TRABAJO PRÁCTICO II: Java

1. Verificación de Año Bisiesto.
Escribe un programa en Java que solicite al usuario un año y determine si es bisiesto.
Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.

Ejemplo de entrada/salida:
Ingrese un año: 2024
El año 2024 es bisiesto.
Ingrese un año: 1900
El año 1900 no es bisiesto.

```java
import java.util.Scanner;
public class punto1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int anio;

        System.out.println("Ingrese el año para ver si es bisiesto: ");
        anio = Integer.parseInt(input.nextLine());

        if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {
            System.out.println("El año " + anio + " es bisiesto");
        } else {
            System.out.println("El año " + anio + " no es bisiesto");
        }
    }
}
```

Salida por pantalla ingresando 2024:

```
run:
Ingrese el a□o para ver si es bisiesto:
2024
El a□o 2024 es bisiesto
BUILD SUCCESSFUL (total time: 275 minutes 58 seconds)
```

Salida por pantalla ingresando 1900:

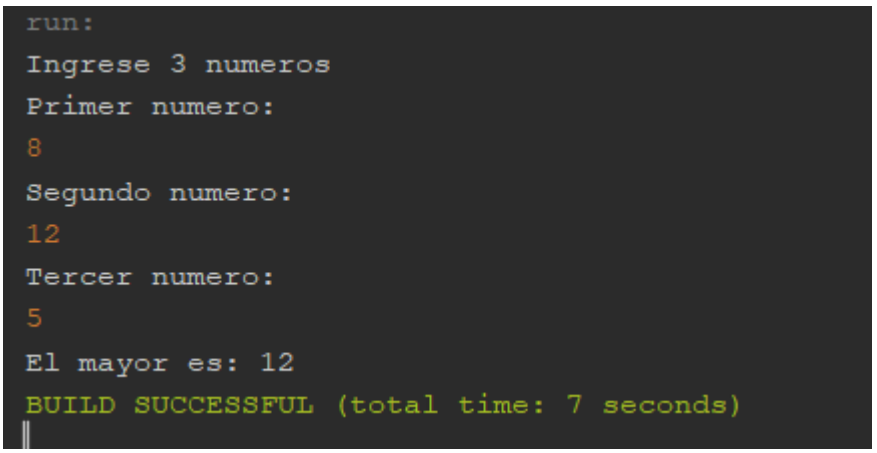
```
run:
Ingrese el a?o para ver si es bisiesto:
1900
El a?o 1900 no es bisiesto
BUILD SUCCESSFUL (total time: 4 seconds)
```

2. Determinar el Mayor de Tres Números.

Escribe un programa en Java que pida al usuario tres números enteros y determine cuál es el mayor.

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner (System. in) ;

int numUno, numDos, numTres, mayor;

System.out.println("Ingrese 3 numeros");
System.out.println("Primer numero: ");
numUno = Integer.parseInt(input.nextLine());
System.out.println("Segundo numero: ");
numDos = Integer.parseInt (input.nextLine());
System.out.println ("Tercer numero: ");
numTres = Integer.parseInt(input.nextLine());

mayor = numUno;

if (numDos > mayor) {
   mayor = numDos;
}
if (numTres > mayor) {
mayor = numTres;
}
System.out.println ("El mayor es: " + mayor) ;
    }
  }
```

Salida por pantalla:



Ejemplo de entrada/salida:
Ingrese el primer número: 8
Ingrese el segundo número: 12
Ingrese el tercer número: 5
El mayor es: 12

3. Clasificación de Edad.
Escribe un programa en Java que solicite al usuario su edad y clasifique su etapa de vida según la siguiente tabla:

Menor de 12 años: "Niño"
Entre 12 y 17 años: "Adolescente"
Entre 18 y 59 años: "Adulto"
60 años o más: "Adulto mayor"

```java
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System. in);

    int edad;
    String etapaVida = "";


      System.out.println("Ingrese su edad:");
      edad = Integer.parseInt(input.nextLine());

      if (edad < 12) {
      etapaVida = "Niño";
      } else if (edad >= 12 && edad <= 17) {
      etapaVida = "Adolescente";
      } else if (edad >= 18 && edad <= 59) {
      etapaVida = "Adulto";
      } else if (edad >= 60 ) {
      etapaVida = "Adulto mayor";
      }

System.out.println("Eres un " + etapaVida);
    }

}
```

Ejemplo de entrada/salida:
Ingrese su edad: 25
Eres un Adulto.
Salida por pantalla:

```
run:
Ingrese su edad:
25
Eres un Adulto
BUILD SUCCESSFUL (total time: 2 seconds)
```

Ingrese su edad: 10
Eres un Niño.
Salida por pantalla:

```
run:
Ingrese su edad:
10
Eres un Ni?o
BUILD SUCCESSFUL (total time: 4 seconds)
```

4. Calculadora de Descuento según categoría.
Escribe un programa que solicite al usuario el precio de un producto y su categoría (A, B o C).
Luego, aplique los siguientes descuentos:

Categoría A: 10% de descuento
Categoría B: 15% de descuento
Categoría C: 20% de descuento
El programa debe mostrar el precio original, el descuento aplicado y el precio final

Ejemplo de entrada/salida:
Ingrese el precio del producto: 1000
Ingrese la categoría del producto (A, B o C): B
Descuento aplicado: 15%
Precio final: 850.0

```
        Scanner input = new Scanner (System. in);

int precio;
double precioConDescuento;
char categoria;

System.out.println("Ingrese el precio del producto");
precio = Integer.parseInt(input.nextLine());

System.out.println("Ingrese la categoria del producto (A, B o C) ");
categoria = input.nextLine().charAt(0);

switch (categoria) {
   case 'A', 'a' -> {
       precioConDescuento = precio -(precio * 0.10);
       System.out.print("Descuento aplicado 10%");
       System.out.println("");
       System.out.print("Precio final: $" + precioConDescuento);
           }
   case 'B', 'b' -> {
       precioConDescuento = precio - (precio * 0.15);
       System.out.print("Descuento aplicado 15%");
       System.out.println("");
       System.out.print ("Precio final: $" + precioConDescuento);
           }

   case 'C', 'c' -> {
       precioConDescuento = precio - (precio * 0.20);
       System.out.print("Descuento aplicado 20%");
       System.out.println("");
       System.out.print("Precio final: $" + precioConDescuento);
           }
   default -> System.out.println("No ingreso una categoria valida");
```

Salida por pantalla:

```
run:
Ingrese el precio del producto
1000
Ingrese la categoria del producto (A, B o C)
B
Descuento aplicado 15%
Precio final: $850.0BUILD SUCCESSFUL (total time: 24 seconds)
```

Estructuras de Repetición:

5. Suma de Números Pares (while).

Escribe un programa que solicite números al usuario y sume solo los números pares. El ciclo debe continuar hasta que el usuario ingrese el número 0, momento en el que se debe mostrar la suma total de los pares ingresados.

```java
import java.util.Scanner;
public class punto5 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int num = 1;
        int sumaPares = 0;

        while (num != 0) {
            System.out.print("Ingrese un numero (0 para terminar): ");
            num = Integer.parseInt(input.nextLine());

            if (num % 2 == 0 && num != 0) {
                sumaPares = sumaPares + num;
            }
        }

        System.out.println("La suma de los numeros pares es: " + sumaPares);
    }
}
```

Ejemplo de entrada/salida:

Ingrese un número (0 para terminar): 4
Ingrese un número (0 para terminar): 7
Ingrese un número (0 para terminar): 2
Ingrese un número (0 para terminar): 0
La suma de los números pares es: 6

Salida por pantalla:

```
run:
Ingrese un numero (0 para terminar): 4
Ingrese un numero (0 para terminar): 7
Ingrese un numero (0 para terminar): 2
Ingrese un numero (0 para terminar): 0
La suma de los numeros pares es: 6
BUILD SUCCESSFUL (total time: 21 seconds)
```

6. Contador de Positivos, Negativos y Ceros (for).
Escribe un programa que pida al usuario ingresar 10 números enteros y cuente cuántos son positivos, negativos y cuántos son ceros.

```java
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    int contadorPositivos = 0, contadorNegativos = 0, contadorCeros = 0;
    int num;

    for (int i = 1; i <= 10; i++) {
        System.out.println("Ingrese el número " + i + ": ");
        num = Integer.parseInt(input.nextLine());

        if (num > 0) {
            contadorPositivos++;
        } else if (num < 0) {
            contadorNegativos++;
        } else {
            contadorCeros++;
        }
    }

    System.out.println("Resultados:");
    System.out.println("Positivos: " + contadorPositivos);
    System.out.println("Negativos: " + contadorNegativos);
    System.out.println("Ceros: " + contadorCeros);
}
```

Ejemplo de entrada/salida:
Ingrese el número 1: -5
Ingrese el número 2: 3
Ingrese el número 3: 0
Ingrese el número 4: -1
Ingrese el número 5: 6
Ingrese el número 6: 0
Ingrese el número 7: 9
Ingrese el número 8: -3
Ingrese el número 9: 4
Ingrese el número 10: -8
Resultados: Positivos: 4

Negativos: 4
Ceros: 2

Salida por pantalla:

```
run:
Ingrese el n�mero 1:
-5
Ingrese el n�mero 2:
3
Ingrese el n�mero 3:
0
Ingrese el n�mero 4:
-1
Ingrese el n�mero 5:
6
Ingrese el n�mero 6:
0
Ingrese el n�mero 7:
9
Ingrese el n�mero 8:
-3
Ingrese el n�mero 9:
4
Ingrese el n�mero 10:
-8
Resultados:
Positivos: 4
Negativos: 4
Ceros: 2
BUILD SUCCESSFUL (total time: 1 minute 6 seconds)
```

7. Validación de Nota entre 0 y 10 (do-while).
Escribe un programa que solicite al usuario una nota entre 0 y 10. Si el usuario ingresa un número fuera de este rango, debe seguir pidiéndole la nota hasta que ingrese un valor válido.

```java
import java.util.Scanner;
public class punto7 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int nota;

        do {
            System.out.println("Ingrese una nota (0-10): ");
            nota = Integer.parseInt(input.nextLine());

            if (nota < 0 || nota > 10) {
                System.out.println("ERROR: nota inválida. Ingrese una nota entre 0 y 10.");
            } else {
                System.out.println("Nota guardada correctamente.");
            }
        } while (nota < 0 || nota > 10);
    }
```

Ejemplo de entrada/salida:
Ingrese una nota (0-10): 15
Error: Nota inválida.

Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): -2
Error: Nota inválida.
Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10): 8
Nota guardada correctamente.

Salida por pantalla:

```
run:
Ingrese una nota (0-10):
15
ERROR: nota inv�lida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10):
-2
ERROR: nota inv�lida. Ingrese una nota entre 0 y 10.
Ingrese una nota (0-10):
8
Nota guardada correctamente.
BUILD SUCCESSFUL (total time: 50 seconds)
```

Funciones:

8. Cálculo del Precio Final con impuesto y descuento.
Crea un método calcularPrecioFinal(double impuesto, double descuento) que calcule el precio final de un producto en un e-commerce. La fórmula es:

PrecioFinal = PrecioBase + (PrecioBase×Impuesto) − (PrecioBase×Descuento)

PrecioFinal = PrecioBase + (PrecioBase \times Impuesto) - (PrecioBase \times Descuento)

Desde main(), solicita el precio base del producto, el porcentaje de impuesto y el porcentaje de descuento, llama al método y muestra el precio final.

```java
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    double precio, impuesto, descuento;

    System.out.println("Ingrese el precio base del producto: ");
    precio = input.nextDouble();

    System.out.println("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
    impuesto = input.nextDouble();

    System.out.println("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
    descuento = input.nextDouble();

    double precioFinal = calcularPrecioFinal(precio, impuesto, descuento);
    System.out.println("El precio final del producto es: $" + precioFinal);
}

/**
 * Calcula el precio final de un producto aplicando un impuesto y un descuento.
 * @param precioBase El precio base del producto (sin impuesto ni descuento).
 * @param impuesto El porcentaje de impuesto a aplicar (por ejemplo, 21 para 21%).
 * @param descuento El porcentaje de descuento a aplicar (por ejemplo, 15 para 15%).
 * @return El precio final del producto con el impuesto sumado y el descuento restado.
 */
public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
    double precioFinal = precioBase + ((precioBase * impuesto) / 100) - ((precioBase * descuento) / 100);
    return precioFinal;
}
```

Ejemplo de entrada/salida:

Ingrese el precio base del producto: 100
Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): 10
Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): 5
El precio final del producto es: 105.0

Salida por pantalla:

```
run:
Ingrese el precio base del producto:
100
Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%):
10
Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%):
5
El precio final del producto es: $105.0
BUILD SUCCESSFUL (total time: 36 seconds)
```

9. Composición de funciones para calcular costo de envío y total de compra.

a. calcularCostoEnvio(double peso, String zona): Calcula el costo de envío basado en la zona de envío (Nacional o Internacional) y el peso del paquete.

Nacional: $5 por kg

Internacional: $10 por kg

b. calcularTotalCompra(double precioProducto, double costoEnvio): Usa calcularCosto Envio para sumar el costo del producto con el costo de envío.

Desde main(), solicita el peso del paquete, la zona de envío y el precio del producto. Luego, muestra el total a pagar.

Ejemplo de entrada/salida:
Ingrese el precio del producto: 50
Ingrese el peso del paquete en kg: 2
Ingrese la zona de envío (Nacional/Internacional): Nacional
El costo de envío es: 10.0
El total a pagar es: 60.0

Calcular costo envío:

```java
/**
 * Calcula el costo de envío en función del peso y la zona de destino.
 *
 * @param peso Peso del paquete en kilogramos.
 * @param zona Zona de envío: puede ser "Nacional" o "Internacional".
 * @return El costo de envío calculado: $5 por kilo para envíos nacionales, $10 por kilo para internacionales.
 */
public static double calcularCostoEnvio(double peso, String zona) {
    double costoEnvio = 0;

    if (zona.equalsIgnoreCase("Nacional")) {
        costoEnvio = peso * 5;
    } else if (zona.equalsIgnoreCase("Internacional")) {
        costoEnvio = peso * 10;
    } else {
        System.out.println("Zona no válida, se asume costo 0.");
    }

    return costoEnvio;
}
```

Calcular total compra:

```java
/**
 * Calcula el total de la compra sumando el precio del producto y el costo de envío.
 *
 * @param precioProducto Precio base del producto.
 * @param costoEnvio     Costo de envío previamente calculado.
 * @return El monto total a pagar por la compra.
 */
public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
    return precioProducto + costoEnvio;
}
```

Main:

```java
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // Declaramos input para leer los datos
    Scanner input = new Scanner(System.in);

    // Declaramos variables para guardar los datos que pedimos al usuario
    double peso, precioProducto;
    String zona;

    // Solicitamos el precio del producto
    System.out.print("Ingrese el precio del producto: ");
    precioProducto = input.nextDouble();

    // Solicitamos el peso del paquete
    System.out.print("Ingrese el peso del paquete: ");
    peso = input.nextDouble();
    input.nextLine(); // Consumimos el Enter pendiente

    // Solicitamos la zona de envío
    System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
    zona = input.nextLine();

    // Calculamos el costo de envío
    double costoEnvio = calcularCostoEnvio(peso, zona);

    // Calculamos el precio final de la compra
    double precioFinal = calcularTotalCompra(precioProducto, costoEnvio);

    // Mostramos los resultados
    System.out.println("El costo de envío es: " + costoEnvio);
    System.out.println("El total a pagar es: " + precioFinal);
}
```

Salida por pantalla:

```
Ingrese el precio del producto: 50
Ingrese el peso del paquete: 2
Ingrese la zona de env?o (Nacional/Internacional): Nacional
El costo de env?o es: 10.0
El total a pagar es: 60.0
BUILD SUCCESSFUL (total time: 11 seconds)
```

10. Actualización de stock a partir de venta y recepción de productos.

Crea un método actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida), que calcule el nuevo stock después de una venta y recepción

de productos:
NuevoStock = StockActual − CantidadVendida + CantidadRecibida

NuevoStock = CantidadVendida + CantidadRecibida

Desde main(), solicita al usuario el stock actual, la cantidad vendida y la cantidad recibida, y muestra el stock actualizado.

Ejemplo de entrada/salida:
Ingrese el stock actual del producto: 50
Ingrese la cantidad vendida: 20
Ingrese la cantidad recibida: 30
El nuevo stock del producto es: 60

```java
public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Pedimos datos al usuario
        System.out.print("Ingrese el stock actual del producto: ");
        int stockActual = input.nextInt();

        System.out.print("Ingrese la cantidad vendida: ");
        int cantidadVendida = input.nextInt();

        System.out.print("Ingrese la cantidad recibida: ");
        int cantidadRecibida = input.nextInt();

        // Calculamos el nuevo stock
        int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);

        // Mostramos resultado
        System.out.println("El nuevo stock del producto es: " + nuevoStock);
    }

    /**
     * Actualiza el stock disponible de un producto según la cantidad vendida y recibida.
     *
     * @param stockActual      Cantidad actual en stock antes de la operación.
     * @param cantidadVendida  Cantidad de unidades vendidas (se restan del stock).
     * @param cantidadRecibida Cantidad de unidades recibidas o repuestas (se suman al stock).
     * @return El nuevo stock disponible después de aplicar ventas y reposiciones.
     */
    public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
        int nuevoStock = (stockActual - cantidadVendida) + cantidadRecibida;
        return nuevoStock;
    }
}
```

Salida por pantalla:

```
run:
Ingrese el stock actual del producto: 50
Ingrese la cantidad vendida: 20
Ingrese la cantidad recibida: 30
El nuevo stock del producto es: 60
BUILD SUCCESSFUL (total time: 13 seconds)
```

11. Cálculo de descuento especial usando variable global.

Declara una variable global Ejemplo de entrada/salida: = 0.10. Luego, crea un método calcularDescuentoEspecial(double precio) que use la variable global para calcular el descuento especial del 10%.

Dentro del método, declara una variable local descuentoAplicado, almacena el valor del descuento y muestra el precio final con descuento.

Ejemplo de entrada/salida:

Ingrese el precio del producto: 200
El descuento especial aplicado es: 20.0
El precio final con descuento es: 180.0

Calcular descuento especial:

```java
/**
 * Calcula y muestra el descuento especial aplicado a un producto,
 * junto con el precio final.
 *
 * @param precio Precio original del producto antes de aplicar el descuento
 */
public static void calcularDescuentoEspecial(double precio) {
    double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
    double precioFinal = precio - descuentoAplicado;

    // Mostramos los resultados
    System.out.println("El descuento aplicado es de: $" + descuentoAplicado);
    System.out.println("El precio final del producto es: $" + precioFinal);
}
```

Main:

```java
import java.util.Scanner;
public class punto_11 {

    // Variable global: descuento especial del 10%
    public static final double DESCUENTO_ESPECIAL = 0.10;

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Pedir el precio original al usuario
        System.out.print("Ingrese el precio del producto: ");
        double precio = input.nextDouble();

        // Llamamos al método que aplica el descuento
        calcularDescuentoEspecial(precio);
    }
```

Salida por pantalla:

```
run:
Ingrese el precio del producto: 200
El descuento aplicado es de: $20.0
El precio final del producto es: $180.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

Arrays y Recursividad:

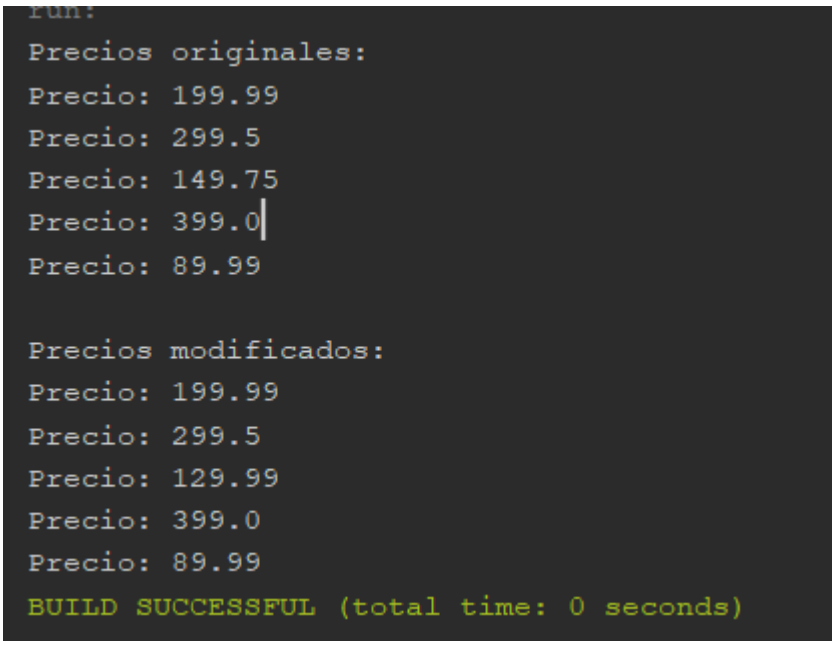
12. Modificación de un array de precios y visualización de resultados. Crea un programa que:
a. Declare e inicialice un array con los precios de algunos productos.
b. Muestre los valores originales de los precios.
c. Modifique el precio de un producto específico.
d. Muestre los valores modificados.

Salida esperada:
Precios originales:
Precio: $199.99
Precio: $299.5
Precio: $149.75
Precio: $399.0
Precio: $89.99

Precios modificados:
Precio: $199.99
Precio: $299.5
Precio: $129.99
Precio: $399.0
Precio: $89.99

```java
public class punto_12 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        double[] precioProducto = {199.99, 299.5, 149.75, 399.0, 89.99};


        System.out.println("Precios originales:");
        for (double precio : precioProducto) {
            System.out.println("Precio: " + precio);
        }

        precioProducto[2] = 129.99;

        System.out.println("\nPrecios modificados:");
        for (double precio : precioProducto) {
            System.out.println("Precio: " + precio);
        }
    }
}
```

Salida por pantalla:



13. Impresión recursiva de arrays antes y después de modificar un elemento.

Crea un programa que:
a. Declare e inicialice un array con los precios de algunos productos.
b. Use una función recursiva para mostrar los precios originales.
c. Modifique el precio de un producto específico.
d. Use otra función recursiva para mostrar los valores modificados.

Salida esperada:

Precios originales:
Precio: $199.99
Precio: $299.5
Precio: $149.75
Precio: $399.0
Precio: $89.99

Precios modificados:
Precio: $199.99
Precio: $299.5
Precio: $129.99
Precio: $399.0
Precio: $89.99

```java
/**
 * Muestra los precios de un arreglo de manera recursiva.
 *
 * @param precios arreglo de precios
 * @param indice posición actual que se está mostrando
 */
public static void mostrarPreciosRecursivo(double[] precios, int indice) {
    // Caso base: si el índice llegó al final del array, detenemos
    if (indice >= precios.length) {
        return;
    }

    // Imprimimos el precio en la posición actual
    System.out.println("Precio[" + indice + "] = " + precios[indice]);

    // Llamada recursiva para el siguiente elemento
    mostrarPreciosRecursivo(precios, indice + 1);
}

public static void main(String[] args) {
    // Declaramos e inicializamos el array
    double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

    // Mostramos los precios originales
    System.out.println("Precios originales:");
    mostrarPreciosRecursivo(precios, 0);

    // Modificamos el precio de la posición 2
    precios[2] = 129.99;

    // Mostramos los precios modificados
    System.out.println("Precios modificados:");
    mostrarPreciosRecursivo(precios, 0);
}
```

Salida por pantalla:

```
run:
Precios originales:
Precio[0] = 199.99
Precio[1] = 299.5
Precio[2] = 149.75
Precio[3] = 399.0
Precio[4] = 89.99
Precios modificados:
Precio[0] = 199.99
Precio[1] = 299.5
Precio[2] = 129.99
Precio[3] = 399.0
Precio[4] = 89.99
BUILD SUCCESSFUL (total time: 0 seconds)
```