

GUÍA DE PRÁCTICA DE LABORATORIO Nro 04

EXPERIENCIA CURRICULAR: Metodología de Programación

DOCENTE: Mg. Ing. Ivan Petrlik Azabache / Ing. Eric Gustavo Coronel Castillo

SESIÓN: 4

DESARROLLO DE LA PRÁCTICA

CAPACIDAD A TRABAJAR:

Construye Programas básicos usando clases, objetos, estructuras selectivas y repetitivas

CONCEPTOS:

Estructuras de control en Java

Las estructuras de control determinan la secuencia de ejecución de las sentencias de un programa.

Las estructuras de control se dividen en tres categorías:

Secuencial

Condicional o Selectiva

Iterativa o Repetitiva.

1. ESTRUCTURA SECUENCIAL

El orden en que se ejecutan por defecto las sentencias de un programa es secuencial. Esto significa que las sentencias se ejecutan en secuencia, una después de otra, en el orden en que aparecen escritas dentro del programa.

La estructura secuencial está formada por una sucesión de instrucciones que se ejecutan en orden una a continuación de la otra.

Cada una de las instrucciones están separadas por el carácter punto y coma (;).

Las instrucciones se suelen agrupar en bloques.

El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio del mismo, y el carácter llave de cierre (}) para marcar el final.

Ejemplo:

```
{  
instrucción 1;  
instrucción 2;  
instrucción 3;  
}
```

En Java si el bloque de sentencias está constituido por una única sentencia no es obligatorio el uso de las llaves de apertura y cierre ({ }), aunque sí recomendable.

Ejemplo de programa Java con estructura secuencial: Programa que lee dos números por teclado y los muestra por pantalla.

```
/* Programa que lea dos números por teclado y los muestre por pantalla.  
 */  
import java.util.*;  
public class Main {  
    public static void main(String[] args){  
        //declaración de variables  
        int n1, n2;  
        Scanner sc = new Scanner(System.in);  
        //leer el primer número  
        System.out.println("Introduce un número entero: ");  
        n1 = sc.nextInt();        //lee un entero por teclado  
        //leer el segundo número  
        System.out.println("Introduce otro número entero: ");  
        n2 = sc.nextInt();        //lee un entero por teclado  
  
        //mostrar resultado  
        System.out.println("Ha introducido los números: " + n1 + " y " + n2);  
    }  
}
```

Ejemplo de programa Java con estructura secuencial: Programa que lee dos números de tipo double por teclado y calcula y muestra por pantalla su suma, resta y multiplicación.

```
/*
```

```

* Programa que lee dos números de tipo double por teclado
* y muestra su suma, resta y multiplicación.
*/
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double numero1, numero2;
        System.out.println("Introduce el primer número:");
        numero1 = sc.nextDouble();
        System.out.println("Introduce el segundo número:");
        numero2 = sc.nextDouble();
        System.out.println("Números introducido: " + numero1 + " " + numero2);
        System.out.println
            (numero1 + " + " + numero2 + " = " + (numero1+numero2));
        System.out.println
            (numero1 + " - " + numero2 + " = " + (numero1-numero2));
        System.out.println
            (numero1 + " * " + numero2 + " = " + numero1*numero2);
    }
}

```

Para modificar el orden de ejecución de las instrucciones de un programa Java se utilizan las estructuras condicionales y repetitivas.

2. ESTRUCTURA CONDICIONAL, ALTERNATIVA O SELECTIVA

La estructura condicional determina si se ejecutan unas instrucciones u otras según se cumpla o no una determinada condición.

En java la estructura condicional se implementa mediante:

- Instrucción if.
- Instrucción switch.
- Operador condicional ? :

2.1 INSTRUCCION if

Puede ser del tipo:

- Condicional simple: if
- Condicional doble: if ... else ...
- Condicional múltiple: if .. else if ..

La condición debe ser una **expresión booleana** es decir debe dar como resultado un valor booleano (**true ó false**).

Condicional simple: se evalúa la condición y si ésta se cumple se ejecuta una determinada acción o grupo de acciones. En caso contrario se saltan dicho grupo de acciones.

```

    if(expresión_booleana){
        instrucción 1
        instrucción 2
        .....
    }

```

Si el bloque de instrucciones tiene **una sola instrucción** no es necesario escribir las llaves { } aunque para evitar confusiones se recomienda escribir las llaves siempre.

Ejemplo de programa Java con estructura condicional: Programa que pide por teclado la nota obtenida por un alumno y muestra un mensaje si el alumno ha aprobado.

```

/*
 * Programa que pide una nota por teclado y muestra un mensaje si la nota es
 * mayor o igual que 5
 */
import java.util.*;
public class Ejemplo0If {
    public static void main( String[] args ){
        Scanner sc = new Scanner( System.in );
        System.out.print("Nota: ");
        int nota = sc.nextInt();
        if (nota >= 5 ){
            System.out.println("Enorabuena!!");
            System.out.println("Has aprobado");
        }
    }
}

```

Condicional doble: Se evalúa la condición y si ésta se cumple se ejecuta una determinada instrucción o grupo de instrucciones. Si no se cumple se ejecuta otra instrucción o grupo de instrucciones.

```

    if(expresión booleana){
        instrucciones 1
    }
    else{
        instrucciones 2
    }

```

Ejemplo de programa Java que contiene una estructura condicional doble: Programa que lee la nota de un alumno y muestra si el alumno ha aprobado o no.

```

/*
 * Programa que pide una nota por teclado y muestra si se ha aprobado o no
 */
import java.util.*;
public class Ejemplo0If {
    public static void main( String[] args ){
        Scanner sc = new Scanner( System.in );
        System.out.print("Nota: ");
        int nota = sc.nextInt();
        if (nota >= 5 ){
            System.out.println("Enorabuena!!");
            System.out.println("Has aprobado");
        }
        else
            System.out.println("Lo Siento, has suspendido");
    }
}

```

```
}  
}
```

Otro ejemplo de programa Java que contiene una estructura condicional doble: Calcular si un número es par. El programa lee un número por teclado y muestra un mensaje indicando si es par o impar.

```
/*  
 * programa que pide un número por teclado y calcula si es par o impar  
 */  
import java.util.*;  
public class EjemploIf {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int num;  
        System.out.println("Introduzca numero: ");  
        num = sc.nextInt();  
        if ((num%2)==0)  
            System.out.println("PAR");  
        else  
            System.out.println("IMPAR");  
    }  
}
```

Condicional múltiple: Se obtiene anidando sentencias if ... else. Permite construir estructuras de selección más complejas.

```
if (expresion_booleana1)  
    instruccion1;  
else if (expresion_booleana2)  
    instruccion2;  
else  
    instruccion3;
```

Cada else se corresponde con el if más próximo que no haya sido emparejado.

Una vez que se ejecuta un bloque de instrucciones, la ejecución continúa en la siguiente instrucción que aparezca después de las sentencias if .. else anidadas.

Ejemplo de programa Java que contiene una estructura condicional múltiple: Programa que lee una hora (número entero) y muestra un mensaje según la hora introducida.

```
/*  
 * Programa que muestra un saludo distinto según la hora introducida  
 */  
import java.util.*;  
public class Ejemplo2If {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int hora;  
        System.out.println("Introduzca una hora (un valor entero): ");  
        hora = sc.nextInt();  
        if (hora >= 0 && hora < 12)  
            System.out.println("Buenos días");  
        else if (hora >= 12 && hora < 21)  
            System.out.println("Buenas tardes");  
        else if (hora >= 21 && hora < 24)  
            System.out.println("Buenas noches");  
        else  
            System.out.println("Hora no válida");  
    }  
}
```

```
}
```

Ejemplo de programa Java que contiene una estructura condicional múltiple: Programa que lee una nota (número entero entre 0 y 10) y muestra la calificación equivalente en forma de texto.

```
/*
 * programa que lee una nota y escribe la calificación correspondiente
 */
import java.util.*;
public class Ejemplo3If {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double nota;
        System.out.println("Introduzca una nota entre 0 y 10: ");
        nota = sc.nextDouble();
        System.out.println("La calificación del alumno es ");
        if(nota < 0 || nota > 10)
            System.out.println("Nota no válida");
        else if(nota==10)
            System.out.println("Matrícula de Honor");
        else if (nota >= 9)
            System.out.println("Sobresaliente");
        else if (nota >= 7)
            System.out.println("Notable");
        else if (nota >= 6)
            System.out.println("Bien");
        else if (nota >= 5)
            System.out.println("Suficiente");
        else
            System.out.println("Suspenso");
    }
}
```

Comparar String en Java

Para comparar el contenido de dos Strings en Java se usa el método **equals**:

```
if ((cadena1.equals(cadena2))
```

En caso de que una cadena coincida exactamente con una constante se puede usar ==

```
String nombre = "Lucas";
```

```
if (nombre == "Lucas")
```

Para comparar Strings en el orden alfabético se usa el método **compareTo**

```
if (cadena1.compareTo(cadena2) < 0) // cadena1 antes que cadena2
```

```
if (cadena1.compareTo(cadena2) > 0) // cadena1 después que cadena2
```

```
if (cadena1.compareTo(cadena2) == 0) // cadena1 igual que cadena2
```

2.2 INSTRUCCION switch

Se utiliza para seleccionar una de entre múltiples alternativas.

La forma general de la instrucción switch en Java es la siguiente:

```
switch (expresión){
    case valor 1:
        instrucciones;
        break;
    case valor 2:
        instrucciones;
        break;
    . . .
    default:
```

instrucciones;

}

La instrucción switch se puede usar con datos de tipo byte, short, char e int. También con tipos enumerados y con las clases envolventes Character, Byte, Short e Integer. A partir de Java 7 también pueden usarse datos de tipo String en un switch.

Funcionamiento de la instrucción switch:

Se evalúa la expresión y salta al case cuya constante coincida con el valor de la expresión. Se ejecutan las instrucciones que siguen al case seleccionado hasta que se encuentra un break o hasta el final del switch. El break produce un salto a la siguiente instrucción a continuación del switch.

Si ninguno de estos casos se cumple se ejecuta el bloque default (si existe). No es obligatorio que exista un bloque default y no tiene por qué ponerse siempre al final, aunque es lo habitual.

Ejemplo de programa Java que contiene una instrucción switch: Programa que lee por teclado un mes (número entero) y muestra el nombre del mes.

```
/*  
 * Programa que pide un número de mes y muestra el nombre correspondiente  
 */  
import java.util.*;  
public class Ejemplo0Switch {  
    public static void main(String[] args) {  
        int mes;  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Introduzca un numero de mes: ");  
        mes = sc.nextInt();  
        switch (mes)  
        {  
            case 1: System.out.println("ENERO");  
                    break;  
            case 2: System.out.println("FEBRERO");  
                    break;  
            case 3: System.out.println("MARZO");  
                    break;  
            case 4: System.out.println("ABRIL");  
                    break;  
            case 5: System.out.println("MAYO");  
                    break;  
            case 6: System.out.println("JUNIO");  
                    break;  
            case 7: System.out.println("JULIO");  
                    break;  
            case 8: System.out.println("AGOSTO");  
                    break;  
            case 9: System.out.println("SEPTIEMBRE");  
                    break;  
            case 10: System.out.println("OCTUBRE");  
                    break;  
            case 11: System.out.println("NOVIEMBRE");  
                    break;  
            case 12: System.out.println("DICIEMBRE");  
                    break;  
            default : System.out.println("Mes no válido");  
        }  
    }  
}
```

3. ESTRUCTURA ITERATIVA O REPETITIVA

Permiten ejecutar de forma repetida un bloque específico de instrucciones.

Las instrucciones se repiten mientras o hasta que se cumpla una determinada condición. Esta condición se conoce como **condición de salida**.

Tipos de estructuras repetitivas:

ciclo while

ciclo do – while

ciclo for

3.1 CICLO WHILE

Las instrucciones se repiten mientras la condición sea cierta. La condición **se comprueba al principio** del bucle por lo que las acciones se pueden ejecutar **0 ó más veces**.

La ejecución de un bucle while sigue los siguientes pasos:

1. Se evalúa la condición.

Si el resultado es false las instrucciones no se ejecutan y el programa sigue ejecutándose por la siguiente instrucción a continuación del while.

Si el resultado es true se ejecutan las instrucciones y se vuelve al paso 1

Ejemplo de programa Java que contiene una instrucción while:

Programa que lee números por teclado. La lectura acaba cuando el número introducido sea negativo. El programa calcula y muestra la suma de los números leídos.

```
/*
 * Programa que lee números hasta que se lee un negativo y muestra la
 * suma de los números leídos
 */
import java.util.*;
public class Ejemplo1While {
    public static void main(String[] args) {
        int suma = 0, num;
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduzca un número: ");
        num = sc.nextInt();
        while (num >= 0){
            suma = suma + num;
            System.out.print("Introduzca un número: ");
            num = sc.nextInt();
        }
        System.out.println("La suma es: " + suma );
    }
}
```

Ejemplo de programa Java que contiene una instrucción while:

Programa que lee un número entero N y muestra N asteriscos.

```
/*
 * programa que lee un número n y muestra n asteriscos
 */
import java.util.*;
public class Ejemplo2While {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n, contador = 0;
        System.out.print("Introduce un número: ");
        n = sc.nextInt();
        while (contador < n){
            System.out.println(" * ");
        }
    }
}
```



```

        contador++;
    }
}
}

```

Ejemplo de programa Java con una instrucción while:

```

/*
 * programa que muestra una tabla de equivalencias entre
 * grados Fahrenheit y grados celsius
 */
public class Ejemplo3While {
    public static void main(String[] args) {
        final int VALOR_INICIAL = 10; // limite inf. tabla
        final int VALOR_FINAL = 100; // limite sup. tabla
        final int PASO = 10 ; // incremento
        int fahrenheit;
        double celsius;
        fahrenheit = VALOR_INICIAL;
        System.out.printf("Fahrenheit \t Celsius \n");
        while (fahrenheit <= VALOR_FINAL ){
            celsius = 5*(fahrenheit - 32)/9.0;
            System.out.printf("%7d \t %8.3f \n", fahrenheit, celsius);
            fahrenheit += PASO;
        }
    }
}

```

3.2 CICLO DO - WHILE

Las instrucciones se ejecutan mientras la condición sea cierta.

La condición **se comprueba al final** del bucle por lo que el bloque de instrucciones se ejecutarán **al menos una vez**. Esta es la diferencia fundamental con la instrucción while. Las instrucciones de un bucle while es posible que no se ejecuten si la condición inicialmente es falsa.

La ejecución de un bucle do - while sigue los siguientes pasos:

1. se ejecutan las instrucciones a partir de do{

2. se evalúa la condición.

3. si el resultado es false el programa sigue ejecutándose por la siguiente instrucción a continuación del while.

4. si el resultado es true se vuelve al paso 1

Ejemplo de programa Java que contiene una instrucción do while:

Programa que lee un número entero N. El número debe ser menor que 100.

```

/*
 * Programa que obliga al usuario a introducir un número menor que 100
 */
import java.util.*;
public class Ejemplo1DoWhile {
    public static void main(String[] args) {
        int valor;
        Scanner in = new Scanner( System.in );
        do {
            System.out.print("Escribe un entero < 100: ");
            valor = in.nextInt();
        }while (valor >= 100);
        System.out.println("Ha introducido: " + valor);
    }
}

```

Ejemplo de programa Java con una instrucción do while:

```
/*
 * Programa que lee un número entre 1 y 10 ambos incluidos
 */
import java.util.*;
public class Ejemplo2DoWhile {
    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner( System.in );
        do {
            System.out.print("Escribe un número entre 1 y 10: ");
            n = sc.nextInt();
        }while (n<1 || n >10);
        System.out.println("Ha introducido: " + n);
    }
}
```

3.3 CICLO FOR

Hace que una instrucción o bloque de instrucciones se repitan un **número determinado de veces mientras se cumpla la condición**.

La estructura general de una instrucción for en Java es la siguiente:

```
for(inicialización; condición; incremento/decremento){
instrucción 1;
.....
instrucción N;
}
```

A continuación de la palabra for y entre paréntesis debe haber siempre **tres zonas separadas por punto y coma**:

zona de inicialización.

zona de condición

zona de incremento ó decremento.

Si en alguna ocasión no es necesario escribir alguna de ellas se pueden dejar en blanco, pero los dos punto y coma deben aparecer.

Inicialización es la parte en la que la variable o variables de control del bucle toman su valor inicial. Puede haber una o más instrucciones en la inicialización, separadas por comas. La inicialización se realiza solo una vez.

Condición es una expresión booleana que hace que se ejecute la sentencia o bloque de sentencias mientras que dicha expresión sea cierta. Generalmente en la condición se compara la variable de control con un valor límite.

Incremento/decremento es una expresión que decrementa o incrementa la variable de control del bucle.

La ejecución de un bucle for sigue los siguientes pasos:

1. Se inicializa la variable o variables de control (inicialización)
2. Se evalúa la condición.
3. Si la condición es cierta se ejecutan las instrucciones. Si es falsa, finaliza la ejecución del bucle y continúa el programa en la siguiente instrucción después del for.
4. Se actualiza la variable o variables de control (incremento/decremento)
5. Se vuelve al punto 2.

Ejemplo de programa Java que contiene una instrucción for:

```
/*
 * programa que muestra los números del 1 al 10
 */
public class Ejemplo0For {
    public static void main(String[] args) {
        int i;
        for(i=1; i<=10;i++)
            System.out.println(i + " ");
    }
}
```

```
}  
}
```

La instrucción for del ejemplo anterior la podemos interpretar así:

Asigna a i el valor inicial 1, mientras que i sea menor o igual a 10 muestra i + " ", a continuación incrementa el valor de i y comprueba de nuevo la condición.

Ejemplo de programa Java con una instrucción for:

```
/*  
 * programa que muestra los números del 10 al 1  
 */  
public class Ejemplo2For {  
    public static void main(String[] args) {  
        int i;  
        for(i=10; i>0;i--)  
            System.out.println(i + " ");  
    }  
}
```

Ejemplo de programa Java con una instrucción for:

```
/*  
 * programa que muestra una tabla de equivalencias entre  
 * grados Fahrenheit y grados celsius  
 */  
public class Ejemplo1For {  
    public static void main(String[] args) {  
        final int VALOR_INICIAL = 10; // limite inf. tabla  
        final int VALOR_FINAL = 100; // limite sup. tabla  
        final int PASO = 10 ; // incremento  
        int fahrenheit;  
        double celsius;  
        fahrenheit = VALOR_INICIAL;  
        System.out.printf("Fahrenheit \t Celsius \n");  
        for (fahrenheit = VALOR_INICIAL; fahrenheit <= VALOR_FINAL;  
            fahrenheit+= PASO) {  
            celsius = 5*(fahrenheit - 32)/9.0;  
            System.out.printf("%7d \t %8.3f \n", fahrenheit, celsius);  
        }  
    }  
}
```

En las zonas de inicialización e incremento/decremento puede aparecer más de una variable. En ese caso deben ir separadas por comas.

Ejemplo:

```
/*  
 * programa que muestra el valor de a, b y su suma mientras que la suma de  
 * ambas es menor de 10. En cada iteración el valor de a se incrementa en  
 * 1 unidad y el de b en 2  
 */  
public class Ejemplo3For {  
    public static void main(String[] args) {  
        int a, b;  
        for(a = 1, b = 1; a + b < 10; a++, b+=2){  
            System.out.println("a = " + a + " b = " + b + " a + b = " + (a+b));  
        }  
    }  
}
```

La salida de este programa es:

a = 1 b = 1 a + b = 2

$a = 2$ $b = 3$ $a + b = 5$

$a = 3$ $b = 5$ $a + b = 8$

Aunque la instrucción repetitiva for, al igual que las instrucciones while y do- while, se puede utilizar para realizar repeticiones cuando no se sabe a priori el número de pasadas por el bucle, esta instrucción es especialmente indicada para bucles donde se conozca el número de pasadas. Como regla práctica podríamos decir que las instrucciones while y do-while se utilizan generalmente cuando no se conoce a priori el número de pasadas, y la instrucción for se utiliza generalmente cuando sí se conoce el número de pasadas.

Se ha de tener cuidado con escribir el punto y coma (;) después del paréntesis final del bucle for. Un bucle for generalmente no lleva punto y coma final.

Por ejemplo el bucle:

```
int i;  
for (i = 1; i <= 10; i++);  
{  
    System.out.println("Elementos de Programación");  
}
```

no visualiza la frase "Elementos de Programación" 10 veces, ni produce un mensaje de error por parte del compilador.

En realidad lo que sucede es que se visualiza una vez la frase "Elementos de Programación", ya que aquí la sentencia for es una sentencia vacía al terminar con un punto y coma (;).

La sentencia for en este caso hace que i empiece en 1 y acabe en 11 y tras esas iteraciones, se ejecuta la sentencia

```
System.out.println("Elementos de Programación");
```

MATERIAL Y/O EQUIPO A UTILIZAR:

- ☐ Computadora personal
- ☐ Programa JAVA Netbeans instalado
- ☐ Cuaderno de clases, donde están los ejercicios resueltos en clase

EJERCICIOS DIRIGIDOS

EJERCICIO # 01

En la universidad UCV, el rendimiento de un alumno lo clasifican de acuerdo a lo siguiente:

CLASIFICACION	PROMEDIOS
BUENO	si su promedio está entre 16 y 20
REGULAR	si su promedio está entre 11 y 15
DEFICIENTE	si su promedio está entre 6 y 10
PESIMO	si su promedio está entre 0 y 5

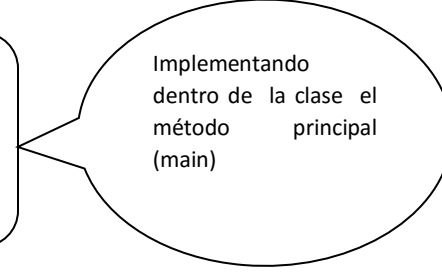
Desarrollar un programa en java que me permita ingresar por teclado 3 notas de un alumno y calcular su respectivo promedio.

Imprimir por pantalla la respectiva clasificación del alumno, en función al rango de promedios.

Solución:

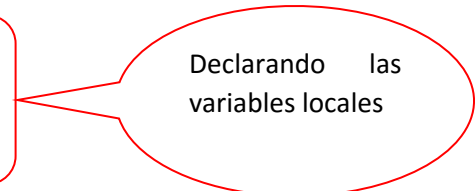
- a) Primeramente tenemos que crear un programa en java de nombre **Ejercicio1_1** que implemente el método **main()**.

```
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
    }
}
```



- b) Dentro del método principal (**main**), tenemos que declarar localmente tres variable de tipo de datos **String** y tres variables de tipo de dato **int**, una variable de tipo de dato **double**.

```
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
        int    n1 , n2 , n3 ;
        String n1Cad , n2Cad , n3Cad;
        double promedio ;
    }
}
```



- c) Después que hemos declarado las variables locales, vamos a implementar código que me permita el ingreso de dato por teclado a través de cajas de dialogo, para esto tenemos que primeramente importar el paquete de nombre **javax.swing.***

Utilizando el método `showInputDialog()` de la clase `JOptionPane`.

```
import javax.swing.*;
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
        int    n1 , n2 , n3 ;
        String n1Cad , n2Cad , n3Cad;
        double promedio ;

        n1Cad=JOptionPane.showInputDialog(null,"Ingrese primera nota","Titulo",1);
        n2Cad=JOptionPane.showInputDialog(null,"Ingrese segunda nota","Titulo",1);
        n3Cad=JOptionPane.showInputDialog(null,"Ingrese tercera nota","Titulo",1);

    }
}
```

1) Importando el paquete que me permite la utilización de la clase

2) Estos código me permiten el ingreso de datos por teclado a través de la clase `JOptionPane` y el método es `showInputDialog()`

- d) A continuación vamos a convertir todas las cadenas numéricas ingresadas por el teclado a valores enteros.

```
import javax.swing.*;
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
        int    n1 , n2 , n3 ;
        String n1Cad , n2Cad , n3Cad;
        double promedio ;

        n1Cad=JOptionPane.showInputDialog(null,"Ingrese primera nota","Titulo",1);
        n2Cad=JOptionPane.showInputDialog(null,"Ingrese segunda nota","Titulo",1);
        n3Cad=JOptionPane.showInputDialog(null,"Ingrese tercera nota","Titulo",1);

        n1=Integer.parseInt(n1Cad);
        n2=Integer.parseInt(n2Cad);
        n3=Integer.parseInt(n3Cad);

    }
}
```

Convirtiendo de una cadena numérica a un valor

- e) Ahora vamos a calcular el promedio de las notas ingresadas por el teclado

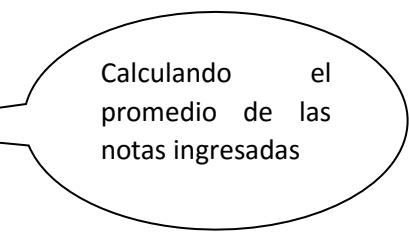
```
import javax.swing.*;
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
        int    n1 , n2 , n3 ;
        String n1Cad , n2Cad , n3Cad;
        double promedio ;

        n1Cad=JOptionPane.showInputDialog(null,"Ingrese primera nota","Titulo",1);
        n2Cad=JOptionPane.showInputDialog(null,"Ingrese segunda nota","Titulo",1);
        n3Cad=JOptionPane.showInputDialog(null,"Ingrese tercera nota","Titulo",1);

        n1=Integer.parseInt(n1Cad);
        n2=Integer.parseInt(n2Cad);
        n3=Integer.parseInt(n3Cad);

        promedio =(double) (n1+n2+n3)/3;

    }
}
```



- f) Para poder evaluar la clasificación del alumno en función a su promedio, tenemos que utilizar un conjunto de estructuras condicionales simples.

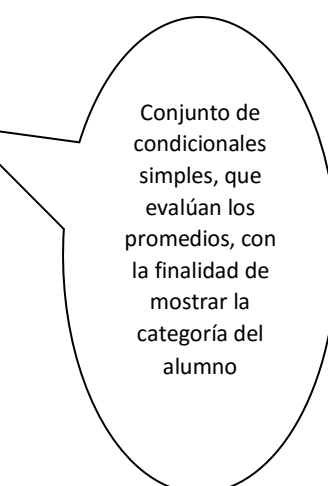
```
import javax.swing.*;
public class Ejercicio1_1
{
    public static void main(String[] args)
    {
        int    n1 , n2 , n3 ;
        String n1Cad , n2Cad , n3Cad;
        double promedio ;

        n1Cad=JOptionPane.showInputDialog(null,"Ingrese primera nota","Titulo",1);
        n2Cad=JOptionPane.showInputDialog(null,"Ingrese segunda nota","Titulo",1);
        n3Cad=JOptionPane.showInputDialog(null,"Ingrese tercera nota","Titulo",1);

        n1=Integer.parseInt(n1Cad);
        n2=Integer.parseInt(n2Cad);
        n3=Integer.parseInt(n3Cad);

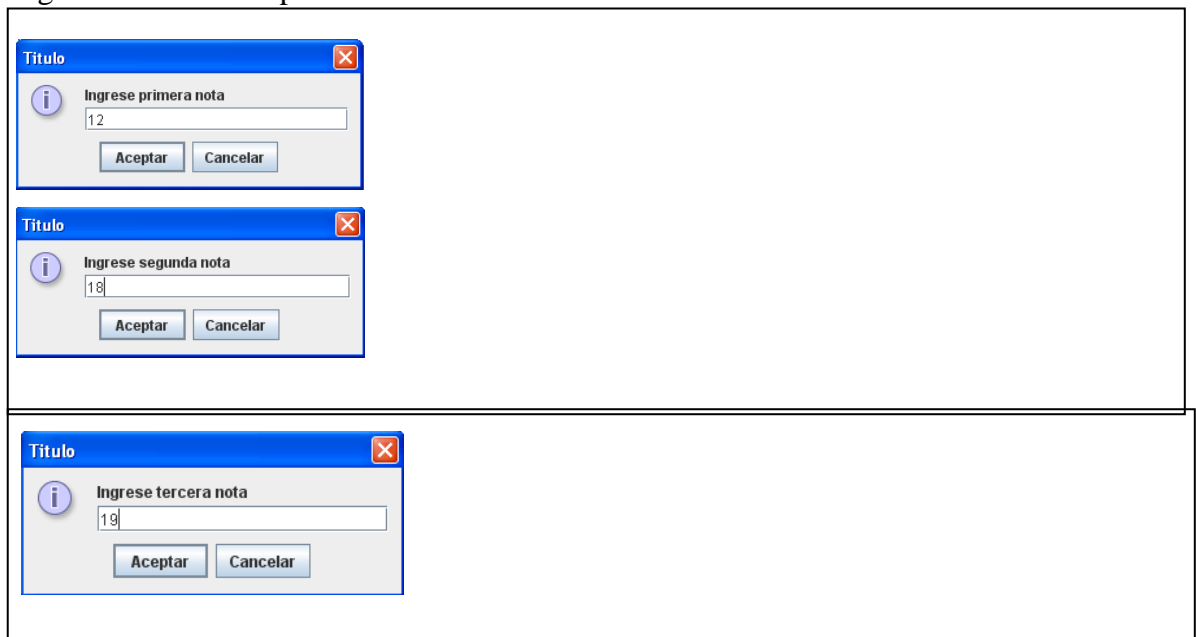
        promedio =(double) (n1+n2+n3)/3;

        if( promedio >=0  && promedio<=5)
        {
            JOptionPane.showMessageDialog(null,"PESIMO");
        }
        if( promedio >=6  && promedio<=10)
        {
            JOptionPane.showMessageDialog(null,"DEFICIENTE");
        }
        if( promedio >=11 && promedio<=15)
        {
            JOptionPane.showMessageDialog(null,"REGULAR");
        }
        if( promedio >=16 && promedio<=20)
        {
            JOptionPane.showMessageDialog(null,"BUENO");
        }
    }
}
```



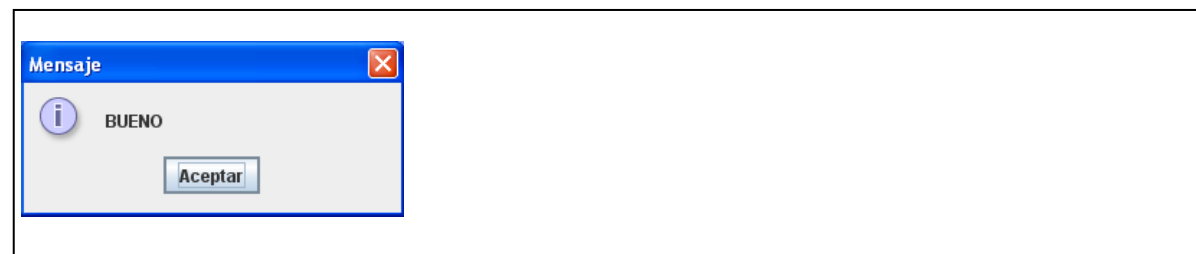
g) Finalmente vamos a compilar, ejecutar el programa.

Ingresando las notas por el teclado



The image shows three sequential dialog boxes, each with a blue title bar and a close button (X) in the top right corner. The first dialog box is titled 'Titulo' and contains an information icon (i) and the text 'Ingrese primera nota'. It has a text input field containing the number '12' and two buttons: 'Aceptar' and 'Cancelar'. The second dialog box is also titled 'Titulo' and contains an information icon (i) and the text 'Ingrese segunda nota'. It has a text input field containing the number '18' and two buttons: 'Aceptar' and 'Cancelar'. The third dialog box is titled 'Titulo' and contains an information icon (i) and the text 'Ingrese tercera nota'. It has a text input field containing the number '19' and two buttons: 'Aceptar' and 'Cancelar'.

Cálculo y salida datos por pantalla



The image shows a single dialog box with a blue title bar and a close button (X) in the top right corner. The title bar is labeled 'Mensaje'. The dialog box contains an information icon (i) and the text 'BUENO'. There is a single button labeled 'Aceptar' at the bottom.

EJERCICIO #02

Una empresa desea adquirir cierta cantidad de unidades de un producto para lo cual cuenta con la oferta de dos proveedores A y B, tal como se explica a continuación:



- Costo unitario igual a s/ 25
- 15 % de descuento para más de 50 unidades adquiridas

Proveedor A



- Costo unitario igual a s/ 27.5
- 10 % de descuento para más de 35 unidades adquiridas

Proveedor B

Desarrollar un programa en Java que determine quién de los proveedores es el más conveniente.

Solución:

- a) Primeramente tenemos que crear un programa en java de nombre **Ejercicio2_1** que implemente el método principal (main).

```
public class Ejercicio2_1
{
    public static void main(String[] args)
    {
    }
}
```

Implementando el
método principal,
dentro de la clase

- b) Dentro del método principal (main), tenemos que declarar localmente , una variable de tipo de dato **entero (cant)** , una variable de tipo de dato **String (cantCad)**, dos variables de tipo de dato real(**descA , descB**).

```
public class Ejercicio2_1
{
    public static void main(String[] args)
    {
        long cant;
        String cantCad;
        double descA , descB;

    }
}
```

Declarando dentro del método principal (main), variables locales.

- c) Ahora que hemos declarado las variables locales, vamos implementar código que me permita ingresar por teclado la **cantidad**, para eso tenemos que importar el paquete **javax.swing.***

Además este paquete importado nos proporciona un método que me permite invocar una caja de dialogo **showInputDialog()** de la clase **JOptionPane**.

La cadena numérica ingresada por el teclado se almacena en la variable **cantCad**, para que luego en la siguiente se convierta en un valor numérico entero a través del método **parseInt** de la clase **Integer**.

```
import javax.swing.*;
public class Ejercicio2_1
{
    public static void main(String[] args)
    {
        long cant;
        String cantCad;
        double descA , descB;

        cantCad=JOptionPane.showInputDialog(null, "Ingrese la Cantidad !!");

        cant=Integer.parseInt(cantCad);

    }
}
```

1) En este código logramos ingresar por el teclado a través de una caja de dialogo, que se guarda en la variable **cantCad**

2) En este código convertimos a un valor entero el valor ingresado por el teclado, y se guarda lo convertido a una variable **cant**

- d) A continuación vamos a calcular el descuento del proveedor A y B y para eso tenemos que validar la cantidad.

Tenemos que implementar una primera condicional doble para calcular el descuento del proveedor A y otra condicional doble para calcular el descuento del proveedor B.

```
import javax.swing.*;
public class Ejercicio2_1
{
    public static void main(String[] args)
    {
        long cant;
        String cantCad;
        double descA , descB;

        cantCad=JOptionPane.showInputDialog(null, "Ingrese la Cantidad !!");

        if(cant>50)
        {
            descA=0.15*cant*25;
        }
        else
        {
            descA=0;
        }

        if(cant>30)
        {
            descB=0.10*cant*27.5;
        }
        else
        {
            descB=0;
        }

    }
}
```

1) Implementando la primera condicional doble, validando la cantidad (**cant>50**) con la finalidad de calcular el descuento del

2) Implementando la segunda condicional doble, validando la cantidad (**cant>30**) con la finalidad de calcular el descuento del

- e) Para consolidar nuestro programa solo faltaría comparar los respectivos descuentos de ambos proveedores.

Vamos a implementar una condicional doble para poder validar el respectivo descuento de ambos proveedores.

```
import javax.swing.*;
public class Ejercicio2_1
{
    public static void main(String[] args)
    {
        long cant;
        String cantCad;
        double descA , descB;

        cantCad=JOptionPane.showInputDialog(null, "Ingrese la Cantidad !!");
        cant=Integer.parseInt(cantCad);
```

```

if(cant>50)
{
    descA=0.15*cant*25;
}
else
{
    descA=0;
}

if(cant>30)
{
    descB=0.10*cant*27.5;
}
else
{
    descB=0;
}

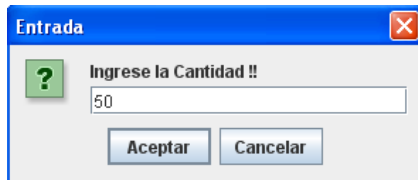
if(descA>descB)
{
    JOptionPane.showMessageDialog(null,"CONVIENE EL PROVEEDOR A");
}
else
{
    JOptionPane.showMessageDialog(null,"CONVIENE EL PROVEEDOR B");
}
}
}

```

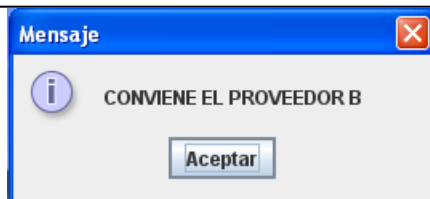
Implementado una condicional doble donde valide los respectivos descuentos de ambos proveedores (**descA>descB**), para mostrar cuál de los dos proveedores es el más

f) Finalmente vamos a compilar, ejecutar el programa.

Ingresando la cantidad por el teclado



Mostrar la salida por pantalla



Guía Práctica 3

1. Diseñe un programa que determine la categoría de un estudiante en base a su promedio, de acuerdo a la siguiente tabla:

Promedio	Categoría
20 - 17	A
16 - 14	B
13 - 12	C
11 - 0	D

2. Una empresa de bienes y raíces ofrece casas de interés social bajo las siguientes condiciones: si el ingreso mensual del comprador es menos de S/2250 la cuota inicial será igual al 15% del costo de la casa y el resto se distribuirá en 120 cuotas mensuales. Si el ingreso mensual del comprador es mayor o igual a S/.2250 la cuota inicial será igual al 30% del costo de la casa y el resto se distribuirá en 75 cuotas mensuales. Diseñe un programa que permita determinar cuanto debe pagar un comprador por concepto de cuota inicial y cuanto por cada cuota mensual.

3. Una tienda de venta de polos ha establecido porcentajes de descuento, indicados a continuación, de acuerdo a las características de la prenda: Tipo de algodón (Nacional, Importado) y Detalle de la prenda (Sin estampado, Con estampado)

	Detalle de la prenda	
	S	C
Tipo de algodón		
Nacional	3	5
Importado	5	3

Calcular el importe a pagar por una prenda

4. Calcular el pago de un obrero que trabaja al destajo. El pago que recibe el obrero por cada unidad producida depende de su categoría y del tipo de producto que produce, como se muestra en el siguiente cuadro:

Categoría	Tarifa (S/. x unidad)	
	Tejas	Losetas
A	2.50	2.00
B	2.00	1.50
C	1.50	1.00

Así mismo, el obrero recibe una bonificación especial de acuerdo a la cantidad que produce:

Unidades Producidas	Bonificación (%)
1 - 250	0.00
251 - 500	50.00
501 - 1000	100.00
1001 - mas	150.00

Además del total de ingresos se descuenta 75 soles por seguro.

5. La tarifas de SEDAPAL para el consumo doméstico de agua potable es la siguiente:

Consumo (m3)	Tarifa (S/.)
0 – 20	0.80
21- 30	1.10
31 – 50	1.55
51 – más	2.13

Las tarifas se aplican a los excesos sobre los límites establecidos. Así para un consumo de 38 m3, los primeros 30m3 se pagan a S/1.10 por m3 y los 8 restantes a S/.1.55 por m3. las tarifas no incluyen el 19% de IGV. El importe total a pagar es la suma de una pensión básica de S/.3.40, mas el importe por el consumo del mes y más el importe del impuesto. Diseñe un programa que determine todos los importes correspondientes a la factura de un cliente. Considere que el consumo es una cantidad entera.

6. Una empresa desea adquirir cierta cantidad de unidades de un producto para lo cual cuenta con la oferta de dos proveedores A y B, tal como se explica a continuación:
- Proveedor A: Costo unitario igual a S/./25.00 y 15% de descuento para mas de 50 unidades adquiridas
 - Proveedor B: Costo unitario igual a S/./27.50 y 10% de descuento para mas de 35 unidades adquiridas

Calcular cuanto pagaría por una determinada cantidad de unidades al Proveedor A y cuanto al Proveedor B y poder recomendar cual proveedor le conviene.

7. Una compañía de seguros ofrece a sus clientes cuatro tipos de seguro de sepelio:

Tipo	Máximo número de Personas	Pago mensual (S/.)
A	8	40
B	6	30
C	4	20
D	2	10

Si el cliente asegura a más personas de la indicadas en el cuadro anterior tendrá que pagar S/./8.00 mensuales por cada persona adicional si es que el seguro es de tipo A o B, y S/./5.00 mensuales por cada persona adicional si es que el seguro es de tipo C o D. Calcular el monto anual que tiene que pagar un determinado cliente.

8. Un almacén ofrece descuentos de prendas de vestir de acuerdo a la siguiente tabla:

Un cliente solo puede adquirir un solo tipo de prenda. Si el cliente compra más de las prendas indicadas deberá pagar por cada prenda adicional, S/./50.00 por sacos o pantalones y S/./58.00 por zapatos o correas. Determinar el importe total a pagar.

Prendas	Precio	Máximo de prendas
Sacos	45	7
Pantalones	48	6
Zapatos	53	5
Correas	56	4

Bibliografía:

- THOMAS WU C. Introducción a la programación orientada a objetos con Java. 1ª Edición. España. McGraw-Hill Interamericana de España. 2008. 214-324pp. ISBN: 978-0-07-352339-2
- LEOBARDO LOPEZ. Román. Metodología de la programación orientada a objetos. 1ª Edición. México. Alafomega grupo editor de México. 2006. 257-342pp ISBN: 970-15-1173-5
- HERBERT SHILDT. JAVA 2 v5.0. España. Ediciones Anaya multimedia.2005. 131-138pp, 834pp ISBN: 131-138-1865-3.

REFERENCIAS EN INTERNET

- <http://gcoronelc.blogspot.pe>
- <http://gcoronelc.blogspot.pe/2016/11/programando-pensando-en-servicios-parte.html>
- <http://gcoronelc.blogspot.pe/2016/11/prog-pensando-en-servicios-parte-2.html>
- <http://gcoronelc.blogspot.pe/2016/06/separata-java-orientado-objetos.html>
- <http://gcoronelc.blogspot.pe/p/java.html>
- <http://gcoronelc.blogspot.pe/2017/01/java-fundamentos-01-introduccion.html>
- <http://gcoronelc.blogspot.pe/2013/09/java-poo-leccion-01.html>
- <http://gcoronelc.blogspot.pe/2013/09/java-poo-leccion-02.html>