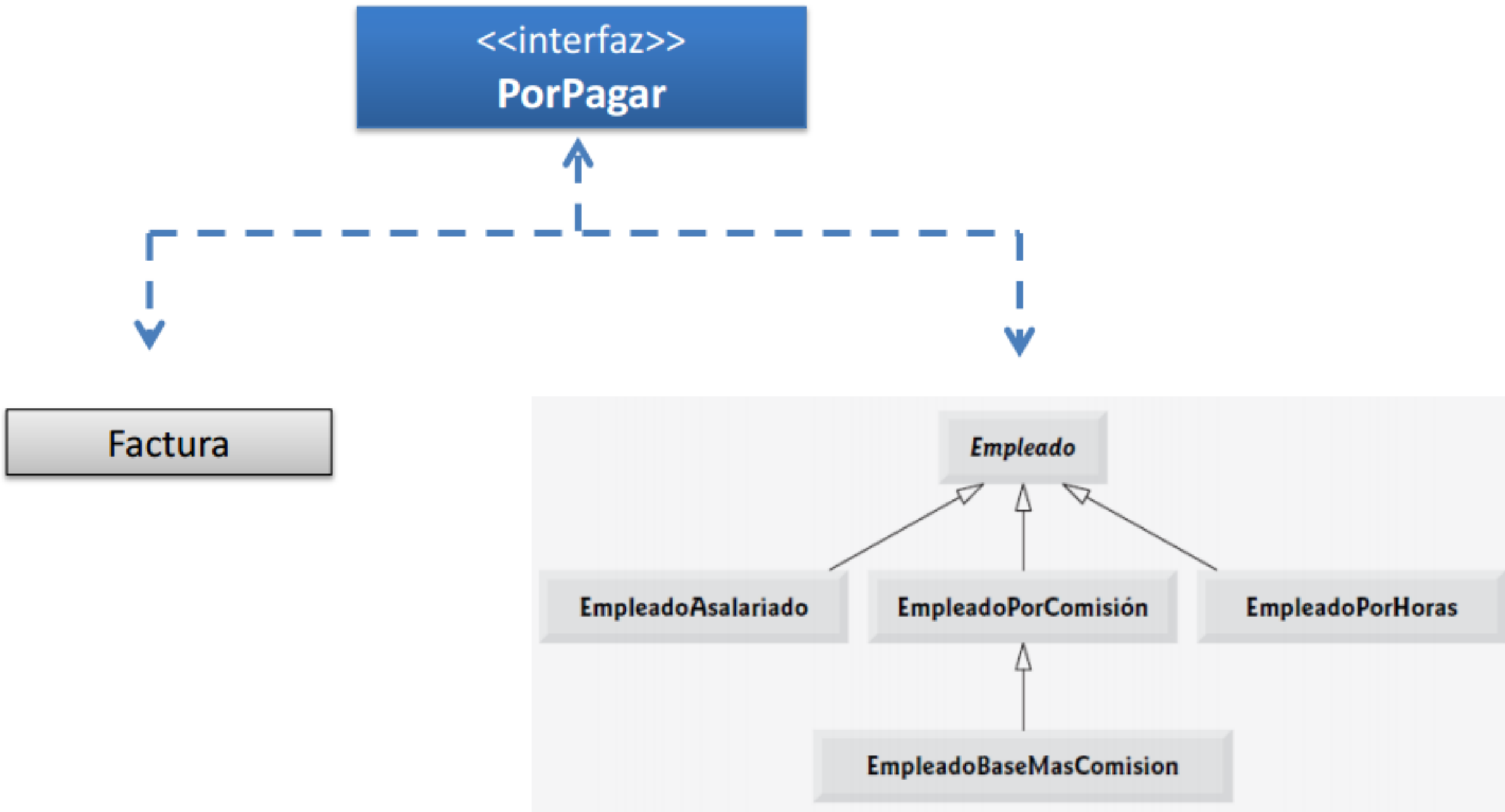
La declaración de una interfaz empieza con la palabra clave **interface** y sólo puede contener constantes y métodos abstract.

Los métodos de una interfaz deben ser **public** y **abstract**, por lo cual no necesitan declararse como tales

A diferencia de las clases, todos los miembros de una interfaz deben ser **public**, y las interfaces no pueden especificar **ningún detalle de implementación**, como las declaraciones de métodos concretos y variables de instancia.

Para utilizar una interfaz, una clase debe especificar que implementa **implements** a esa interfaz y debe declarar cada uno de sus métodos con la firma especificada en la declaración de la interfaz.

Implementar una interfaz es como firmar un contrato con el compilador que diga, “Declararé todos los métodos especificados por la interfaz, o declararé mi clase como **abstract**”.



Creación Interface PorPagar

```
1
2  package ejemplointerfaz;
3
4  /**
5   *
6   * @author JAIRO F
7   */
8  public interface PorPagar {
9
10     public abstract double obtenerMontoPago();
11     public abstract void display();
12     //double obtenerMontoPago();
13
14 }
```

Implementar interfaz en clase Factura


```
3 public class Factura implements PorPagar {
4     private String numeroPieza;
5     private String descripcionPieza;
6     private int cantidad;
7     private double precioArticulo;
8
9     public Factura(String numeroPieza, String descripcionPieza, int cantidad, double precioArticulo) {
10         this.numeroPieza = numeroPieza;
11         this.descripcionPieza = descripcionPieza;
12         this.cantidad = cantidad;
13         this.precioArticulo = precioArticulo;
14     }
15
16     public String getNumeroPieza() {return numeroPieza;}
17     public void setNumeroPieza(String numeroPieza) { this.numeroPieza = numeroPieza;}
18     public String getDescripcionPieza() {return descripcionPieza;}
19     public void setDescripcionPieza(String descripcionPieza) {
20         this.descripcionPieza = descripcionPieza;
21     }
22     public int getCantidad() {return cantidad;}
23     public void setCantidad(int cantidad) { this.cantidad = cantidad;}
24     public double getPrecioArticulo() {return precioArticulo;}
25     public void setPrecioArticulo(double precioArticulo) { this.precioArticulo = precioArticulo; }
26     public double calcularValor(){ return this.getCantidad()*this.getPrecioArticulo(); }
27     @Override
28     public void display(){
29         System.out.println(""+this.getNumeroPieza()+"-" +this.getDescripcionPieza()+"-" +
30             this.getCantidad()+"-" +this.getPrecioArticulo()+"-" +this.calcularValor());
31     }
32     // Metodo implementado de la interfaz
33     @Override
34     public double obtenerMontoPago() {return this.calcularValor();}
```


Implementar interfaz en clase Empleado


```
5 package ejemplointerfaz;
6
7
8 ① public abstract class Empleado implements PorPagar{
9     private String nombre;
10    private String apellido;
11    private String cedula;
12
13    public Empleado(String nombre, String apellido, String cedula) {
14        this.nombre = nombre;
15        this.apellido = apellido;
16        this.cedula = cedula;
17    }
18
19    public String getNombre() { return nombre;}
20    public void setNombre(String nombre) { this.nombre = nombre; }
21    public String getApellido() { return apellido; }
22    public void setApellido(String apellido) { this.apellido = apellido; }
23    public String getCedula() { return cedula; }
24    public void setCedula(String cedula) { this.cedula = cedula; }
25
26    @Override
27    public void display(){
28        System.out.println("Cedula: " + this.getCedula() + " - Nombre: " + this.getNombre()
29            + " - Apellido: " + this.getApellido());
30    }
31
32    public abstract double ingresos();
33 }
```

Implementar Método ObtenerMontoPago() en clases derivadas


Asalariado

```
27  @Override  
    public double obtenerMontoPago() { return this.getSsalarioSemanal(); }
```

EmpleadoPorComision

```
33  @Override  
    public double obtenerMontoPago() { return this.ingresos(); }
```

EmpleadoBasePorComision

```
37  @Override  
    public double obtenerMontoPago() { return this.ingresos(); }
```

EmpleadoPorHora

```
37  @Override  
    public double obtenerMontoPago() { return this.ingresos(); }
```


Implementar Método ObtenerMontoPago() en clases derivadas

```
2 package ejemplointerfaz;
3 import java.util.ArrayList;
4
5 public class EjemploInterfaz {
6
7     public static void main(String[] args) {
8         // TODO code application logic here
9         // Empleado emp = new Empleado("jairo","seoanes","77097");
10        EmpleadoAsalariado empleadoAsalariado = new EmpleadoAsalariado(500.00, "John", "Smith", "111-11-1111" );
11        EmpleadoPorHoras empleadoPorHoras = new EmpleadoPorHoras(100, 42, "Karen", "Price", "222-22-2222" );
12        EmpleadoPorComision empleadoPorComision =new EmpleadoPorComision(10000,6,"Sue", "Jones", "333-33-3333" );
13        EmpleadoBaseMasComision empleadoBaseMasComision =
14            new EmpleadoBaseMasComision(300, 300, 5, "Bob", "Lewis", "444-44-4444");
15
16        Factura factural = new Factura("123","Monitor L.C.D",5,120000);
17
18        System.out.println("\n Uso de la inferfaz PorPagar: ");
19        ArrayList <PorPagar> CuentasPorPagar = new ArrayList();
20
21        CuentasPorPagar.add(factural);
22        CuentasPorPagar.add(empleadoAsalariado);
23        CuentasPorPagar.add(empleadoPorHoras);
24        CuentasPorPagar.add(empleadoPorComision);
25        CuentasPorPagar.add(empleadoBaseMasComision);
26
27        for(PorPagar p: CuentasPorPagar)
28        {
29            p.display();
30            System.out.println("Por Pagar: " + p.obtenerMontoPago());
31            System.out.println("-----\n");
32        }
33    }
```

