

# Ejercicios Java 5

## 41. Clase básica Persona

```
public class Persona {  
  
    String nombre;  
    int edad;  
  
    public Persona(String nombre, int edad) { //constructor  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
  
    public void mostrarDatos() { //muestra los datos  
  
        System.out.println("Nombre: " + nombre);  
        System.out.println("Edad: " + edad);  
    }  
}
```

## 42. Uso de Getters y Setters

```
public class Persona {  
  
    String nombre;  
    int edad;  
  
    public Persona(String nombre, int edad) { //constructor  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
  
    public void mostrarDatos() { //muestra los datos  
  
        System.out.println("Nombre: " + nombre);  
        System.out.println("Edad: " + edad);  
    }  
  
    void setNombre(String nombre) { //Setter nombre  
        this.nombre = nombre;  
    }  
  
    public String getNombre() { //Getter nombre  
        return nombre;  
    }  
  
    void setEdad(int edad) { //Setter edad  
        this.edad = edad;  
    }  
  
    public int getEdad() { //Getter edad  
        return edad;  
    }  
}
```

```
}  
}
```

## 43. Clase Estudiante que hereda Persona

### • Persona.java

```
public class Persona {  
  
    String nombre;  
    int edad;  
  
    public Persona(String nombre, int edad) { //constructor  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
  
    public void mostrarDatos() { //muestra los datos  
  
        System.out.println("Nombre: " + nombre);  
        System.out.println("Edad: " + edad);  
    }  
  
    void setNombre(String nombre) { //Setter nombre  
        this.nombre = nombre;  
    }  
  
    public String getNombre() { //Getter nombre  
        return nombre;  
    }  
  
    void setEdad(int edad) { //Setter edad  
        this.edad = edad;  
    }  
  
    public int getEdad() { //Getter edad  
        return edad;  
    }  
}
```

### • Estudiante.java

```
public class Estudiante extends Persona {  
  
    String curso;  
  
    public Estudiante(String nombre, int edad, String curso) {  
        //constructor  
  
        super(nombre, edad);  
        this.curso = curso;  
    }  
  
    @Override  
    public void mostrarDatos() { //muestra los datos
```

```

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Curso: " + curso);
    }
}

```

## 44. Polimorfismo con mostrarDatos()

### • Persona.java

```

public class Persona {

    String nombre;
    int edad;

    public Persona(String nombre, int edad) { //constructor
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }

    void setNombre(String nombre) { //Setter nombre
        this.nombre = nombre;
    }

    public String getNombre() { //Getter nombre
        return nombre;
    }

    void setEdad(int edad) { //Setter edad
        this.edad = edad;
    }

    public int getEdad() { //Getter edad
        return edad;
    }
}

```

### • Estudiante.java

```

public class Estudiante extends Persona {

    String curso;

    public Estudiante(String nombre, int edad, String curso) {
        //constructor

        super(nombre, edad);
        this.curso = curso;
    }
}

```

```

    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Curso: " + curso);
    }
}

```

## • Principal.java

```

public class Principal {

    public static void main(String[] args) {

        Persona p;
        Persona persona = new Persona("Juan", 20);
        Estudiante estudiante = new Estudiante("Alberto", 14,
"segundo");

        p = persona;
        p.mostrarDatos(); //Muestra datos de la persona

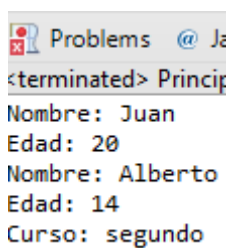
        p = estudiante;
        p.mostrarDatos(); //Muestra datos del estudiante

    }

}

```

## • Resultado



```

Problems @ Ja
<terminated> Princip
Nombre: Juan
Edad: 20
Nombre: Alberto
Edad: 14
Curso: segundo

```

## 45. Uso de super en el constructor

```

public class Estudiante extends Persona {

    String curso;

    public Estudiante(String nombre, int edad, String curso) {
//constructor

```

```

        super(nombre, edad);
        this.curso = curso;
        System.out.println("Mensaje");
    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Curso: " + curso);
    }
}

```

## 46. Clase Animal y subclases Perro y Gato

### • Animal.java

```

public class Animal {

    public void hacerSonido() { //Emite sonido

        System.out.println("Sonido genérico");
    }
}

```

### • Gato.java

```

public class Gato extends Animal {

    @Override
    public void hacerSonido() { //Emite sonido

        System.out.println("Miau miau");
    }
}

```

### • Perro.java

```

public class Perro extends Animal{

    @Override
    public void hacerSonido() { //Emite sonido

        System.out.println("Guau guau");
    }
}

```

## 47. Uso de Super.hacerSonido()

### • Animal.java

```
public class Animal {  
  
    public void hacerSonido() { //Emite sonido  
  
        System.out.println("Sonido genérico");  
    }  
}
```

### • Gato.java

```
public class Gato extends Animal {  
  
    @Override  
    public void hacerSonido() { //Emite sonido  
  
        super.hacerSonido();  
        System.out.println("Miau miau");  
    }  
}
```

### • Perro.java

```
public class Perro extends Animal{  
  
    @Override  
    public void hacerSonido() { //Emite sonido  
  
        super.hacerSonido();  
        System.out.println("Guau guau");  
    }  
}
```

## 48. Constructor con this

```
public class Libro {  
  
    String titulo;  
    String autor;  
  
    public Libro(String titulo, String autor) { //Constructor  
  
        this.titulo = titulo;  
        this.autor = autor;  
    }  
}
```

## 50. Clase abstracta Figura

```
public abstract class Figura {  
  
    abstract double calcularArea(); //Calcula el area de la figura  
  
    public void mostrarTipo() { //Muestra el tipo de la figura  
  
        System.out.println("Soy una figura");  
    }  
  
}
```

## 51. Subclase Circulo que hereda de figura

### • Figura.java

```
public abstract class Figura {  
  
    abstract double calcularArea(); //Calcula el area de la figura  
  
    public void mostrarTipo() { //Muestra el tipo de la figura  
  
        System.out.println("Soy una figura");  
    }  
  
}
```

### • Circulo.java

```
public class Circulo extends Figura {  
  
    double radio;  
  
    public Circulo(double radio) {  
        this.radio = radio;  
    }  
  
    @Override  
    double calcularArea() { //Calcula el área del círculo  
        return Math.PI * radio * radio;  
    }  
  
}
```

## 52. Subclase Rectangulo que hereda de Figura

### • Figura.java

```
public abstract class Figura {  
  
    abstract double calcularArea(); //Calcula el area de la figura  
  
    public void mostrarTipo() { //Muestra el tipo de la figura
```

```

        System.out.println("Soy una figura");
    }
}

```

### • Rectangulo.java

```

public class Rectangulo extends Figura {

    double base;
    double altura;

    public Rectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

    @Override
    double calcularArea() { //Calcula el área del rectángulo
        return base * altura;
    }
}

```

## 53. Uso de clases abstractas

### • Figura.java

```

public abstract class Figura {

    abstract double calcularArea(); //Calcula el area de la figura

    public void mostrarTipo() { //Muestra el tipo de la figura

        System.out.println("Soy una figura");
    }
}

```

### • Circulo.java

```

public class Circulo extends Figura {

    double radio;

    public Circulo(double radio) {
        this.radio = radio;
    }

    @Override
    double calcularArea() { //Calcula el área del círculo
        return Math.PI * radio * radio;
    }
}

```

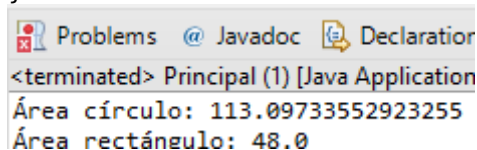


## • Rectangulo.java

```
public class Rectangulo extends Figura {  
  
    double base;  
    double altura;  
  
    public Rectangulo(double base, double altura) {  
        this.base = base;  
        this.altura = altura;  
    }  
  
    @Override  
    double calcularArea() { //Calcula el área del rectángulo  
        return base * altura;  
    }  
}
```

## • Principal.java

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Figura circulo = new Circulo(6);  
        Figura rectangulo = new Rectangulo(8, 6);  
  
        System.out.println("Área círculo: " + circulo.calcularArea());  
        System.out.println("Área rectángulo: " +  
rectangulo.calcularArea());  
  
    }  
}
```



```
<terminated> Principal (1) [Java Application]  
Área círculo: 113.09733552923255  
Área rectángulo: 48.0
```

## 54. Clase Vehiculo y subclase Coche

### • Vehiculo.java

```
public class Vehiculo {  
  
    String marca;  
    String modelo;  
}
```

## • Cohe.java

```
public class Coche extends Vehiculo{

    int numPuertas;

    public Coche(String marca, String modelo, int numPuertas) {
//Constructor

        this.marca = marca;
        this.modelo = modelo;
        this.numPuertas = numPuertas;
    }

    public String getMarca() { //Devuelve la marca

        return marca;
    }

    public String getModelo() { //Devuelve el modelo

        return modelo;
    }

    public int getPuertas() { //Devuelve el número de puertas

        return numPuertas;
    }
}
```

## • Principal.java

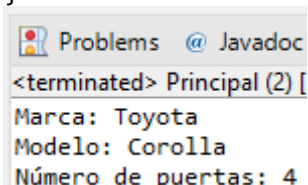
```
public class Principal {

    public static void main(String[] args) {

        Coche c = new Coche("Toyota", "Corolla", 4);

        System.out.println("Marca: " + c.getMarca());
        System.out.println("Modelo: " + c.getModelo());
        System.out.println("Número de puertas: " + c.getPuertas());

    }
}
```



The screenshot shows a 'Problems' window in an IDE. It contains a single entry: '<terminated> Principal (2) ['. Below this entry, the output of the program is displayed: 'Marca: Toyota', 'Modelo: Corolla', and 'Número de puertas: 4'.

## 55. Composición de clases (Libro y autor)

### • Autor

```
public class Autor {  
  
    String nombre;  
    String nacionalidad;  
  
}
```

### • Libro

```
public class Libro {  
  
    Autor autor;  
  
}
```

## 56. Subclase Profesor que hereda de persona

### • Persona

```
public class Persona {  
  
    String nombre;  
    int edad;  
  
    public Persona(String nombre, int edad) { //constructor  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
  
    public void mostrarDatos() { //muestra los datos  
  
        System.out.println("Nombre: " + nombre);  
        System.out.println("Edad: " + edad);  
    }  
  
    void setNombre(String nombre) { //Setter nombre  
        this.nombre = nombre;  
    }  
  
    public String getNombre() { //Getter nombre  
        return nombre;  
    }  
  
    void setEdad(int edad) { //Setter edad  
        this.edad = edad;  
    }  
  
    public int getEdad() { //Getter edad  
        return edad;  
    }  
}
```

## • Profesor

```
public class Profesor extends Persona{

    String asignatura;

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Asignatura: " + asignatura);
    }

}
```

## 57. Constructor de profesor con uso de super y this

### • Persona

```
public class Persona {

    String nombre;
    int edad;

    public Persona(String nombre, int edad) { //constructor
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }

    void setNombre(String nombre) { //Setter nombre
        this.nombre = nombre;
    }

    public String getNombre() { //Getter nombre
        return nombre;
    }

    void setEdad(int edad) { //Setter edad
        this.edad = edad;
    }

    public int getEdad() { //Getter edad
        return edad;
    }

}
```

### • Profesor

```
public class Profesor extends Persona{
```

```

        String asignatura;

        public Profesor(String nombre, int edad, String asignatura) {
//constructor
            super(nombre, edad);
            this.asignatura = asignatura;
        }

        @Override
        public void mostrarDatos() { //muestra los datos

            System.out.println("Nombre: " + nombre);
            System.out.println("Edad: " + edad);
            System.out.println("Asignatura: " + asignatura);
        }
    }
}

```

## 58. Jerarquía de clases con polimorfismo básico

### • Persona

```

public class Persona {

    String nombre;
    int edad;

    public Persona(String nombre, int edad) { //constructor
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }

    void setNombre(String nombre) { //Setter nombre
        this.nombre = nombre;
    }

    public String getNombre() { //Getter nombre
        return nombre;
    }

    void setEdad(int edad) { //Setter edad
        this.edad = edad;
    }

    public int getEdad() { //Getter edad
        return edad;
    }
}

```

### • Profesor

```

public class Profesor extends Persona{

    String asignatura;

    public Profesor(String nombre, int edad, String asignatura) {
//constructor
        super(nombre, edad);
        this.asignatura = asignatura;
    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Asignatura: " + asignatura);
    }

}

```

## • Estudiante

```

public class Estudiante extends Persona {

    String curso;

    public Estudiante(String nombre, int edad, String curso) {
//constructor

        super(nombre, edad);
        this.curso = curso;
    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Curso: " + curso);
    }

}

```

## • Principal

```

public class Principal {

    public static void main(String[] args) {

        Persona estudiante = new Estudiante("Juan", 13, "primero");
        Persona profesor = new Profesor("Manuel", 32, "Lengua");

        estudiante.mostrarDatos();
        profesor.mostrarDatos();
    }

}

```

```
Problems @ Javad
<terminated> Principal (
Nombre: Juan
Edad: 13
Curso: primero
Nombre: Manuel
Edad: 32
Asignatura: Lengua
```

## 59. Uso de instanceof

### • Persona

```
public class Persona {

    String nombre;
    int edad;

    public Persona(String nombre, int edad) { //constructor
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }

    void setNombre(String nombre) { //Setter nombre
        this.nombre = nombre;
    }

    public String getNombre() { //Getter nombre
        return nombre;
    }

    void setEdad(int edad) { //Setter edad
        this.edad = edad;
    }

    public int getEdad() { //Getter edad
        return edad;
    }
}
```

### • Profesor

```
public class Profesor extends Persona{

    String asignatura;

    public Profesor(String nombre, int edad, String asignatura) {
//constructor
        super(nombre, edad);
    }
}
```

```

        this.asignatura = asignatura;
    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Asignatura: " + asignatura);
    }
}

```

## • Estudiante

```

public class Estudiante extends Persona {

    String curso;

    public Estudiante(String nombre, int edad, String curso) {
//constructor

        super(nombre, edad);
        this.curso = curso;
    }

    @Override
    public void mostrarDatos() { //muestra los datos

        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Curso: " + curso);
    }
}

```

## • Principal

```

public class Principal {

    public static void main(String[] args) {

        Persona estudiante = new Estudiante("Juan", 13, "primero");
        Persona profesor = new Profesor("Manuel", 32, "Lengua");

        estudiante.mostrarDatos();

        if (estudiante instanceof Estudiante) { //Muestra el mensaje si es estudiante

            System.out.println("Es un estudiante");
        }

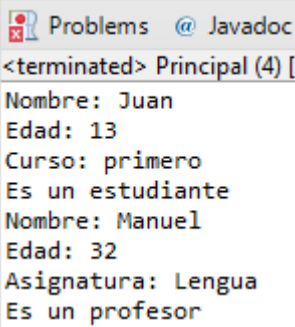
        profesor.mostrarDatos();

        if (profesor instanceof Profesor) { //Muestra el mensaje si es profesor

```



```
        System.out.println("Es un profesor");
    }
}
```



The screenshot shows the 'Problems' window of an IDE. The title bar includes a red 'x' icon, the word 'Problems', and a '@ Javadoc' button. The main content area displays a message: '<terminated> Principal (4) ['. Below this, the following text is listed: 'Nombre: Juan', 'Edad: 13', 'Curso: primero', 'Es un estudiante', 'Nombre: Manuel', 'Edad: 32', 'Asignatura: Lengua', and 'Es un profesor'.

<https://github.com/Rodrigo-Garcia-Ortiz/Programacion.git>