

Programación Aplicada I  
Sección B  
Guía de trabajos prácticos  
**Parte II**

Prof. Javier Borrás

Analista Universitario en Sistemas  
Informáticos





# Índice

## Contents

Modalidad de trabajo .....	2
Control de Flujos .....	3
Condicionales (flujos de decisión) .....	3
Estructuras repetitivas.....	4
Estructuras repetitivas anidadas .....	9
.....	9
Entrada de teclado y archivos.....	10
Ingreso de datos y operaciones con String .....	13
Ejercicios de Arrays.....	17
Bibliografía y documentación.....	21
Sobre el autor:.....	21
Contacto .....	21

## Modalidad de trabajo

Durante el dictado de clases, se explicarán conceptos teóricos-prácticos en lo relativo a la sintaxis y uso de Java.

Utilizaremos las convenciones de nombrado descritas en el apunte teórico.

Se recomienda que el alumno pueda instalar la versión de JAVA SDK y el entorno de desarrollo (IDE) en su propia computadora, para ello hay que descargar la versión de java <http://goo.gl/i3aaGx> y la de Eclipse <http://goo.gl/mDDbzS>

Si te resulta complicado, puedes buscar algún video tutorial en youtube o seguir este: <https://youtu.be/xCMZ-EVW2sl>





## Control de Flujos

### Condicionales (flujos de decisión)

#### 1. Verificar Pasa Falla (if-else)

Escribir un programa llamado **VerificarPasaFalla** que imprima: “Pasó” si la variable *marca* es mayor igual a 50, caso contrario, imprimir: “falla”. Probar con distintos valores

```
public class VerificarPasaFalla{ // guardar como "VerificarPasaFalla.java"
    public static void main(String[] args) {
        int marca = 49;           // inicializar la variable marca aquí!
        System.out.println("El valor de marca es: " + marca);
        if ( ..... ) {
            System.out.println( ..... );
        } else {
            System.out.println( ..... );
        }
    }
}
```

#### 2. Verificar Par Impar (if-else)

Escribir un programa llamado **VerificarParImpar**, que imprima “Número Impar” si la variable del tipo **int numero** es impar, sino imprimir “Número Par”. Probar con distintos valores.



*$n$  es un número par si  $(n \% 2)$  es cero*

```
public class VerificarParImpar{ // guardar como "VerificarParImpar.java"
    public static void main(String[] args) {
        int numero = 49;         // poner el valor aquí!
        System.out.println("El valor de numero es " + numero);
        if ( ..... ) {
            System.out.println( ..... );
        } else {
            System.out.println( ..... );
        }
    }
}
```

#### 3. Imprimir Números en palabras (If anidados, switch-case)

Escribir un programa llamado **ImprimirNumeroEnPalabras**, que imprima “Uno”, “Dos”, “Tres”, ... , “Nueve”; “Otro” si la variable, del tipo **int numero** es 1,2,3,...,9; u otro respectivamente.

Escribir el mismo programa utilizando dos algoritmos:

- Utilizando if anidados.
- Utilizando la sentencia “switch-case”

```
public class ImprimirNumeroEnPalabras{
    public static void main(String[] args) {
        int numero= 5;

        //a) usar if anidado
        if (numero== 1) {
            System.out.println("UNO");
        } else if (.....) {
            .....
        }
    }
}
```





```
    } else if (.....) {  
        .....  
        .....  
    } else {  
        .....  
    }  
  
    //b) Usando switch-case  
    switch(numero) {  
        case 1: System.out.println("UNO"); break;  
        case 2: .....  
        .....  
        .....  
        default: System.out.println("OTRO");  
    }  
}  
}
```

De manera similar, escribir un programa llamado **ImprimirDiasEnPalabras**, que imprima: “lunes”, “martes”, “miércoles”,..., “domingo”, si la variable **día** es 0,1,2,...6 respectivamente, caso contrario imprimir “No es un día válido”.

## Estructuras repetitivas

### 4. Suma y promedio (loop)

Escribir un programa llamado **SumaYpromedio** para generar la suma acumulada de: 1,2,3,4,5,..., hasta lo que indique la variable **int limiteSuperior**, por ejemplo 100.

También calcular y mostrar el promedio de dicha suma.

La consola deberá mostrar algo similar a esto:

```
La suma es 5050  
El promedio es 50.5
```

```
public class SumaYpromedio{  
    public static void main (String[] args) {  
        int sum = 0;           // guardar la suma acumuladas, inicializando a 0  
        double promedio;      // promedio en double o float  
        int limiteInferior = 1; // el límite inferior del valor a sumar  
        int limiteSuperior = 100; // el límite superior del valor a sumar  
  
        for (int numero= limiteInferior ; numero <= limiteSuperior ; numero++) { // bucle for  
            sum += numero;           // equivalente a "sum = sum + numero"  
        }  
        // calcular el promedio  
        .....  
        // Imprimir suma y promedio  
        .....  
    }  
}
```

- a. Modificar el programa anterior para usar una estructura repetitiva del tipo “**while-do**” en lugar del bucle for (no borrar lo que ya está hecho)

```
int numero = limiteInferior;  
int sum = 0;  
while (numero <= limiteSuperior) {  
    sum += numero;  
}
```



```
    ++numero;  
}
```

- b. Modificar el programa para usar un bucle del tipo “**do-while**” (no borrar lo que ya está hecho)

```
int numero = limiteInferior;  
int sum = 0;  
do {  
    sum += numero;  
    ++numero;  
} while (numero <= limitesuperior);
```

- c. Modificar el programa para que sume desde 111 hasta 8899 y calcular el promedio. Introducir una variable del tipo int llamada *contador* para contar los números en el rango especificado. (no borrar lo que ya está hecho)

```
int count = 0;    // contador del número dentro de rangos, inicializado en 0  
for (...; ...; ...) {  
    .....  
    ++count;  
}
```

- d. Modificar el programa para sumar sólo los números impares desde el 1 al 100, y calcular el promedio. (no borrar lo que ya está hecho)
- e. Modificar el programa para sumar aquellos números del 1 al 100 que sean divisibles por 7 y calcular el promedio. (no borrar lo que ya está hecho)
- f. Modificar el programa para encontrar la “suma de los cuadrados” de todos los números del 1 al 100, por ejemplo  $1*1 + 2*2 + 3*3 + \dots + 100*100$  (no borrar lo que ya está hecho)

## 5. Producto del uno al n.

Escribir un programa llamado **Producto1AN** para calcular el producto de los enteros del 1 al 10 (por ejemplo:  $1 * 2 * 3 * \dots * 10$ )

```
public class Producto1AN {  
    public static void main(String[] args) {  
        int resultado = 1;  
        for (;;) {  
            .....  
            System.out.print("i vale: " + i);  
        }  
        System.out.print(resultado);  
    }  
}
```



*Intenta calcular el producto desde 1 hasta 11, 1 hasta 12, 1 hasta 13, 1 hasta 14.  
¿Qué sucede? Investigar si es necesario.*

## 6. Series Armónicas (Loop - [Serie Armónica](#))

Escribir un programa llamdo **SerieArmonica** que calcule la suma de una serie armónica finita, cuyo máximo denominador será 5000.

Para recordar: la serie armónica es una función matemática con la forma:



$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \dots$$

Para nuestro ejercicio, el resultado será la suma de las fracciones del tipo  $1/n$  en donde el máximo valor de  $n$  será 5000. Se deberá plantear el algoritmo de manera tal que se acumule el resultado obtenido en DOS variables, una que acumulará recorriendo los denominadores desde 1 hasta 5000 y otra que acumulará los valores recorriendo los denominadores desde 5000 hasta 1, al final mostrar los resultados y compararlos. Además, mostrar el promedio.

```
public class SerieArmonica{ // guardar como "SerieArmonica.java"
    public static void main (String[] args) {
        int maxDenominador = 50000;
        double sumMenorAmayor = 0.0; // sum de menor a mayor
        double sumMayorAmenor= 0.0; // sum de mayor a menor
        // for-loop para la suma de menor a mayor
        for (int denominador = 1; denominador <= maxDenominador; ++denominador) {
            .....
            // recordar que: int/int retorna un int.
        }
        // for-loop para la suma de mayor a menor
        .....
        //mostrar los resultados
        .....
    }
}
```

## 7. Cálculo de Pi (bucles y condiciones)

Escribir un programa, llamado **CalculoDePi** para aproximarse al valor de  $\pi$  utilizando la [Serie de Leibniz](#) que se expresa de la siguiente forma:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4}.$$

por lo que tenemos:

$$\pi = 4 \times \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \dots \right)$$

Se deberá decidir el criterio de finalización del cálculo, que puede ser en base los números de términos usados o la magnitud de un término de adición.

También se deberá mostrar el resultado de la aproximación obtenida y el valor de Math.PI.

Además, se deberá mostrar el porcentaje de precisión de Math.PI sobre nuestro cálculo obtenido.



el JDK tiene el valor de  $\pi$  en una constante del tipo **double** llamada **Math.PI**.  
Agregar el término para sumar si el módulo del denominador por 4 es 1, y restarlo, si el mismo es 3 (denom % 4)

```
double sum = 0;
int maxDenom = 10000000;
for (int denom = 1; ..... ; denom = denom + 2) {
    if (denom % 4 == 1) {
        sum += .....;
    } else if (denom % 4 == 3) {
        sum -= .....;
    } else {
        System.out.println("Esto se volvió loco!!");
    }
}
```





## 8. Piedra Papel y Tijera (bucle & condiciones)

Escribir un programa, llamado **PiedraPapelTijera** que imprima los números del 1 al 110, mostrando 11 números por línea.

El programa deberá imprimir:

“Piedra” en lugar de los números que son múltiplos de 3

“Papel” en lugar de los números que son múltiplos de 5 y

“Tijera” para aquellos números múltiplos de 7, además,

imprimir “PiedraPapel” para aquellos números que sean a su vez divisibles por 3 y 5 simultáneamente,

“PapelTijera” para los divisibles entre 5 y 7,

“PiedraTijera” para los que son divisibles entre 3 y 7.

La salida del programa, deberá mostrar algo como esto:

```
1 2 Piedra 4 Papel Piedra Tijera 8 Piedra Papel 11
Piedra 13 Tijera PiedraPapel 16 17 Piedra 19 Papel PiedraTijera 22
23 Piedra Papel 26 Piedra Tijera 29 PiedraPapel 31 32 Piedra
.....
```

```
public class PiedraPapelTijera{ // Guardar como "PiedraPapelTijera.java"
    public static void main(String[] args) {
        int indiceInferior= 1;
        int indiceSuperior= 110;
        for (int numero= indiceInferior; numero <= indiceSuperior; ++numero) {
            // Imprimir "Piedra" si es divisible por 3
            if (.....) {
                System.out.print("Piedra");
            }
            // Imprimir "Papel" si es divisible por 5
            if (.....) {
                System.out.print(.....);
            }
            // Imprimir "Tijera" si el número es divisible por 7
            .....
            // imprimir el número si no es divisible por 3, 5 y 7
            if (.....) {
                .....
            }
            // imprimir una nueva línea si el número es divisible por 11; sino, imprimir
            un espacio
            if (.....) {
                System.out.println();
            }
        }
    }
}
```

## 9. Sucesión de Fibonacci (while)

Escribir un programa llamado **Fibonacci**, para mostrar los primeros 20 números de la [sucesión de Fibonacci](#):

La sucesión de Fibonacci se explica cómo “cada término es la suma de los dos anteriores”.

Matemáticamente hablando, la función se escribe:

**$F(n) = F(n-1) + F(n-2)$  y  $F(1) = F(2) = 1$**  dicho de otra manera, cada número de la sucesión, es igual a la suma de la serie para el mismo número - 1 con la suma de la función de la sucesión del mismo número - 2. Excepto cuando  **$n = 1$  y  $n = 2$** , cuyo resultado siempre es **1**.

Por ejemplo:

Si  **$n = 3$**

**$F(3) = F(3-1) + F(3-2) \Rightarrow F(3) = F(2) + F(1) \Rightarrow F(3) = 1 + 1$**  (por definición  $F(2) = F(1) = 1$ )





Entonces: **F(3) = 2**

Si **n = 4**

$F(4) = F(4-1) + F(4-2) \Rightarrow F(4) = F(3) + F(2) \Rightarrow F(4) = 2 + 1$

Entonces: **F(4) = 3**

Si **n = 5**

$F(5) = F(5-1) + F(5-2) \Rightarrow F(5) = F(4) + F(3) \Rightarrow F(5) = 3 + 2$

Entonces: **F(5) = 5**

Resumiendo:

**F(1) = 1**

**F(2) = 1**

**F(3) = F(2) + F(1)**

**F(4) = F(3) + F(2)**

**F(5) = F(4) + F(3)**

**F(6) = F(5) + F(4)**

**F(7) = F(6) + F(5)**

También calcular el promedio.

La salida de la consola, deberá mostrar algo como esto:

```
Los primeros 20 números de Fibonacci son:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
el promedio es: 885.5
```

```
public class Fibonacci {
    public static void main (String args[]) {
        int n = 3;           // el índice n para F(n), comenzando con n=3
        int fn;              // F(n) que se calculará
        int fnMenos1 = 1;    // F(n-1), inicialización para F(2) = 1 (por definición)
        int fnMenos2 = 1;    // F(n-2), inicialización para F(1) = 1 (Por definición)
        int nMax = 20;       // valor máximo de n, inclusive
        int sum = fnMenos1 + fnMenos2;
        double promedio;

        System.out.println("Los primeros " + nMax + " números de Fibonacci son:");
        .....

        while (n <= nMax) {
            // calcular F(n), imprimirlo y agregar a ""sum""
            .....
            // ajustar el índice n e intercambiar los números
            .....
        }
        // calcular y mostrar el promedio (=sum/nMax)
        .....
    }
}
```






Estructuras repetitivas anidadas

10. Tabla Cuadrada (for anidados):

Escribir un programa llamado **TablaCuadrada**, que muestre el patrón nXn (n=5) que se muestra abajo, usando dos bucles for anidados.  
La consola deberá mostrar algo así:

```
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

```
System.out.print("# "); // imprime # y un espacio, sin salto de línea
System.out.println(); // imprime una nueva línea
public class TablaCuadrada{ // guardar como "TablaCuadrada.java"
    public static void main (String[] args) {
        int dimension = 5; // tamaño de la tabla
        for (int fila = 1; .....; ..... ) {
            for (int columna = 1; .....; ..... ) {
                .....
            }
            .....
        }
    }
}
```

 Se deberá utilizar sólo dos sentencias para la salida

11. Tablas de multiplicar (for anidados)

Escribir un programa llamado **TablaDeMultiplicar** que muestre las tablas de multiplicación del 1 al 9 utilizando dos for anidados. La consola deberá mostrar algo como esto:

*	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81





## Entrada de teclado y archivos

### 12. Lector de Teclado (Ingreso por teclado)

Escribir un programa llamado **LectorDeTeclado** para pedir al usuario el ingreso de datos del tipo int, del tipo double y del tipo String respectivamente.

La salida debería verse de esta forma (los ingresos se muestran en **negrita**)

```
Ingrese un entero: 12
Ingrese un número de coma flotante: 33.44
Ingrese su nombre: Nicolás
Hola Nicolás, la suma de 12 y 34.44, es: 45.44
```

```
import java.util.Scanner;    // !importante esta referencia es para el manejo del
teclado
public class LectorDeTeclado{
    public static void main(String[] args) {
        int num1;
        double num2;
        String nombre;
        double sum;
        // Inicializar un tipo Scanner llamado "in" para leer el teclado(System.in)
        Scanner in = new Scanner(System.in);
        System.out.print("Ingrese un entero: ");
        num1 = in.nextInt();    // usar nextInt() para leer un tipo int
        System.out.print("Ingrese un número de coma flotante: ");
        num2 = in.nextDouble(); // usar nextDouble() para leer un tipo double
        System.out.print("Ingrese su nombre: ");
        name = in.next();       // usar next() para leer un tipo String
        // Mostrar mensaje
        .....
    }
}
```

### 13. Lector de Archivos

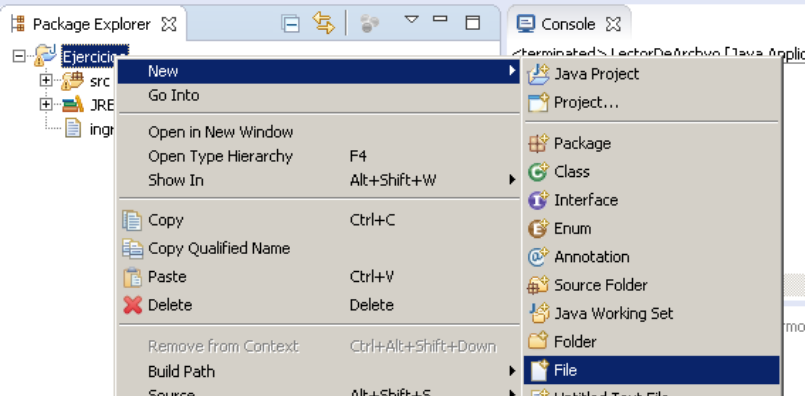
Escribir un programa llamado **LectorDeArchivo** para leer datos del tipo int, double y String, desde un archivo de texto llamado “entrada.txt”, y mostrar en consola lo siguiente:

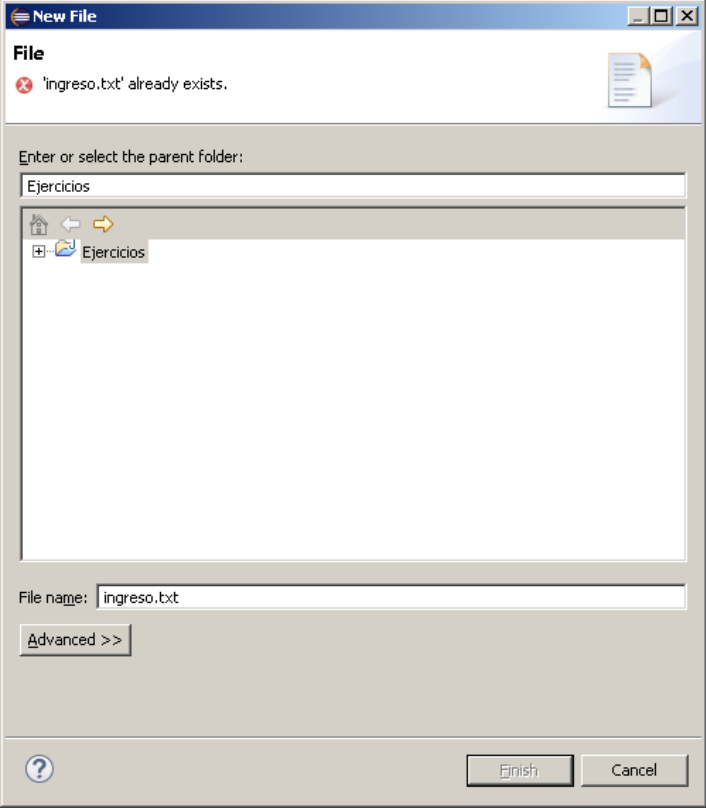
```
Ingrese un entero: 12
Ingrese un número de coma flotante: 33.44
Ingrese su nombre: Nicolás
Hola Nicolás, la suma de 12 y 34.44, es: 45.44
```

**Nota:** Necesita crear un archivo simple de texto, llamado "ingreso.txt" (en Eclipse, click derecho en el proyecto, New => File

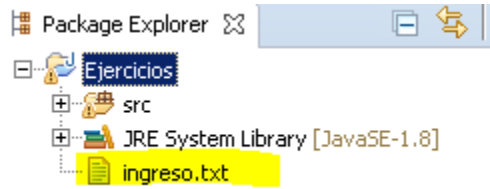








**!IMPORTANTE:** el archivo debe quedar en la raíz del proyecto:



Luego ingresar el siguiente contenido en el archivo “ingreso.txt”

12  
33,44  
Nicolás

**Nota:** ingresar 33 , (coma) 44

```
import java.util.Scanner;    // Necesario para el manejo del teclado
import java.io.File;        // Necesario para el manejo de archivos
import java.io.FileNotFoundException; // Necesario para operaciones con archivos
public class LectorDeArchvo{
    public static void main(String[] args)
        throws FileNotFoundException { // Necesario para el manejo de archivos
        int num1;
        double num2;
        String nombre;
```





```
double sum;
// Inicializar Scanner para la lectura de archivos
Scanner in = new Scanner(new File("ingreso.txt"));
num1 = in.nextInt();    // usar nextInt() para leer un int
num2 = in.nextDouble(); // usar nextDouble() para leer un double
name = in.next();       // usar next() para leer un String
in.close();             // Importante!!! cierra y libera el recurso
// Mostrar resultado
.....
}
```

#### 14. Cálculo del perímetro de la circunferencia: (Ingreso teclado)

Escribir un programa llamado **CalculoDeCircunferencia** que pregunte por el ingreso del radio (double) y computar el área y la longitud de una circunferencia.

La salida debe mostrar algo como esto:

```
Ingresar el radio: 1.2
el área es 4.5239
la longitud es 7.5398223686155035
```



*Tener en cuenta que el área de la circunferencia es*

$$A = \pi \cdot r^2$$

*la longitud de la circunferencia es*

$$\ell = \pi \cdot 2r$$

**OBLIGATORIO:** Obtener el valor de  $\pi$  del algoritmo utilizado en el ejercicio CalculoDePi (ejercicio número 7 Cálculo de Pi (bucles y condiciones)), no se puede usar la constante *Math.PI*





## Ingreso de datos y operaciones con String

### 15. String al reverso (String - for)

Escribir un programa llamado **StringReverso** que solicite el ingreso de un dato del tipo **String** e imprimirlo en forma inversa. La consola deberá mostrar algo como esto:

Ingrese un String: abcdef  
El reverso del String "abcdef" es"fedcba".



Para la variable llamada *inStr*, se puede usar *inStr.length()* para obtener la cantidad de caracteres que conforman la cadena; y *inStr.charAt(indice)* para obtener el caracter en la posición indicada a través de “índice”, donde el índice empieza en 0 (cero)

Un “string” en inglés quiere decir “cadena”, entonces el tipo *String* es una cadena, cuyos eslabones son caracteres, pues cualquier caracter ocupa el mismo “tamaño” (4 bytes para el estándar [unicode](#)) fijo en la memoria, funcionando de manera distintas a los tipos numéricos en donde 1 ocupa 1 bit, 4 ocupa 3 bit, etc., si de alguna manera pudiéramos “representar” un string en memoria, a lo mejor podríamos ver algo así:

H	O	L	A
0	1	2	3

El tipo *String*, al tener características particulares, se puede interpretar como una especie de “vector” de datos del tipo “char”, por eso hay funciones especiales, como *.length()* que indica la cantidad de char que tiene un string o *.charAt(3)* que en el caso del ejemplo del cuadrado retornaría el valor del caracter en la posición 3, que es el char “A”.

En el string al tener cada caracter una y sólo una representación binaria, podemos afirmar que el caracter **A** no es lo mismo que el caracter **a** (diferenciación entre mayúsculas y minúsculas, esto en informática se conoce como **Case Sensitive**)

```
import java.util.Scanner;
public class StringReverso{
    public static void main(String[] args) {
        String inStr;           // entrada del String
        int inStrLen;           // cantidad de caracteres en el String
        Scanner in = new Scanner(System.in);
        System.out.print("Ingrese un string: ");
        inStr = in.next();       // usar next() para leer String
        inStrLen = inStr.length();
        // usar inStr.charAt(indice)para extraer el caracter correspondiente al índice
        indicado
        .....
    }
}
```



## 16. Teclado Telefónico (Scanner, switch-case)

En los teclados telefónicos, se encuentra el alfabeto inglés mapeado en los números de la siguiente forma:



Escribir un programa, llamado **TecladoTelefonico** que solicite al usuario el ingreso de los string:  
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z.  
y convertirlos a una secuencia de dígitos numéricos del pad del teclado.  
Por ejemplo, si el usuario ingresa gtf, la consola deberá mostrar 483



*Puede usar `in.next().toLowerCase()` para convertir todos los ingresos en minúsculas y reducir el problema de la sensibilidad a los caracteres mayúsculas y minúsculas.*

## 17. Palíndromos

*Un palíndromo es una palabra, número o frase que se lee igual hacia adelante que hacia atrás (también suele decirse capicúa)*

Escribir una programa llamado **PruebaPalindromo** que pida al usuario que ingrese una palabra o frase, e indique si la misma es o no un palíndromo. Por ejemplo "Neuquen", "La ruta natural", "1221".



*leer la palabra y convertirla a minúscula utilizando `in.nextLine().toLowerCase()`, recorrer la palabra ingresada y eliminar los espacios en blanco, luego recorrerla en forma inversa y compararla. También usar la sentencia switch-case para reemplazar los caracteres acentuados á,é,í,ó,ú*

## 18. BinarioAdecimal

Escribir un programa llamado **BinarioAdecimal**, que convierta un ingreso de String en formato binario (unos y ceros) y escribir su equivalente en base decimal.

La salida de pantalla, deberá mostrar algo como esto:

Ingrese un string binario: 11101001

El equivalente decimal al número binario "11101001" es 233

Ingrese un string binario: 1234

Error: String binario no válido "1234"



para un número binario n-bit  $b_{n-1}b_{n-2}...b_1b_0$ ,  $b_i \in \{0,1\}$ , el equivalente en número decimal es:  $b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + ... + b_1 \times 2^1 + b_0 \times 2^0$

Por ejemplo: el número binario: **11101001** se convertiría:

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

1	1	1	0	1	0	0	1
---	---	---	---	---	---	---	---

```
import java.util.Scanner;
public class BinarioAdecimal{
    public static void main(String[] args) {
        String binStr;    // ingreso de un string binario
        int binStrLen;    // cantidad de caracteres del string ingresado
        int dec = 0;      // equivalente en decimales
        char binChar;     // cada caracter individual en el string binario

        Scanner in = new Scanner(System.in);

        // Leer el ingreso del string binario
        .....

        // Convertir el string binario en decimal
        .....
    }
}
```

```
binStr          : 1 0 1 1 1 0 0 1
charAt(idx)     : 0 1 2 3 4 5 6 7
Math.pow(2, order) : 7 6 5 4 3 2 1 0
binStr.length() = 8
idx + order = binStr.length() - 1
```



Se puede usar el método del JDK `Math.pow(x, y)` para calcular  $x$  elevado a  $y$  ( $x^y$ ), este método recibe dos tipos `double` como argumento y retorna un `double` como resultado de la operación. Se tiene que convertir el resultado al tipo `int` (entero)

Para convertir un **char** (de dígitos del '0' al '9') a un tipo **int** (0 al 9), simplemente puede restar por el carácter '0', por ejemplo '9' - '0' = 9 (del tipo `int`)

### 19. Decimal a Binario

Escribir un programa llamado **DecimalABinario**, para convertir un número en base decimal, a su equivalente en binario. Uno de los métodos para el cambio de base de decimal a binario, es el "Método de la división sucesiva por 2", el cual se obtiene un número binario dado un decimal entero, dividirlo, sucesivamente entre 2 hasta obtener un cociente más pequeño que el 2, el número que estamos buscando, lo compondrá el último cociente y los restos que se hayan ido obtenido, tomados en orden inverso.

Entonces, el resultado sería: **1100**

Por ejemplo:

12	<u>2</u>		
0	6	<u>2</u>	
←	0	3	<u>2</u>
	←	1	1
		←	←



se puede usar el método `String.valueOf(resto);` para convertir un número en una cadena.





```
public class DecimalAbinario {  
  
    public static void main(String[] args) {  
        int numeroEntero;  
        //ingreso de la variable numero entero  
        System.out.println("Ingrese un número entero" );  
        ...  
        int cociente = numeroEntero;  
        int resto;  
        String binarioString = "";  
        String binarioReverso = "";  
        while(.... >= 2){  
            resto = ...% 2;  
            cociente = ... / 2;  
            //acumular el resultado, convirtiendo el numero resto en el  
            tipo String  
            binarioString += String.valueOf(resto);  
        }  
        // usar el algoritmo para obtener el reverso del string  
        binarioReverso = ...;  
        System.out.println("El numero en binario es:" );  
        System.out.println(binarioReverso );  
    }  
}
```

## 20. HexadecimalAdecimal

Escribir un programa llamado **HexAdec** para convertir un valor en hexadecimal ingresado como String a su equivalente en base decimal. La salida de la consola, deberá mostrar algo como esto:

```
Ingrese un string Hexadecimal: 1a  
El número hexadecimal "1a" en base decimal es 26  
  
Ingrese un string Hexadecimal: 1y3  
Error: String Hexadecimal no válido: "1y3"
```

Para un decimal de n-dígitos:  $h_{n-1}h_{n-2}...h_1h_0$ ,  $h_i \in \{0, ..., 9, A, ..., F\}$ , el equivalente en base decimal es:  $h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + ... + h_1 \times 16^1 + h_0 \times 16^0$ .

No se necesita una maraña de if anidados para los 16 valores (o 22 si consideramos las mayúsculas y minúsculas). Extraer cada caracter individual del string hexadecimal, por ejemplo: si el char c está entre '0' y '9', se puede obtener un entero utilizando el desplazamiento  $\text{int } c - '0'$ . Ahora si c está entre 'a' y 'f', se puede obtener el desplazamiento entero haciendo  $c - 'a' + 10$ .

```
String hexStr;  
char hexChar;  
.....  
hexChar = hexStr.charAt(i);  
.....  
if (hexChar >= '0' && hexChar <= '9') {  
    ... (hexChar-'0') ...  
    ...  
} else if (hexChar >= 'a' && hexChar <= 'f') {    // minúscula  
    ... (hexChar-'a'+10) ...  
    ...  
} else if (hexChar >= 'A' && hexChar <= 'F') {    // mayuscula  
    ... (hexChar-'A'+10) ...  
    ...  
} else {  
    System.out.println("Error: .....");  
    System.exit(1);    // Salir del programa  
}
```







## Ejercicios de Arrays

### 21. Inicializar

Escribir un programa llamado **Inicializar** que declare e inicialice un array `miArray[]` que permita 1000 elementos del tipo `int`. Imprimir el último elemento utilizando `miArray.length` y ver que sucede, escribe brevemente lo que sucedió y por qué, y por último calcula el último índice y asignalo a la variable

```
int ultimoIndice = miArray.length ...;
```

Respuesta:

### 22. Intercambio

Escribir un programa llamado **Intercambio** que tenga el siguiente array: {2,1,4,5,6,7,8,6}, e intercambiar el primero elemento con el último. Reemplazar el elemento 4 por 0.

### 23. MayorMenor

Escribir un programa llamado **MayorMenor** que contenga el array siguiente: {1,2,5,4,7,6,6,8,9,0,1,-1} recorrerlo mediante un bucle for, e indicar si el elemento actual es mayor ó menor o igual que el elemento siguiente.

```
public static void main(String[] args) {  
    int[] miArray = {2,1,4,5,6,7,8,6};  
    int hasta = ...;  
    System.out.println("El tamaño del array es: " + miArray.length + " Se  
debe recorrer hasta: " + hasta);  
    for(int i = 0; i < hasta; i++)  
    {  
        System.out.println("i vale: " + i + " El elemento actual es: " +  
miArray[i] + " El elemento siguiente es: " + miArray[i +1]);  
        if(...)  
        {  
            System.out.println(" El elemento actual es menor o igual  
que el siguiente ");  
        }  
        else  
        {  
            ...  
        }  
    }  
}
```

### 24. Desplazamiento de un array

Escribir un programa llamado **DesplazamientoArray** que reemplace el elemento actual con el elemento siguiente hasta llegar al último elemento y reemplazarlo por el primero, de manera que, si tengo un array {1,2,4,7,8} el mismo quede {2,4,7,8,1}, utilizando un bucle for.

```
...  
int[] miArray = { 1, 2, 3 };  
int primerElemento = miArray[0];  
for (int i = 1; i < ...; i++) {  
    miArray[...] = miArray[i];  
}  
miArray[miArray.length - 1] = primerElemento;  
for (int indice = 0; indice < miArray.length; indice++) {  
    System.out.print(...);  
}  
...
```



## 25. IntercambioDePares

Escribir un programa llamado **IntercambioDePares** que tenga el siguiente array: {2,1,4,5,6,7,8,6}, e intercambiar el primero elemento con el segundo, el tercero con el cuarto y así sucesivamente hasta obtener el siguiente array: {1,2,5,4,7,6,6,8}

## 26. Ingreso de datos

Escribir un programa llamado **IngresoDeDatos** que solicite al usuario el tamaño del array para luego ingresar cada uno de los elementos del tipo int.



*Debemos importar los paquetes `java.io.BufferedReader` y `java.io.InputStreamReader`.*

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class IngresoDeDatos{
    public static void main(String[] args) {
        /*creación del objeto para leer por teclado*/
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        /*ingreso del tamaño de arreglos*/
        System.out.print("\n Ingrese Numero de Datos a Ingresar : ");
        int tam = Integer.parseInt(in.readLine());
        /*creación del arreglo*/
        int arreglo[] = new int[tam];
        System.out.println();
        /*lectura del arreglo*/
        for (int i = 0 ;...) {
            System.out.print("Elemento " + i + 1 + " : ");
            arreglo[i] = Integer.parseInt(in.readLine());
        }
    }
}
```

## 27. Calcular Promedios

Escribir un programa llamado **CalcularPromedios** que solicite al usuario el ingreso del número de estudiantes, cuyo valor lo guardará en la variable numEstudiantes. Luego preguntará para cada uno de los estudiantes la nota de la evaluación (enteros del 1 al 10) si el usuario ingresa un valor fuera del rango del 1 al 10 se deberá volver a pedir que ingrese el dato hasta que se haya ingresado un dato válido.

Luego deberá imprimir el promedio general.

La salida de la consola deberá mostrar algo como esto:

```
Ingrese el número de estudiantes: 3
Ingrese la nota para el estudiante 1: 5
Ingrese la nota para el estudiante 2: 11
Nota invalida, intente nuevamente
Ingrese la nota para el estudiante 2: 0
Nota invalida, intente nuevamente
Ingrese la nota para el estudiante 2: 10
Ingrese la nota para el estudiante 3: 2
El promedio es: 5.666
```



## 28. Ordenamiento de Vector

Escribir un programa llamado **OrdenamientoDeVector**, que dado un vector cualesquiera, compuesto por números del tipo double, lo ordene de menor a mayor mediante el uso del algoritmo del “Método de la burbuja” el cual usa dos for anidados. Imprimir el resultado antes y después.

```
for(int i = 0; i < arreglo.length - 1; i++)
{
    for(int j = 0; j < arreglo.length - 1; j++)
    {
        if (arreglo[j] > arreglo[j + 1])
        {
            int tmp = arreglo[j+1];
            arreglo[j+1] = arreglo[j];
            arreglo[j] = tmp;
        }
    }
}
//Imprimir el resultado
for(....)
{
    System.out.print(arreglo[i]+"\\n");
}
```

## 29. Hexadecimal A Binario

Escribir un programa llamado **HexAbinario** que solicite el ingreso de un hexadecimal string, y lo convierta en su equivalente binario

La salida de la consola, deberá mostrar algo como esto:

```
Ingrese un Hexadecimal string: 1abc
El equivalente binario al hexadecimal "1abc" es 0001 1010 1011 1100
```



Usar un array de 16 strings binarios correspondientes a los números decimales del '0' al 'F', como se ve en este ejemplo:

```
String[] hexBits = {"0000", "0001", "0010", "0011",
                    "0100", "0101", "0110", "0111",
                    "1000", "1001", "1010", "1011",
                    "1100", "1101", "1110", "1111"};
```

## 30. TecladoTelefonico 2

Escribir un programa llamado **TecladoTelefonico2** que modifique el ejercicio 16 (TecladoTelefonico) usando arrays.

## 31. Imprimir Números en palabras 2

Escribir un programa llamado **ImprimirNumeroEnPalabras2** que modifique el ejercicio número 3 **ImprimirNumeroEnPalabras**, para utilizar arrays en lugar de switch.

## 32. Imprimir Dias En Palabras 2

Escribir un programa llamado **ImprimirDiasEnPalabras2** similar al ejercicio número 3 **ImprimirDiasEnPalabras** pero usando arrays para manipular los días





---

### 33. Piedra Papel y Tijera 2

Escribir un programa llamado **PiedraPapelTijera2** que utilice vectores para agregar los valores Piedra, papel y tijera y acceder a los mismos mediante sus índices.

---

### 34. Tablas de multiplicar 2

Escribir un programa llamado **TablasDemultiplicar2** que modifique el ejercicio 11 **TablasDemultiplicar**, para usar un array multidimensional para almacenar los valores.

---



---

## Bibliografía y documentación

Documentación oficial de Oracle: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>

Nanyang Technological University:

<https://www3.ntu.edu.sg/home/ehchua/programming/index.html>

Princeton University

<http://introcs.cs.princeton.edu/java/home/>

## Sobre el autor:

Javier E. Borrás es Analista de Sistemas de Informática, graduado en el año 2010 en la Escuela Superior de Comercio Manuel Belgrano - UNC - Reconocido con medalla de honor por la mejor calificación de la carrera, además obtuvo mención especial del Consejo Profesional de Ciencias Informáticas de la Provincia de Córdoba por el mismo mérito.

Fué ayudante de cátedra de la materia Programación Aplicada II durante 2 años consecutivos.

Dictó un curso en la ESCMB - UNC de larga duración "Desarrollo integral de una aplicación de escritorio bajo entorno .NET" de Junio a Diciembre del año 2012.

También ha desarrollado tareas de coaching para tesis de grado y pregrado en distintas instituciones.

Involucrado en la TI desde el año 2000, pasando por todas las áreas, desde soporte técnico, mantenimiento de servidor, programación, hasta el liderazgo de proyectos de software, ha trabajado en diversas empresas TI en la provincia de Córdoba, desde PyMES a multinacionales.

Desarrolla software principalmente sobre el stack de tecnologías Microsoft, especialmente en .Net, aunque ha participado en proyectos de desarrollos Java, Mobile, Robótica y automatización, e inclusive en desarrollos de hardware propietario.

Como docente ha participado en distintas conferencias, como así también en proyectos de capacitación en tecnologías Microsoft orientadas a Web. Además de Auxiliar Docente en las carreras de Ingeniería en electrónica en la UTN y en la ESCMB.

También es especialista en Comercio Electrónico, Marketing Digital y Redes Sociales.

Actualmente se desempeña como Líder Técnico de un equipo de desarrollo de aplicaciones web con tecnologías Microsoft .Net en [Globant](https://www.globant.com), una de las empresas de software más importantes de América Latina.

## Contacto



[javierborras@yahoo.es](mailto:javierborras@yahoo.es)



[@javierborras](https://twitter.com/javierborras)



<https://www.linkedin.com/in/javierborras>