	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**	
CICLO: I/2021	**GUIA DE LABORATORIO #01**	
	Nombre de la Práctica:	Introducción Java
	MATERIA:	Desarrollo de Software para Móviles

II. INTRODUCCION TEORICA

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems y actualmente propiedad de Oracle.

Características de Java:

1. Lenguaje Simple: "Se lo conoce como lenguaje simple porque viene de la misma estructura de c y c++; ya que c++ fue un referente para la creación de java por eso utiliza determinadas características de c++ y se han eliminado otras."

2. Orientado a Objetos.

3. Multihilos: Java tiene una facilidad de cumplir varias funciones al mismo tiempo, gracias a su función de multihilos ya que por cada hilo que el programa tenga se ejecutaran en tiempo real muchas funciones al mismo tiempo.

JRE

El JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma

JDK

Es el acrónimo de "Java Development Kit", es decir Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

API

Define tres plataformas en un intento por cubrir distintos entornos de aplicación. Así, ha distribuido muchas de sus API (Application Program Interface) de forma que pertenezcan a cada una de las plataformas:

1. **Java ME** (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.
2. **Java SE** (Java Platform, Standard Edition) o J2SE — para entornos de gama media y

estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.

3. **Java EE** (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.

TIPOS DE DATOS

Tipo	Tamaño	Rango
byte	8	-128 a 127
short	16	-32,768 a 32,767
int	32	-2,147,483,648 a 2,147,483,647
long	64	-9,223,372,036,854,775,808L a 9,223,372,036,854,775,807L
float	32	+/- 3.4E+38F (6-7 dígitos importantes)
double	64	+/- 1.8E+308 (15 dígitos importantes)
char	16	Conjunto de caracteres Unicode ISO
boolean	1	verdadero o falso

Valores por defecto en una variable sin inicializar.

- Object referencia un valor null (es decir no referencia ningún objeto)
- byte, short, int, long valor por defecto es 0
- float, double valor por defecto es 0.0
- boolean valor por defecto es false

Identificadores

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores:

- Todos los identificadores han de comenzar con una letra, el carácter subrayado (_) o el carácter dollar ($).
- Puede incluir, pero no comenzar por un número
- No puede incluir el carácter espacio en blanco
- Distingue entre letras mayúsculas y minúsculas
- No se pueden utilizar las palabras reservadas como identificadores

Además de estas restricciones, hay ciertas convenciones que hacen que el programa sea más legible, pero que no afectan a la ejecución del programa. La primera y fundamental es la de encontrar un nombre que sea significativo, de modo que el programa sea lo más legible posible. El tiempo que se pretende ahorrar eligiendo nombres cortos y poco significativos se pierde con creces cuando se revisa el programa después de cierto tiempo.

Operadores

En una condición deben disponerse únicamente variables, valores constantes y operadores relacionales.

Operadores Relacionales:

Operador	Uso	Devuelve verdadero si
>	ope1 > ope2	ope1 es mayor que ope2
>=	ope1 >= ope2	ope1 es mayor o igual que ope2
<	ope1 < ope2	ope1 es menor que ope2
<=	ope1 <= ope2	ope1 es menor o igual que ope2
= =	ope1 = = ope2	ope1 y ope2 son iguales
! =	ope1 != ope2	ope1 y ope2 son distintos

Operadores Matemáticos:

Operador	Uso	Descripción
+	ope1 + ope2	Suma ope1 y ope2 (*)
-	ope1 - ope2	Resta ope1 de ope2
*	ope1 * ope2	Multiplica ope1 y ope2
/	ope1 / ope2	Divide ope1 por ope2
%	ope1 % ope2	Obtiene el modulo de dividir ope1 por ope2
++	ope++	Incrementa ope en 1; evalúa el valor antes de incrementar
++	++ope	Incrementa ope en 1; evalúa el valor después de incrementar
--	ope--	Decrementa ope en 1; evalúa el valor antes de incrementar
--	--ope	Decrementa ope en 1; evalúa el valor después de incrementar

Hay que tener en cuenta que al disponer una condición debemos seleccionar que operador relacional se adapta a la pregunta. Ejemplos:

Se ingresa un número multiplicarlo por 10 si es distinto a 0. (!=)
Se ingresan dos números mostrar una advertencia si son iguales. (==)

Operadores lógicos:

Operador	Uso	Devuelve verdadero si
&&	ope1 && ope2	ope1 y ope2 son verdaderos
| |	ope1 | | ope2	uno de los dos es verdadero
!	! ope	ope es falso (niega ope)

Nota: Los operadores lógicos siempre devuelven un valor booleano.

Qué es una Clase

Una clase es una plantilla que define la forma de un objeto. Especifica los datos y el código que operará en esos datos. Java usa una especificación de clase para construir objetos. Los objetos son instancias de una clase. Por lo tanto, **una clase es esencialmente un conjunto de planes que especifican cómo construir un objeto**.

Aquí se muestra una forma general simplificada de una definición de clase:

```
class NombreClase {
  //Declarar variables de instancia
  tipo var1;
  tipo var2;
  //..

  //Declarar métodos

  tipo metodo1(parámetros) {
    //Cuerpo del método
  }

  tipo metodo2(parámetros) {
    //Cuerpo del método
  }
}
```

Definición de una Clase

```
class Vehiculo {
  int pasajeros; //números de pasajeros
  int capacidad; //capacidad del combustible en galones
  int mpg; //combustible consumido en millas por galon
}

//Esta clase declara un objeto de tipo Vehiculo

class DemoVehiculo {

  public static void main(String[] args) {
    Vehiculo minivan = new Vehiculo();
    int rango;

    //asignando valores a los campos de minivan
```

```
   minivan.pasajeros = 9;
   minivan.capacidad = 15;
   minivan.mpg = 20;

   //Calcular el rango asumiendo un tanque lleno
   rango = minivan.capacidad * minivan.mpg;

   System.out.println("La Minivan puede llevar " + minivan.pasajeros + " pasajeros con un rango de
" + rango + " millas");
  }
}
```

Conversión de tipos de datos en java

En Java es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado. Este proceso se denomina "conversión", "moldeado" o "tipado" y es algo que debemos manejar con cuidado pues un mal uso de la conversión de tipos es frecuente que dé lugar a errores. De forma general trataremos de atenernos a la norma de que "en las conversiones debe evitarse la pérdida de información". En la siguiente tabla vemos conversiones que son seguras por no suponer pérdida de información.

TIPO ORIGEN	TIPO DESTINO
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	Double

String a INT

```
String numCadena = "1";
int numEntero = Integer.parseInt(numCadena);
```

INT a String

```
int numEntero = 4;
String numCadena= String.valueOf(numEntero);
O
String numCadena= Integer.toString(numEntero);
```

IV. PROCEDIMIENTO

- Para realizar la práctica se puede utilizar una herramienta online o la de su preferencia : https://repl.it/

Lectura de datos de entrada

Para leer la entrada de consola, lo primero que se hace es construir un Scanner que este asociado al flujo de entrada estándar System.in

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

Y ahora se utilizan los distintos métodos de la clase Scanner para leer la entrada. Por ejemplo, el método nextLine lee una línea de entrada

```
System.out.print(“¿Cómo te llamas? ”);
String nombre = in.nextLine();
```

En este caso, se utiliza el método nextLine porque la entrada podría contener espacios. Para leer una sola palabra (delimitada por espacios de separación), se hace la llamada.

```
String nombreDepila=in.next();
```

Para leer un entero utilice el método nextInt.

```
System.out.print(“¿Cuántos años tienes?”);
Int edad = int.nextInt();
```

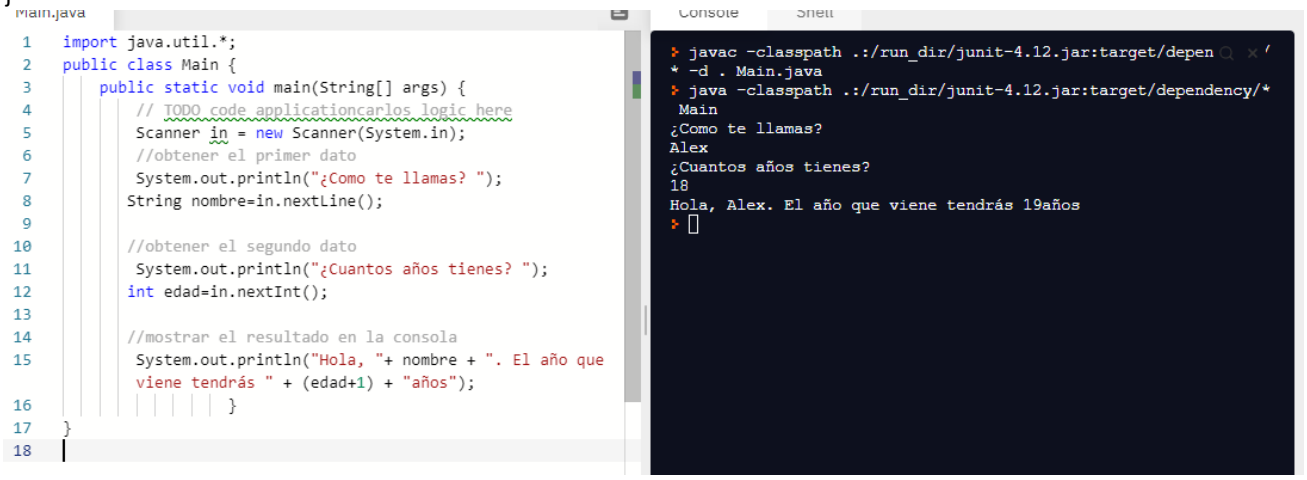
Ejemplo #01

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      // TODO code applicationcarlos logic here
      Scanner in = new Scanner(System.in);
      //obtener el primer dato
      System.out.println("¿Como te llamas? ");
      String nombre = in .nextLine();

      //obtener el segundo dato
      System.out.println("¿Cuantos años tienes? ");
      int edad = in .nextInt();

      //mostrar el resultado en la consola
      System.out.println("Hola, " + nombre + ". El año que viene tendrás " + (edad + 1) + "años");
   }
```

}



Estructura de programación secuencial

Cuando en un problema sólo participan operaciones, entradas y salidas se la denomina una estructura secuencial.

Ejemplo #01

Realizar la carga de dos números enteros por teclado e imprimir su suma y su producto.

Tenemos dos entradas num1 y num2 (recordar cuáles son los nombres de variables correctas), dos operaciones: realización de la suma y del producto de los valores ingresados y dos salidas, que son los resultados de la suma y el producto de los valores ingresados. En el símbolo de impresión podemos indicar una o más salidas, eso queda a criterio del programador, lo mismo para indicar las entradas por teclado.

```
import java.util.Scanner;
public class SumaProductoNumeros {
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      int num1, num2, suma, producto;
      System.out.print("Ingrese primer valor:");
      num1 = teclado.nextInt();

      System.out.print("Ingrese segundo valor");
      num2 = teclado.nextInt();
      suma = num1 + num2;
      producto = num1 * num2;
      System.out.print("La suma de los dos valores es:");
      System.out.println(suma);
```

```
      System.out.print("El producto de los dos valores es:");
      System.out.println(producto);
   }
}
```

Estructuras condicionales simples y compuestas

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión aparecen las estructuras condicionales.

Ejemplo #02

Ingresar el sueldo de una persona, si supera los 3000 pesos mostrar un mensaje en pantalla indicando que debe abonar impuestos.

Podemos observar lo siguiente: Siempre se hace la carga del sueldo, pero si el sueldo que ingresamos supera 3000 pesos se mostrará por pantalla el mensaje "Esta persona debe abonar impuestos", en caso que la persona cobre 3000 o menos no aparece nada por pantalla.

```
import java.util.Scanner;
public class EstructuraCondicionalSimple1 {
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      float sueldo;
      System.out.print("Ingrese el sueldo:");
      sueldo = teclado.nextFloat();
      if (sueldo > 3000) {
         System.out.println("Esta persona debe abonar impuestos");
      }
   }
}
```

Ejemplo #03

Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:
Si el promedio es >=7 mostrar "Promocionado".
Si el promedio es >=4 y <7 mostrar "Regular".
Si el promedio es <4 mostrar "Reprobado".

```
import java.util.Scanner;
public class EstructuraCondicionalAnidada1 {
```

```
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      int nota1, nota2, nota3;
      System.out.print("Ingrese primer nota:");
      nota1 = teclado.nextInt();
      System.out.print("Ingrese segunda nota:");
      nota2 = teclado.nextInt();
      System.out.print("Ingrese tercer nota:");
      nota3 = teclado.nextInt();
      int promedio = (nota1 + nota2 + nota3) / 3;
      if (promedio >= 7) {
         System.out.print("Promocionado");
      } else {
         if (promedio >= 4) {
            System.out.print("Regular");
         } else {
            System.out.print("Reprobado");
         }
      }
   }
}
```

Estructura repetitiva while

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.
Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Ejemplo #04

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno. Ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.

```
import java.util.Scanner;
public class EstructuraRepetitivaWhile2 {
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      int n, x;
```

```
      System.out.print("Ingrese el valor final:");
      n = teclado.nextInt();
      x = 1;
      while (x <= n) {
         System.out.print(x);
         System.out.print(" - ");
         x = x + 1;
      }
   }
}
```

Estructura repetitiva for

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura while. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones. En general, la estructura for se usa en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones. Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita. Veremos, sin embargo, que en el lenguaje Java la estructura for puede usarse en cualquier situación repetitiva, porque en última instancia no es otra cosa que una estructura while generalizada.

Ejemplo #05

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio. Este problema ya lo desarrollamos, lo resolveremos empleando la estructura for.

```
import java.util.Scanner;
public class EstructuraRepetitivaFor {
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      int suma, f, valor, promedio;
      suma = 0;
      for (f = 1; f <= 10; f++) {
         System.out.print("Ingrese valor:");
         valor = teclado.nextInt();
         suma = suma + valor;
      }
      System.out.print("La suma es:");
      System.out.println(suma);
```

```
      promedio = suma / 10;
      System.out.print("El promedio es:");
      System.out.print(promedio);
   }
}
```

Estructura repetitiva do while

La estructura do while es otra estructura repetitiva, la cual ejecuta al menos una vez su bloque repetitivo, a diferencia del while o del for que podían no ejecutar el bloque. Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo. La condición de la estructura está abajo del bloque a repetir, a diferencia del while o del for que está en la parte superior.

Ejemplo #06

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

```
import java.util.Scanner;
public class EstructuraRepetitivaDoWhile2 {
   public static void main(String[] ar) {
      Scanner teclado = new Scanner(System.in);
      int suma, cant, valor, promedio;
      suma = 0;
      cant = 0;
      do {
         System.out.print("Ingrese un valor (0 para finalizar):");
         valor = teclado.nextInt();
         if (valor != 0) {
            suma = suma + valor;
            cant++;
         }
      } while (valor != 0);
      if (cant != 0) {
         promedio = suma / cant;
         System.out.print("El promedio de los valores ingresados es:");
         System.out.print(promedio);
      } else {
         System.out.print("No se ingresaron valores.");
      }
   }
}
```

Las Clases

Ejemplo #07

Confeccionar un programa que permita cargar los nombres de 5 personas y su mail, luego implementar los siguientes métodos:

a) Mostrar por pantalla los datos.
b) Consulta del mail ingresando su nombre.
c) Mostrar los emails que no tienen el carácter @.

```
import java.util.Scanner;

public class Cadena5 {
  private Scanner teclado;
  private String[] nombres;
  private String[] mail;

  public Cadena5() {
    teclado = new Scanner(System. in );
    nombres = new String[5];
    mail = new String[5];
    for (int f = 0; f < nombres.length; f++) {
      System.out.print("Ingrese nombre:");
      nombres[f] = teclado.nextLine();
      System.out.print("Ingrese mail");
      mail[f] = teclado.nextLine();
    }
  }

  public void listar() {
    for (int f = 0; f < nombres.length; f++) {
      System.out.println(nombres[f] + " - " + mail[f]);
    }
  }

  public void consultaMail() {
    String aux;
    System.out.print("Ingrese el nombre de la persona:");
    aux = teclado.nextLine();
    boolean existe = false;
    for (int f = 0; f < nombres.length; f++) {
      if (aux.equals(nombres[f])) {
        System.out.println("Mail de la persona:" + mail[f]);
        existe = true;
```

```
    }
   }
   if (existe == false) {
    System.out.println("No existe una persona con ese nombre.");
   }
  }

  public void sinArroba() {
   for (int f = 0; f < mail.length; f++) {
    boolean tiene = false;
    for (int k = 0; k < mail[f].length(); k++) {
     if (mail[f].charAt(k) == '@') {
      tiene = true;
     }
    }
    if (tiene == false) {
     System.out.println(mail[f] + " no tiene @");
    }
   }
  }

  public static void main(String[] ar) {
   Cadena5 cad = new Cadena5();
   cad.listar();
   cad.consultaMail();
   cad.sinArroba();
  }
}
```

V. DISCUSION DE RESULTADOS

1. Crear un programa en consola que me permita saber si dos números son divisibles entre sí, para saber si un número es divisible por otro se tiene que obtener el modulo y si este es cero entonces este número es divisible por el otro.

2. Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

3. Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.

4. Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano. Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.

VII. BIBLIOGRAFIA

- Aprendiendo Java 2 en 21 Días Lemay, Laura

- Cómo Programar en Java Deitel, Harvey M.