

Program Control Statements

La utilidad de la clase Scanner

Para importar clases de un paquete se usa el comando **import**

La clase Scanner nos facilita mucho la tarea de obtención de datos desde diferentes fuentes.

Una de las utilidades de la clase Scanner es la obtención de datos **tecleados finalizando con un "enter"**.

```
import java.util.Scanner;  
Scanner sc = new Scanner(System.in);  
int i = sc.nextInt();
```

Métodos de lectura con la clase Scanner:

String	<u>next()</u> Finds and returns the next complete token from this scanner.
String	<u>nextLine()</u> Finds and returns the next complete line with spaces.
boolean	<u>nextBoolean()</u> Scans the next token of the input into a boolean value and returns that value.
byte	<u>nextByte()</u> Scans the next token of the input as a byte
double	<u>nextDouble()</u> Scans the next token of the input as a double.
float	<u>nextFloat()</u> Scans the next token of the input as a float.
int	<u>nextInt()</u> Scans the next token of the input as an int.
long	<u>nextLong()</u> Scans the next token of the input as a long.

Ejercicio:

Escribe un programa que solicite al usuario un valor numérico entero y muestre posteriormente el resultado de elevarlo al cubo.

Ejercicio:

Escribe un programa que solicite al usuario un valor booleano y muestre en pantalla el contrario a dicho valor.

Ejercicio:


Solicita el nombre al usuario y escribe en la pantalla un mensaje personalizado para él

1. The if Statement

```
if(condition)
    statement;
else
    statement;
```

Example:

```
// Guess the letter game.
import java.util.Scanner;
class Guess
{
    public static void main(String args[]) throws java.io.IOException {
        Scanner sc= new Scanner(System.in);
        char ch, answer = 'K';
        System.out.println("I'm thinking of a letter between A and Z.");
        System.out.print("Can you guess it: ");
        ch = (char) sc.next().charAt(0); // no existe nextChar()
        if(ch == answer)
            System.out.println("*** Acertaste ***");
    }
}
```



como no existe el metodo nextChar, tenemos que usar solo next(), pero este recoge un string, para atomizar un string usamos el metodo charAt(n) que descompone el string en sus caracteres por la posición comenzando por el 0.

Ejercicio:

Escribe un programa que solicite un valor numérico entero al usuario y muestre a continuación un mensaje que indique si el número es par o impar.

Where the targets of the **if** and **else** are single statements. **The else clause is optional.** The targets of both the **if** and **else** can be blocks of statements. The general form of the **if**, using blocks of statements, is

```
if(condition)
{
    statement sequence
}
else
{
```

```
    statement sequence  
}
```

Ejercicio:

Escribe un programa que pregunte al usuario si desea calcular el área de un rectángulo ('r') o el área de un cuadrado ('c'). En el primer caso, se le solicitarán los valores correspondientes a la base y altura del rectángulo y se mostrará a continuación el resultado de calcular el área correspondiente; en el segundo caso, se pedirá el valor correspondiente al lado del cuadrado y se mostrará posteriormente el resultado de calcular su área.

1.1. Nested ifs

A *nested if* is an **if** statement that is the target of another **if** or **else**.

The main thing to remember about nested **ifs** in Java is that an **else** statement always refers to the nearest **if** statement that is within the same block as the **else** and not already associated with an **else**.

Here is an example:

```
if(i == 10)  
{  
    if(j < 20)  
        a = b;  
    if(k > 100)  
        c = d;  
    else  
        a = c; // this else refers to if(k > 100)  
}  
else  
    a = d; // this else refers to if(i == 10)
```

Ejercicios:

- i. Dados tres valores numéricos leídos a través del teclado, muestra en pantalla el mayor de los tres (nota: consideramos que los tres valores son diferentes).
- ii. Dados dos valores numéricos leídos a través del teclado, muestra en pantalla un mensaje que indique si el primero es par y el segundo impar, si el primero es impar y el segundo es par, si ambos son pares o ambos son impares.

- iii. Dados tres valores numéricos leídos a través del teclado, muéstralos en pantalla ordenados de mayor a menor.
- iv. Dados dos valores numérico-enteros leídos a través del teclado, emite un mensaje en pantalla que indique si alguno de los dos es múltiplo del otro o son iguales.

1.2. The if-else-if Ladder

A common programming construct that is based upon the nested **if** is the **if-else-if ladder**. It looks like this:

```
if(condition)
    statement;
else if(condition)
    statement;
else if(condition)
    statement;
.
.
.
else statement;
```

2. The switch Statement

Básicamente, la expresión puede ser tipos de datos primitivos `byte`, `short`, `char` e `int`. A partir de `JDK7`, también funciona con tipos enumerados (`Enum` en `java`), la clase `String` y las clases `Wrapper`. `Switch` solo permite evaluar valores concretos de la expresión: no permite evaluar intervalos (pertenencia de la expresión a un intervalo o rango) ni expresiones compuestas

```
switch(expression) {
    case constant1:
        statement sequence
        break;
    case constant2:
        statement sequence
        break;
    case constant3:
        statement sequence
        break;
    ...
    default:
        statement sequence
}
tambien asi:
switch (expresión) {
```

```
    case valor1:

case valor2:

case valor3:

instrucciones;

break;
case valor4:
instrucciones;
break;
default:
sentencias;
break;
}
```

Ejemplos:

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

        int month = 8;
        String monthString;
        switch (month) {
            case 1: monthString = "January";
                    break;
            case 2: monthString = "February";
                    break;
            case 3: monthString = "March";
                    break;
            case 4: monthString = "April";
                    break;
            case 5: monthString = "May";
                    break;
            case 6: monthString = "June";
                    break;
            case 7: monthString = "July";
                    break;
            case 8: monthString = "August";
                    break;
            case 9: monthString = "September";
                    break;
            case 10: monthString = "October";
                    break;
        }
    }
}
```

```
        case 11: monthString = "November";
                break;
        case 12: monthString = "December";
                break;
        default: monthString = "Invalid month";
                break;
    }
    System.out.println(monthString);
}
}
```

```
public class StringSwitchDemo {

    public static int getMonthNumber(String month) {

        int monthNumber = 0;

        if (month == null) {
            return monthNumber;
        }

        switch (month.toLowerCase()) {
            case "january":
                monthNumber = 1;
                break;
            case "february":
                monthNumber = 2;
                break;
            case "march":
                monthNumber = 3;
                break;
            case "april":
                monthNumber = 4;
                break;
            case "may":
                monthNumber = 5;
                break;
            case "june":
                monthNumber = 6;
                break;
            case "july":
                monthNumber = 7;
                break;
            case "august":
                monthNumber = 8;
                break;
        }
    }
}
```

```
        break;
    case "september":
        monthNumber = 9;
        break;
    case "october":
        monthNumber = 10;
        break;
    case "november":
        monthNumber = 11;
        break;
    case "december":
        monthNumber = 12;
        break;
    default:
        monthNumber = 0;
        break;
}

return monthNumber;
}

public static void main(String[] args) {

    String month = "August";

    int returnedMonthNumber =
        StringSwitchDemo.getMonthNumber(month);

    if (returnedMonthNumber == 0) {
        System.out.println("Invalid month");
    } else {
        System.out.println(returnedMonthNumber);
    }
}
}
```

Obtener n°s aleatorios

En general, para conseguir un número entero entre M y N con $M < N$ y ambos incluidos, debemos usar esta fórmula

```
int valorEntero = Math.floor(Math.random() * (N-M+1)+M); // Valor entre M y N, ambos incluidos.
```

Si no queremos un valor entero sino *double*, la fórmula es sin el +1

```
double valorAleatorio = Math.random() * (N-M)+M;
```

eso sí, recuerda que el valor N queda excluido y no saldrá nunca.