Programación en Java

Variables

La estructura de las variables es la siguiente `tipoDato nombreVariable = valor;`, un ejemplo de una variable es `float altura = 1.77f;`

- **Tipos de datos primitivos**
  - Números enteros

Nombre	Tamaño	Rango
long	64	-9223372036854775808 a 9223372036854775807
int	32	-2147483648 a 2147483647
short	16	-32768 a 32767
byte	8	-128 a 127

    Cuando tienes una variable de tipo `long` debes colocar una L al final del número para que lo tome como un long. Puede ser minúscula o mayúscula.

  - Número decimales o reales

Nombre	Tamaño	Rango
double	64	4.9e-324 a 1.8e+308
float	32	1.4e-045 a 3.4e+038

    Cuando tienes una variable de tipo `float` debes colocar una F al final del número para que lo tome como un float. Puede ser minúscula o mayúscula.

  - Carácteres

    Hay otro tipo de dato que es el dato `char` el cuál permite almacenar una letra, un número, un símbolo o un signo. Se escribe `char nombreVariable = 'valor';`

  - Boolean

    Permite alamacenar dos datos *true* y *false*. Y permite evaluar si es verdadero o falso. Se escribe `bolean`.

- **Tipos de datos no primitivos**

  Los datos no primitivos pueden almacenar el tipo de dato `null`, estos datos permiten también acceder a los métodos, un ejemplo de esto es `Integer nombreVariable = null;` este tipo de dato lo veremos más afondo cuando tratemos el tema de programación orientada a objetos.

- Cadenas de texto

  Permite almacenar texto escribiendo `String nombreVariable = "texto";`

- Constantes

  Al agregarle `final` antes de declarar la variable a dicha variable no se le podrá cambiar el valor. Ejemplo: `final int nombreVariable = 12;`

Entrada de datos

Por consola

Para poder introducir datos y guardarlos en una variable debemos importar una librería de java y esto se hace así: `import java.util.Scanner;` esto se coloca antes de declarar la clase, al principio del código. Después debemos declarar un objeto de tipo `Scanner` y asignarle el valor de `new Scanner(System.in);` después declaramos una variable del tipo que queremos recibir, por ejemplo, `int numero;` después debemos preguntar al usuario algo y esto lo hacemos con `System.out.println()`, por ejemplo `System.out.println("Hola, ingresa un número")` y para almacenar la respuesta en la variable escribimos `nombreVariable = nombreScanner.nextInt();` en el caso de que el dato sea entero. Un ejemplo de esto es:

```
import java.util.Scanner;

public class entradaDatos{
 public static void main(String[] args){

   Scanner entrada = new Scanner(System.in);
   int numero;

   System.out.println("Hola, ingresa el número que desees");
   numero = entrada.nextInt();

   System.out.println(numero);
 }
}
```

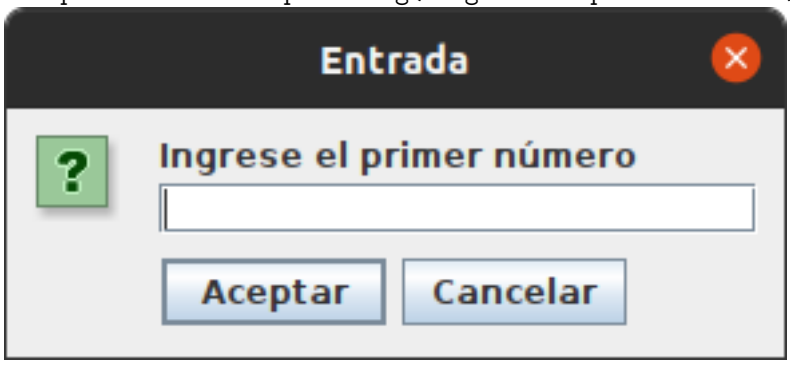
En el caso de que el tipo de dato que se quiera recibir sea *float* se cambia `nextInt` por `nextFloat` si al ingresar el dato te da un error es porque el decimal debe ser con una , y no con un .

Para *double* se debe colocar `nextDouble` y para *String* debes usar `nextLine` y para *char* usas `next().charAt(0)`

Por ventanas emergentes mediante Joption

Si queremos que los datos a ingresar y los datos ingresados se muestren en ventanas y no en la consola debemos usar otra librería llamada JOptionPane.

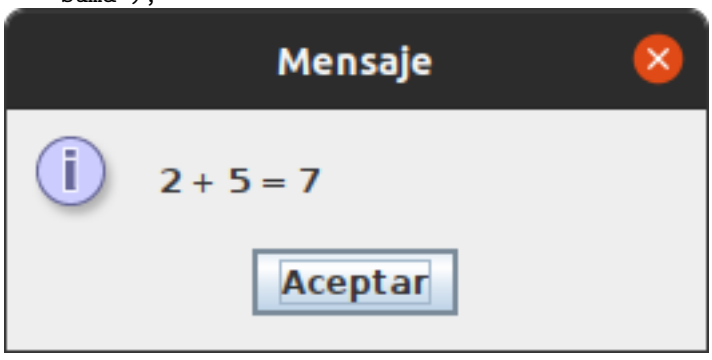
Para empezar importamos la librería con `import javax.swing.JOptionPane;` y para almacenar el valor que ingrese el usuario debemos colocar `nombreVariable = JOptionPane.showInputDialog("Ingrese el primer número")`.



esto muestra una ventana con el mensaje y permite ingresar datos, estos datos serán almacenados en la variable. El dato que devuelve esta variable ese de tipo String por lo que si necesitas almacenar un dato entero, char o double debes colocar lo siguiente:

- **int**: `Integer.parseInt(JOptionPane.showInputDialog("texto"))`
- **char**: `JOptionPane.showInputDialog("texto").charAt(0)`
- **Double**: `Double.parsedouble(JOptionPane.showInputDialog("texto"))`

Y para mostrar un mensaje y que el usuario no pueda ingresar datos entonces se coloca `JOptionPane.showMessageDialog(null, n1 + " + " + n2 + " = " + suma );`



A continuación un ejemplo de una calculadora muy simple:

```
// Calculadora de dos números por JOptionPane

import javax.swing.JOptionPane;
```

```
public class entradaJoption {

  public static void main(String[] args){

    int n1;
    int n2;
    int suma;
    int resta;
    int multiplicacion;
    int division;
    int residuo;

    n1 = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el primer número"));
    n2 = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el segundo número"));

    suma = n1 + n2;
    resta = n1 - n2;
    multiplicacion = n1 * n2;
    division = n1 / n2;
    residuo = n1 % n2;

    JOptionPane.showMessageDialog(null, n1 + " + " + n2 + " = " + suma);
    JOptionPane.showMessageDialog(null, n1 + " - " + n2 + " = " + resta);
    JOptionPane.showMessageDialog(null, n1 + " * " + n2 + " = " + multiplicacion);
    JOptionPane.showMessageDialog(null, n1 + " / " + n2 + " = " + division);
    JOptionPane.showMessageDialog(null, "El residuo de la división es: " + residuo);
  }
}
```

Operadores

Los operadores son los signos que utilizamos para sumar, restar, multiplicar, dividir, comparar, etc. Algunos de estos operadores te los adelanté en el ejercicio pasado cuando realizamos la mini calculadora. Estos operadores son:

Asignación

`=` se utiliza para asignar un valor a una variable, este ya lo hemos utilizado anteriormente.

Aritméticos

- **Suma**: `+` se utiliza para sumar números y también para concatenar palabras como por ejemplo `"hola" + "mundo"` el resultado de esto será “hola mundo”.

- **Resta**: `-` se utiliza para restar números.
- **Multiplicación**: `*` se utiliza para multiplicar números.
- **División**: `/` se utiliza para dividir números.
- **Residuo**: `%` también llamado mod se utiliza para ver el residuo de una división. Un ejemplo de esto es `5 % 2` el resultado de esto sería 1 porque al dividir 5 entre 2 el resultado es 2 y sobra 1.

Estos operadores aritméticos también se pueden combinar con el operador de asignación. Esto nos sirve para realizar una operación a la misma variable, por ejemplo, `variable += 10` esto le sumará a la variable 10, esto se puede hacer con el operador aritmético que queramos.

Incremento y decremento

Estos operadores se utilizan para sumar o restar 1 a la variable, por ejemplo, `variable++` esto le suma a la variable 1 y en el caso de la resta es lo mismo pero cambiando los `++` por `--`. Pero si imprimes `variable++` te va a devolver el valor sin sumar porque esta operación la realiza después de la variable para hacer que primero sume el valor lo debemos colocar antes así `+variable` esto es igual con la resta.

Clase Math

Hay una clase en java que nos permite realizar operaciones matemáticas y trigonométricas. Esta clase tiene muchos parámetros y algunos son, aquí veremos algunos. Esta clase se usa de la siguiente manera:

Primero declaramos una variable que almacene lo que queremos hacer, por ejemplo `double raiz =` ahora le asinamos a la varible el parámetro de la clase que en este caso es de un raiz cuadrada y el parámetro es: `Math.sqrt(9);` aquí estamos calculando la raiz cudrada de 9. Si imprimimos el valor de esta variable nos daría 3.0 .La mayoría de parámetros de la clase Math devuelven valores `double` por lo tanto las variables que lo almacenan deben ser `double` si quieres que la variable sea entera entonces colocas antes del parámetro `(int)` y de esta manera el valor que devuelve es entero, por ejemplo, `int raiz = (int)Math.sqrt(9);` y de esta manera el valor será 3 sin el 0.

Otros parámetros son:

- `Math.pow(2, 8)`: Este parámetro potencia una base al número indicado, en este caso 2 a la 8, lo cual devuelve 256.
- `Math.round(12.8f)`: Este parámetro redondea el número decimal que le demos a un número entero. Se pueden ingresar double o floats y van a devolver enteros.
- `Math.random()`: Esto devuelve un número aleatorio entre 0 y 1, aunque pienses que no sirve para nada le puedes colocar rango multiplicando el

parámetro por el rango que quieres, por ejemplo, `Math.random() * 20` esto devolvería un número aleatorio entre 0 y 20.

Hay muchos parámetros más que los puedes averiguar buscando *parámetros de la clase math en java*. También si escribes `Math.` aparecerán los posibles parámetros y una breve explicación de cada uno.

Ejercicios con lo visto anteriormente

1. Pida al usuario los datos personales y muestre con Scanner y System o JOptionPane una oración con estos datos.
2. Crear una calculadora que ingrese dos número y realice la suma, resta, multiplicación y división de estos dos números.
3. Genera dos números aleatoriamente y con estos realiza la potenciación, los números deben ser float o double.

Condicionales

Los condicionales son estructuras de control que nos permiten realizar las acciones dependiendo de una respuesta o de un valor.

IF

Empecemos por el condicional *if*, este condicional es de los más usados y por lo tanto de los más útiles; la estructura de este condicional es la siguiente:

```
if (condicion){

  código

}else{

  código

}
```

Este condicional evalua la condición (variable) que le pasemos y si esa condición es verdadera ejecuta el primer código y si no, ejecuta el segundo código; vamos a ver un ejemplo y que quede más claro.

```
import javax.swing.JOptionPane;

public class condicional {

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

```
    boolean casado = true;

    if (casado){

      JOptionPane.showMessageDialog(null, "Felicidades, eres casado");

    }else{

      JOptionPane.showMessageDialog(null, "Felicidades, no estás amarrado");

    }

  }
}
```

En este caso casado es verdadero entonces ejecuta el primer código que mostraría lo siguiente:

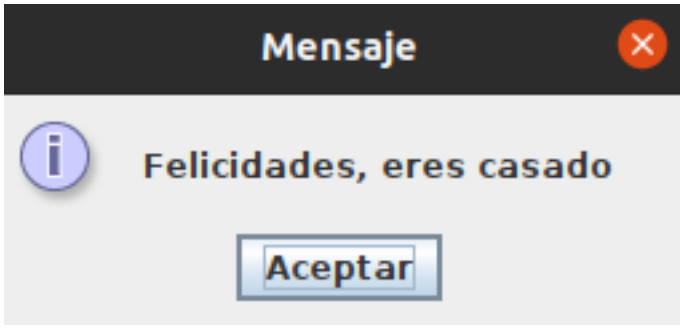


Figure 1: first if

Y en el caso de que cambiemos el valor de casado por *false* nos mostraría lo siguiente:

Estos condicionales pueden tener más condicionales adentro y se pueden repetir agrgándole después del *else* un *if* y así crea la estructura del *if* nuevamente; te quiero comentar que la parte del *else* donde se evalúa lo negativo se puede omitir y solo dejar la parte del *if* verdadero en el caso que lo necesites.

Switch

Otro condicional es el *swift* el cual cumple la misma función que el *if* solo que permite evaluar muchas condiciones sin repetir tantos *if*, esta estructura de conntrol hay muchos programadores que no le gustan pero para menús con números o con varias opciones es ideal. Vamos a ver la estructura básica de este condicional.

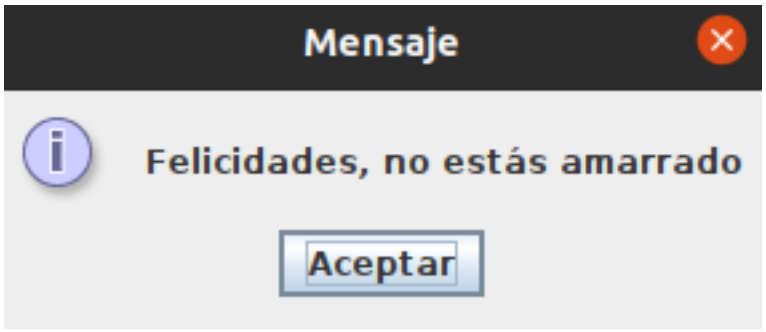


Figure 2: first if false

```
switch (variable){

  case 1: código; break;

  case 2: código; break;

  case 3: código; break;

  default: código; break;

}
```

Aquí se evaluará el contenido de la variable y en cada caso que de ahí viene el nombre se ejecutará un código y si no se cumple ningún caso entonces se ejecuta el código de *default*; el break lo colocamos para que si se ejecuta esa instrucción entonces se acabe ahí, si por algún caso lo quitamos entonces se ejecutará el que cumple la condición hasta el que tenga un brak entonces se ejecutarían muchos y eso no es lo que queremos. Vamos a hacer un ejemplo:

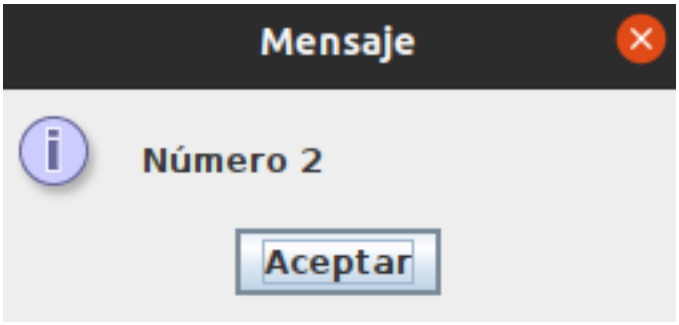
```
import javax.swing.JOptionPane;

public class condicional {

  public static void main(String[] args){

    int numero = 1;

    switch (numero){

      case 1: JOptionPane.showMessageDialog(null, "Número 1"); break;
```

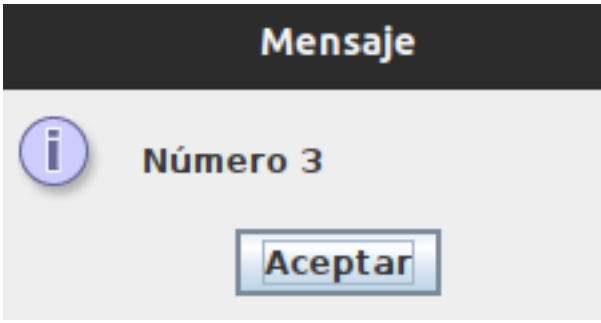
```
      case 2: JOptionPane.showMessageDialog(null, "Número 2"); break;

      case 3: JOptionPane.showMessageDialog(null, "Número 3"); break;

      default: JOptionPane.showMessageDialog(null, "Número desconocido"); break;

    }

  }
}
```

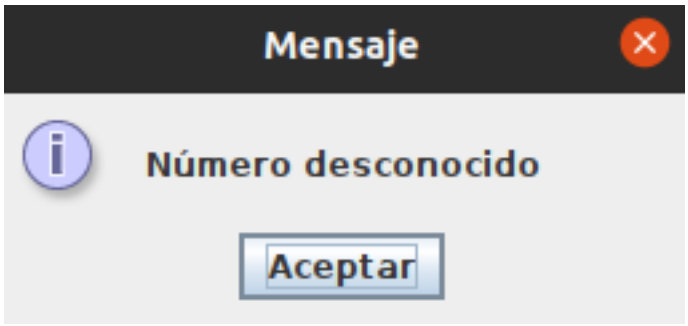
Este código devolverá lo siguiente:

Figure 3: switch-1

Y si cambiamos el valor de *numero* por 2, 3 o 4 mostrará lo siguiente respectivamente:






Mensaje
Número desconocido
Aceptar