

PROGRAMACIÓN CONCURRENTE

TRABAJO PRÁCTICO N° 1

INTEGRANTES GRUPO 20:

CAMACHO, Mauro Rodrigo – 7920 – 41654090@fi.unju.edu.ar

CESPEDES, Alan Julian – 1980 – 45468716@fi.unju.edu.ar

HIDALGO, Hugo Joaquín – 8018 – 41862135@fi.unju.edu.ar

MENDOZA, Franco Nahuel – 1729 – 41609933@fi.unju.edu.ar

JUÁREZ MOYANO, Mauricio Joaquin – 2006 – 38162310@fi.unju.edu.ar

VALDIVIEZO, Gustavo Marcelo A. – 8077 – 42453953@fi.unju.edu.ar



FACULTAD DE INGENIERÍA

TRABAJO PRÁCTICO N° 1

Fecha Inicio: 14/08/2025

Fecha de Entrega: 28/08/2025

Clase Metodos: [Metodos.java](#)

Proyecto en GitHub: [TP01](#)

TRABAJO PRÁCTICO N° 1..... 1

PUNTO 1..... 2

Clase CuentaBancaria.....	2
Clase Main del Punto 1.....	3
Resultados.....	4

PUNTO 2..... 4

Clase CuentaSueldo.....	4
Clase Main del Punto 2.....	6
Resultados.....	7

PUNTO 4..... 8

Clase Forma.....	8
Clase Cuadrilatero.....	8
Clase Main del Punto 4.....	9
Resultados.....	9

PUNTO 5..... 9

Clase Circulo.....	9
Clase Main del Punto 5.....	10
Resultados.....	10

PUNTO 6..... 10

Interfaz Forma.....	10
Clase Cuadrilatero.....	10
Clase Main del Punto 6.....	11
Resultados.....	11

PUNTO 1

Empleando Java, cree una clase CuentaBancaria con atributos para el número de cuenta (un entero largo), el DNI del cliente (otro entero largo), el saldo actual (double) y el interés anual que se aplica a la cuenta (porcentaje). Defina en la clase los siguientes métodos:

Clase CuentaBancaria

```
package punto1; import metodos.Metodo;
public class CuentaBancaria {
    //long contador = 100001,
    long numCuenta;
    protected long dni;

    // El número de cuenta se asignará de forma correlativa a partir de 100001, asignando el
    // siguiente número al último asignado

    static int contador = 100001;
    protected double saldo;
    protected double interes;

    //Constructor por defecto y constructor con DNI, saldo e interés.

    public CuentaBancaria(){ //Constructor por defecto
        this.numCuenta = this.contador;
        this.dni = 0;
        this.saldo = 0;
        this.interes = 0;
        this.contador = this.contador + 1;
    }
    public CuentaBancaria(long dni, double saldo, double interes){ //constructor con DNI, saldo e
interés.
        this.numCuenta = this.contador;
        this.dni = dni;
        this.saldo = Metodo.redondear(saldo, 2);
        this.interes = interes;
        this.contador = this.contador + 1;
    }

    // actualizarSaldo(): actualizará el saldo de la cuenta aplicándole el interés diario (interés
    // anual dividido entre 365 aplicado al saldo actual).

    public void actulizarSaldo(){
        double calculo;
        calculo = this.saldo * (this.interes/365);
        this.saldo += Metodo.redondear(calculo, 2);
        mostrarCuenta();
    }

    // ingresar(double): permitirá ingresar una cantidad en la cuenta.

    public void ingresar(double monto){ //ingresar dinero al saldo actual
        //double monto = pedir.pedirDouble("> INGRESE EL MONTO A INGRESAR: ");
        this.saldo += Metodo.redondear(monto, 2);
        mostrarCuenta();
    }

    // retirar(double): permitirá sacar una cantidad de la cuenta (si hay saldo).

    public void retirar(double extraer) {
        if (this.saldo > 0){
            //double extraer = pedir.pedirDouble("> INGRESE EL MONTO A EXTRAER: ");
```

PROGRAMACIÓN CONCURRENTE

Tema: Programación Orientada a Objetos

```
extraer = Metodo.redondear(extraer, 2);
if (extraer <= this.saldo) {
    this.saldo = this.saldo - extraer;
    mostrarCuenta();
}
else {System.out.println("- NO ES POSIBLE EXTRAER MAS QUE SU SALDO DE NUM CUENTA:
"+this.numCuenta+"");}
}
else {System.out.println("- LA CUENTA"+this.numCuenta+"NO TIENEN FONDOS PARA EXTRAER.");}
}

// Método que nos permita mostrar todos los datos de la cuenta.

public void mostrarCuenta() {
    System.out.println("> NUM. CUENTA:"+numCuenta);
    System.out.println("- DNI: "+dni);
    System.out.println("- SALDO: "+saldo);
    System.out.println("- INTERES: "+interes+"\n");
}
}//Final clase CuentaBancaria
```

Clase Main del Punto 1

```
package punto1;
import metodos.Metodo;
public class Punto1 {
    public static int menu(){
        int opcion = 0;
        System.out.println("\n\n\t -- MENU PUNTO 1 --");
        System.out.println("\t 1. MOSTRAR CUENTA.");
        System.out.println("\t 2. ACTUALIZAR SALDO.");
        System.out.println("\t 3. INGRESAR .");
        System.out.println("\t 4. RETIRAR.");
        System.out.println("\t 5. SALIR");
        opcion = Metodo.validarOpcion(5);
        return opcion;
    }
    public static void main(String[] arg) {
        long dni = Metodo.pedirLong("> INGRESE DNI: ");
        double sueldo = Metodo.pedirDouble("> INGRESE EL SUELDO: ");
        double interes = Metodo.pedirDouble("> INGRESE EL INTERES: ");
        CuentaBancaria cliente = new CuentaBancaria(dni, sueldo, interes);
        int opcion = 0;
        while(opcion != 5) {
            opcion = menu();
            switch (opcion) {
                case 1: System.out.println("\n - MOSTRAR CUENTA");
                    cliente.mostrarCuenta();
                    break;
                case 2: System.out.println("\n - ACTUALIZAR SUELDO DE CUENTA");
                    cliente.actulizarSaldo();
                    break;
                case 3: System.out.println("\n - INGRESAR MONTO A LA CUENTA");
                    double monto = Metodo.pedirDouble(" > MONTO: ");
                    cliente.ingresar(monto);
                    break;
                case 4: System.out.println("\n - INGRESAR MONTO A RETIRAR DE LA CUENTA");
                    double retiro = Metodo.pedirDouble(" > MONTO: ");
                    cliente.retirar(retiro);
                    break;
            }
        }
        System.out.println("A FINALIZADO EL PROGRAMA.");
    }
}
```

Resultados

<pre>> INGRESE DNI: 4160 > INGRESE EL SUELDO: 1000000 > INGRESE EL INTERES: 5</pre>	<pre>-- MENU PUNTO 1 -- 1. MOSTRAR CUENTA. 2. ACTUALIZAR SALDO. 3. INGRESAR . 4. RETIRAR. 5. SALIR > OPCION: 1 - MOSTRAR CUENTA > NUM. CUENTA:100001 - DNI: 4160 - SALDO: 1000000.0 - INTERES: 5.0</pre>	<pre>-- MENU PUNTO 1 -- 1. MOSTRAR CUENTA. 2. ACTUALIZAR SALDO. 3. INGRESAR . 4. RETIRAR. 5. SALIR > OPCION: 2 - ACTUALIZAR SUELDO DE CUENTA > NUM. CUENTA:100001 - DNI: 4160 - SALDO: 1013698.63 - INTERES: 5.0</pre>
<pre>-- MENU PUNTO 1 -- 1. MOSTRAR CUENTA. 2. ACTUALIZAR SALDO. 3. INGRESAR . 4. RETIRAR. 5. SALIR >OPCION: 3 - INGRESAR MONTO A LA CUENTA > MONTO: 500000 > NUM. CUENTA:100001 - DNI: 4160 - SALDO: 1513698.63 - INTERES: 5.0</pre>	<pre>-- MENU PUNTO 1 -- 1. MOSTRAR CUENTA. 2. ACTUALIZAR SALDO. 3. INGRESAR . 4. RETIRAR. 5. SALIR > OPCION: 4 - INGRESAR MONTO A RETIRAR DE LA CUENTA > MONTO: 1000000 > NUM. CUENTA:100001 - DNI: 4160 - SALDO: 513698.63 - INTERES: 5.0</pre>	<pre>-- MENU PUNTO 1 -- 1. MOSTRAR CUENTA. 2. ACTUALIZAR SALDO. 3. INGRESAR . 4. RETIRAR. 5. SALIR >OPCION: 5 A FINALIZADO EL PROGRAMA.</pre>

PUNTO 2

A partir de la clase definida en el ejemplo anterior, cree una nueva clase CuentaSueldo, que herede de Cuenta y posea los siguientes atributos propios: legajo (entero), institución (string), beneficios (string), CBU (entero largo de 18 dígitos), tope (double). Además debe poseer los siguientes métodos:

Clase CuentaSueldo

```
package punto2;
import punto1.CuentaBancaria;
import metodos.Metodo;
public class CuentaSueldo extends CuentaBancaria{ //HIJO
    int legajo = 0;
    String institucion, beneficios;
    long cbu;
    double tope;

//Constructor por defecto y constructor con CBU (el cual es un número de 18 dígitos) y tope
fijado en 15.000

    public CuentaSueldo(){ //POR DEFECTO
        super();
        this.legajo = 0;
        this.institucion = "Desconocido";
        this.beneficios = "Desconocido";
        this.cbu = 0;
        this.tope = 0;
    }
    public CuentaSueldo(
```

PROGRAMACIÓN CONCURRENTE

Tema: Programación Orientada a Objetos

```
long dni, double saldo, double interes,
int legajo, String institucion, String beneficios, long cbu, double tope) {
    this.dni = dni;
    this.saldo = Metodo.redondear(saldo, 2);
    this.interes = interes;
    this.legajo = legajo;
    this.institucion = institucion;
    this.beneficios = beneficios;

    if(Metodo.validarLong(cbu) == 18){ //MODIFICAR A 18
        this.cbu = cbu;
    }
    else {
        System.out.println(" CORRIGA EL CBU");
    }
    if(tope <= 15000){
        this.tope = tope;
        System.out.println("- CARGA EXITOSA");
    }
    else {
        System.out.println("- CORRIGA EL EL TOPE, DEBE SER MENOR A 15000");
    }
}

public void mostrarSueldo() {
    System.out.println("- DNI: "+dni);
    System.out.println("- LEGAJO: "+legajo);
    System.out.println("- INSTITUCION: "+institucion);
    System.out.println("- BENEFICIO: "+beneficios);
    System.out.println("- SALDO: "+saldo);
    System.out.println("- INTERES: "+interes+"\n");
}

//Sobrescribir el método retirar (double), para que además de permitir sacar una cantidad de la cuenta (si hay saldo), no permita extracciones superiores al tope.

public void retirar(double extraer) {
    if (this.saldo > 0){
        //double extraer = pedir.pedirDouble("> INGRESE EL MONTO A EXTRAER: ");
        extraer = Metodo.redondear(extraer, 2);
        if (extraer <= this.saldo && extraer <= this.tope) {
            this.saldo = this.saldo - extraer;
            mostrarCuenta();
        }
        else {System.out.println(" NO ES POSIBLE EXTRAER. USUARIO: "+this.dni+"");}
    }
    else {System.out.println(" EL USUARIO "+this.dni+" NO TIENEN FONDOS PARA EXTRAER.");}
}

//transferir (monto, CBU): este método simulará una transferencia a otra cuenta por el monto ingresado (siempre y cuando haya saldo) y decrementará el saldo de la cuenta. Para la "transferencia", muestre el saldo final de la cuenta.

public void trasferir(double monto, long cbu) {
    monto = Metodo.redondear(monto, 2);
    if(this.saldo >= monto && Metodo.validarLong(cbu) == 18) { //cambiar por 18
        this.saldo = saldo - monto;
        System.out.println("\n - TRASFERENCIA HECHA DE: $" + monto + " A " + cbu);
        System.out.println(" - SALDO ACTUAL: " + this.saldo);
    }
    else {System.out.println(" * NO ES POSIBLE REALIZAR LA TRANSFERENCIA");}
}
```

PROGRAMACIÓN CONCURRENTE

Tema: Programación Orientada a Objetos

//Sobrecargue el método transferir (monto, Alias) de la clase CuentaSueldo para que acepte un alias alfanumérico.

```
public void trasferir(double monto, String alias) {  
    monto = Metodo.redondear(monto, 2);  
    if(this.saldo >= monto) {  
        this.saldo = saldo - monto;  
        System.out.println("\n- TRASFERENCIA HECHA DE: $" + monto + " A " + alias);  
        System.out.println("- SALDO ACTUAL: " + this.saldo);  
    }  
    else {System.out.println(" NO ES POSIBLE REALIZAR LA TRANSFERENCIA");}  
}  
}//CuentaSueldo
```

Clase Main del Punto 2

```
package punto2;  
import metodos.Metodo;  
public class Punto2 {  
    public static int menu(){  
        int opcion = 0;  
        System.out.println("\n\n\t -- MENU PUNTO 2 --");  
        System.out.println("\t 1. MOSTRAR CUENTA.");  
        System.out.println("\t 2. RETIRAR.");  
        System.out.println("\t 3. TRANSFERIR CBU.");  
        System.out.println("\t 4. TRANSFERIR ALIAS");  
        System.out.println("\t 5. SALIR");  
        opcion = Metodo.validarOpcion(5);  
  
        return opcion;  
    }  
  
    public static void main(String[] arg) {  
        CuentaSueldo negocio = new CuentaSueldo(11111111, 0,  
10,1,"Negocio","CuentaSueldo", 123456789987654321L, 15000);  
  
        long dni = Metodo.pedirLong("> INGRESE DNI: ");  
        double saldo = Metodo.pedirDoublePositivo("> INGRESE EL SALDO: ");  
        double interes = Metodo.pedirDoublePositivo("> INGRESE EL INTERES: ");  
        int legajo = Metodo.pedirEntero("> INGRESE LEGAJO: ");  
        String institucion = Metodo.cadena("> INGRESE INSTITUCION: ");  
        String beneficios = Metodo.cadena("> INGRESE BENEFICIOS: ");  
        long cbu = Metodo.pedirLong("> INGRESE CBU: ");  
        double tope = Metodo.pedirDoublePositivo("> INGRESE TOPE: ");  
        CuentaSueldo cliente = new CuentaSueldo(dni, saldo,  
interes,legajo,institucion,beneficios, cbu,tope);  
  
        int opcion = 0;  
        while(opcion != 5) {  
            opcion = menu();  
            switch (opcion) {  
                case 1: System.out.println("\n - MOSTRAR CUENTA");  
                cliente.mostrarSueldo();  
                break;  
                case 2: System.out.println("\n - EXTRAER DINERO DE LA CUENTA");  
                double retiro = Metodo.pedirDouble(" > MONTO: ");  
                cliente.retirar(retiro);  
                break;  
                case 3: System.out.println("\n - TRANSFERIR DINERO POR CBU");  
            }  
        }  
    }  
}
```

```

        double montoT = Metodo.pedirDouble(" > MONTO: ");
        long cbuT = Metodo.pedirLong(" > INGRESE CBU: ");
        cliente.transferir(montoT, cbuT);
    break;
case 4: System.out.println("\n - TRANSFERIR DINERO POR CBU");
        double montoA = Metodo.pedirDouble(" > MONTO: ");
        String alias = Metodo.cadena(" > ALIAS: ");
        cliente.transferir(montoA, alias);
    break;
}
}
System.out.println("A FINALIZADO EL PROGRAMA.");
}
}

```

Resultados

<p>- CARGA EXITOSA</p> <p>> INGRESE DNI: 4160</p> <p>> INGRESE EL SALDO: 1000000</p> <p>> INGRESE EL INTERES: 5</p> <p>> INGRESE LEGAJO: 1729</p> <p>> INGRESE INSTITUCION: UNJu</p> <p>> INGRESE BENEFICIOS: Estudiante</p> <p>> INGRESE CBU: 10000000000000000001</p> <p>> INGRESE TOPE: 15000</p> <p>- CARGA EXITOSA</p>	<p>-- MENU PUNTO 2 --</p> <ol style="list-style-type: none"> 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR <p>> OPCION: 1</p> <ul style="list-style-type: none"> - MOSTRAR CUENTA - DNI: 4160 - LEGAJO: 1729 - INSTITUCION: UNJu - BENEFICIO: Estudiante - SALDO: 1000000.0 - INTERES: 5.0
<p>-- MENU PUNTO 2 --</p> <ol style="list-style-type: none"> 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR <p>> OPCION: 2</p> <p>- EXTRAER DINERO DE LA CUENTA</p> <p>> MONTO: 5000</p> <p>NO ES POSIBLE EXTRAER. USUARIO: 4160.</p>	<p>-- MENU PUNTO 2 --</p> <ol style="list-style-type: none"> 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR <p>> OPCION: 2</p> <p>- EXTRAER DINERO DE LA CUENTA</p> <p>> MONTO: 15000</p> <p>> NUM. CUENTA:100002</p> <p>- DNI: 4160</p> <p>- SALDO: 985000.0</p> <p>- INTERES: 5.0</p>
<p>-- MENU PUNTO 2 --</p> <ol style="list-style-type: none"> 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR <p>> OPCION: 3</p> <p>- TRANSFERIR DINERO POR CBU</p> <p>> MONTO: 85000</p> <p>> INGRESE CBU: 444</p> <p>* NO ES POSIBLE REALIZAR LA TRANSFERENCIA</p>	<p>-- MENU PUNTO 2 --</p> <ol style="list-style-type: none"> 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR <p>> OPCION: 3</p> <p>- TRANSFERIR DINERO POR CBU</p> <p>> MONTO: 85000</p> <p>> INGRESE CBU: 123456789987654321</p> <p>- TRASFERENCIA HECHA DE: \$85000.0 A 123456789987654321</p> <p>- SALDO ACTUAL:900000.0</p>

<pre>-- MENU PUNTO 2 -- 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR > OPCION: 4 - TRANSFERIR DINERO POR CBU > MONTO: 100000 > ALIAS: mendoza - TRASFERENCIA HECHA DE: \$100000.0 A mendoza - SALDO ACTUAL:800000.0</pre>	<pre>-- MENU PUNTO 2 -- 1. MOSTRAR CUENTA. 2. RETIRAR. 3. TRANSFERIR CBU. 4. TRANSFERIR ALIAS 5. SALIR > OPCION: 5 A FINALIZADO EL PROGRAMA.</pre>
---	--

PUNTO 3

Cree una interfaz llamada OperacionesComunes con 3 atributos y 2 métodos (pagarServicio() y cambiarAlias()).

Interfaz OperacionesComunes

```
public interface OperacionesComunes {
    String moneda = "ARS";
    Integer codigoBanco = 123; //123
    double comision = 0.05; //5%
    public void pagarServicio();
    public void cambiarAlias();
}
```

Cree otra interfaz llamada OperacionesImportantes con un método llamado transferenciaAltoMonto(double monto).

Clase OperacionesImportantes

```
public class OperacionesImportantes {
    protected double montoImportante;
    protected long cbu;
    protected String alias;
    public void transferenciaAltoMonto(double monto) {

    }
}
```

Clase OperacionesBancarias

La interfaz OperacionesBancarias hereda de OperacionesComunes y OperacionesImportantes.

```
public interface OperacionesBancarias extends OperacionesComunes {}
```

Clase CuentaSueldo

Luego la clase CuentaSueldo debe implementar la interfaz OperacionesBancarias.

```
public class CuentaSueldo extends CuentaBancaria implements OperacionesBancarias{}
```

PUNTO 4

Defina una clase abstracta llamada **Forma**, con un atributo **NombreForma** y dos métodos **area()** y **perimetro()**.

Clase Forma

```
package punto4;
public abstract class Forma {
    public String nombreForma;
    public Forma() {           //CONSTRUCTORES
    }

    public Forma(String nombreForma) {
        this.nombreForma = nombreForma;
    }

    public abstract double area();
    public abstract double perimetro();
}
```

Luego cree una clase llamada **Cuadrilatero** que herede de **Forma** e implemente los métodos definidos en **Forma**.

Clase Cuadrilatero

```
package punto4;
public class Cuadrilatero extends Forma {
    double L1 = 0, L2 = 0;

    public Cuadrilatero() {}

    public Cuadrilatero(String nombre, double L1, double L2) {
        this.nombreForma = nombre;
        this.L1 = L1;
        this.L2 = L2;
    }

    public double area() {
        double calculo = L1 * L2;
        return calculo;
    }
    public double perimetro() {
        double calculo = 2 * (L1 + L2);
        return calculo;
    }
}
```

Además agregue los atributos y métodos necesarios para poder realizar los cálculos del área y perímetro de un cuadrilátero. Desde el **main()** cree un objeto de tipo **Cuadrilatero** y muestre la ejecución de los métodos **area()** y **perimetro()**.

Clase Main del Punto 4

```
package punto4;
import metodos.Metodo;
public class Punto4 {
    public static void main (String [] arg) {
        Cuadrilatero c = null;
        double l1 = 0, l2 = 0;
        l1 = Metodo.dimencion("> LADO 1: ", 0);
        l2 = Metodo.dimencion("> LADO 2: ", 0);
        if (l1 > 0 || l2 > 0 ) {
            if(l1 == l2) {
                c = new Cuadrilatero ("Cuadrado",l1,l2);
            }
        }
    }
}
```

```

        else{
            c = new Cuadrilatero ("Resctangulo",l1,l2);
        }
    }
System.out.println("\n\n- NOMBRE: "+c.nombreForma);
System.out.println("- AREA: "+c.area());
System.out.println("- PERIMETRO: "+c.perimetro());
}
} // punto4

```

Resultados

> LADO 1: 55 > LADO 2: 55 - NOMBRE: Cuadrado - AREA: 3025.0 - PERIMETRO: 220.0	> LADO 1: 46 > LADO 2: 50 - NOMBRE: Rectangulo - AREA: 2300.0 - PERIMETRO: 192.0	> LADO 1: -4 INGRESE UN NUMERO MAYOR A 0.0 > LADO 1: 5 > LADO 2: 6 - NOMBRE: Rectangulo - AREA: 30.0 - PERIMETRO: 22.0
--	--	--

PUNTO 5

Al igual que en el punto anterior, cree una clase Circulo que herede de Forma e implemente los métodos area() y perimetro() de un círculo.

Clase Circulo

```

package punto5;
import punto4.Forma;
public class Circulo extends Forma {
    double radio=0;
    public Circulo() {}
    public Circulo(String nombre, double radio) {
        this.nombreForma = nombre;
        this.radio = radio;
    }
    public double area() {
        double calculo = Math.PI * Math.pow(radio,2);
        return calculo;
    }
    public double perimetro() {
        double calculo = 2 * Math.PI * radio;
        return calculo;
    }
}

```

Desde el main() cree un objeto de tipo Circulo y muestre la ejecución de los métodos area() y perimetro().

Clase Main del Punto 5

```

package punto5;
import metodos.Metodo;
public class Punto5 {
    public static void main (String [] arg) {
        Circulo c;
    }
}

```

PROGRAMACIÓN CONCURRENTE

Tema: Programación Orientada a Objetos

```
        double radio = Metodo.dimencion("> INGRESE EL RADIO: ", 0);
        c = new Circulo("Circulo",radio);
        System.out.println("\n\nCALCULOS:"+c.nombreForma);
        System.out.println("- AREA: "+Metodo.redondear(c.area(), 4));
        System.out.println("- PERIMETRO: "+Metodo.redondear(c.perimetro(),4));
    }
}
```

Resultados

> INGRESE EL RADIO: 25	> INGRESE EL RADIO: -6
- NOMBRE:Circulo	INGRESE UN NUMERO MAYOR A 0.0
- AREA: 1963.4954	> INGRESE EL RADIO: 5
- PERIMETRO: 157.0796	- NOMBRE:Circulo
	- AREA: 78.5398
	- PERIMETRO: 31.4159

PUNTO 6

Realice lo mismo que en el punto 4) pero en lugar de utilizar una clase abstracta Forma, hágalo mediante una interfaz.

Interfaz Forma

```
package punto6;
public interface Forma {
    public abstract double area();
    public abstract double perimetro();
}
```

Clase Cuadrilatero

```
package punto6;
public class Cuadrilatero implements Forma{
    private double lado1 = 0;
    private double lado2 = 0;
    public Cuadrilatero() {
    }
    public Cuadrilatero(double lado1, double lado2) {
        this.lado1 = lado1;
        this.lado2 = lado2;
    }

    @Override
    public double area() {
        return lado1 * lado2;
    }
    @Override
    public double perimetro() {
        return 2 * (lado1 + lado2);
    }
}
```

Clase Main del Punto 6

```
package punto6;
import metodos.Metodo;
import punto6.Cuadrilatero;
```

PROGRAMACIÓN CONCURRENTE

Tema: Programación Orientada a Objetos

```
public class Punto6 {  
    public static void main (String [] arg) {  
        //Cuadrilatero c = new Cuadrilatero(5,6);  
        Cuadrilatero c = null;  
        double l1 = 0, l2 = 0;  
  
        l1 = Metodo.dimencion("> LADO 1: ", 0);  
        l2 = Metodo.dimencion("> LADO 2: ", 0);  
        if (l1 > 0 || l2 >0 ) {  
            if(l1 == l2) {  
                c = new Cuadrilatero (l1,l2);  
            }  
            else{  
                c = new Cuadrilatero (l1,l2);  
            }  
        }  
        System.out.println("- AREA DEL CUADRILATERO: "+c.area());  
        System.out.println("- PERIMETRO: "+c.perimetro());  
    }  
}
```

Resultados

> LADO 1: 5 > LADO 2: 4 - NOMBRE: Rectangulo - AREA: 20.0 - PERIMETRO: 18.0	> LADO 1: -9 INGRESE UN NUMERO MAYOR A 0.0 - NOMBRE: Rectangulo - AREA: 600.0 - PERIMETRO: 140.0
---	--