UNIVERSIDAD
DE LA FRONTERA

Introducción a la Programación con Bluej

Guía de Referencia Rápida

Octubre 2016

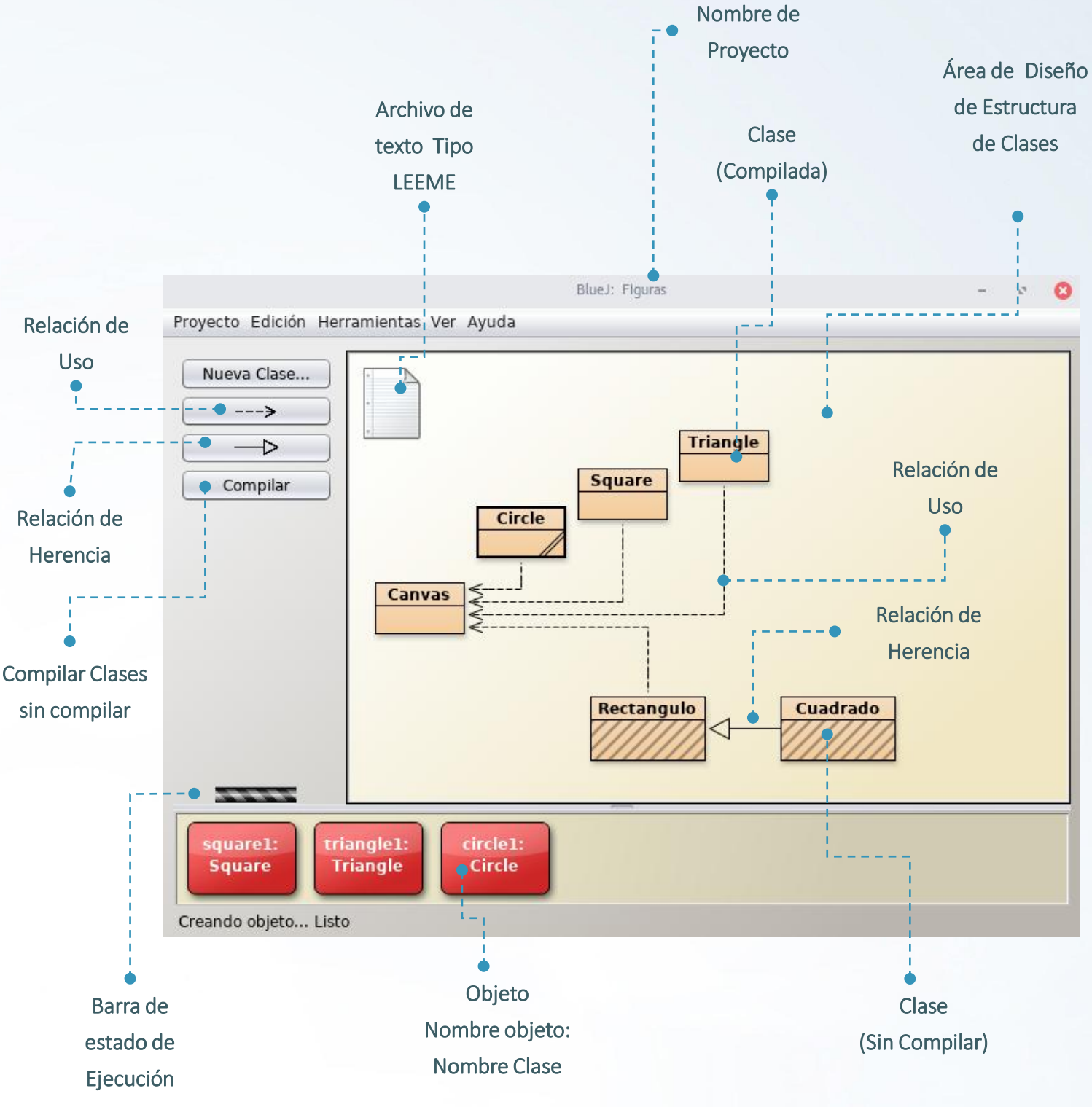
Departamento de
Computación e Informática

BlueJ es un entorno de desarrollo Integrado (IDE), el cual provee un editor, un compilador y un entorno de ejecución. BlueJ es una herramienta muy cómoda para los principiantes en la programación con Java.

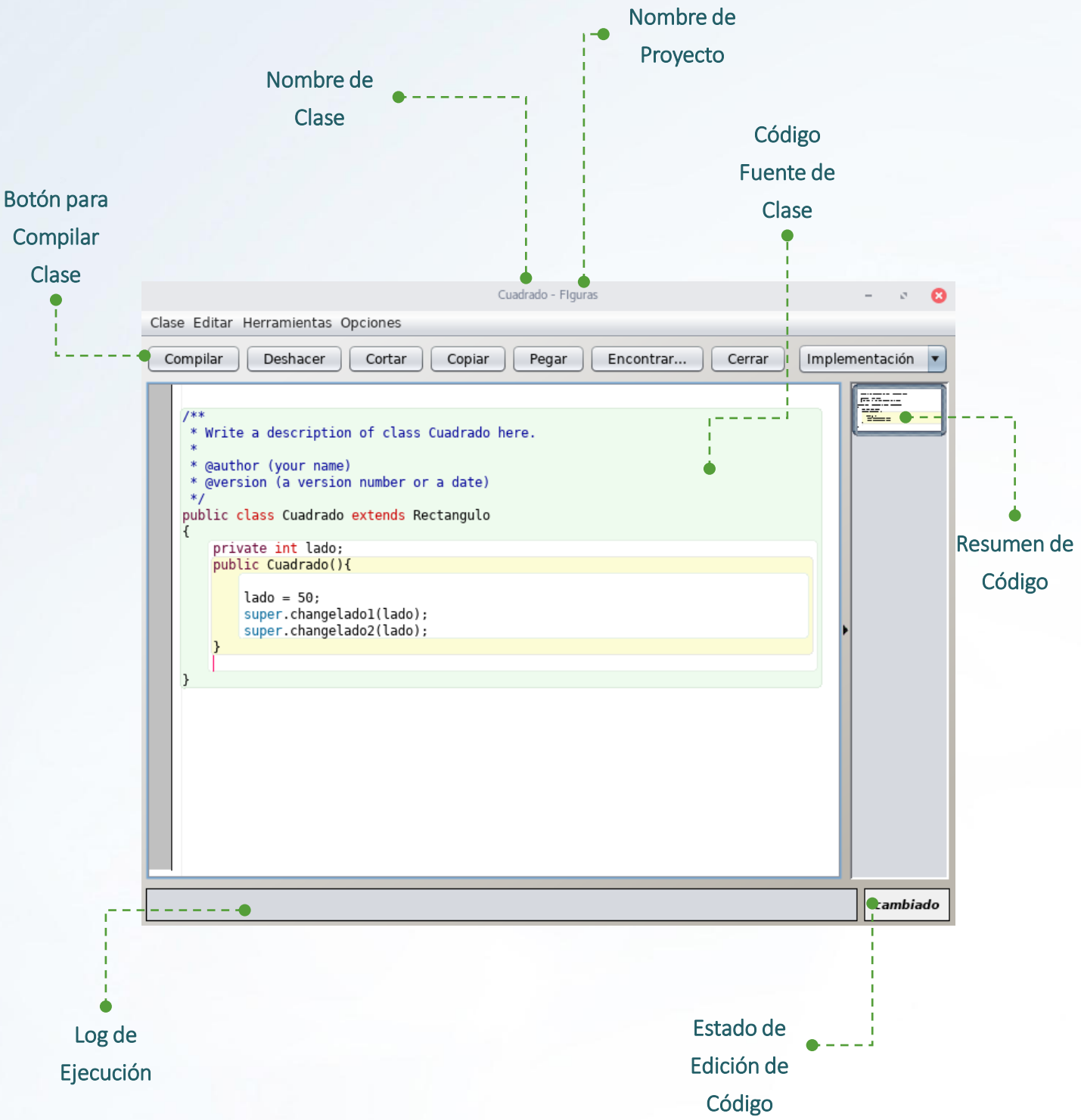
BlueJ se puede instalar en su computador de escritorio bajo cualquier sistema operativo y su instalador se puede descargar desde el link de su pagina oficial (http://bluej.org/).

Esta guía permite acceder rápidamente a los principales elementos de la herramienta para conocer y comenzar a utilizar.

Interfaz Principal BlueJ



Interfaz Editor de Código



Otras Interfaces

Nombre de Clase:
El nombre de una Clase Siempre debe comenzar por una letra MAYUSCULA y continuar por letras minúsculas.

Seleccionar tipos de Clases

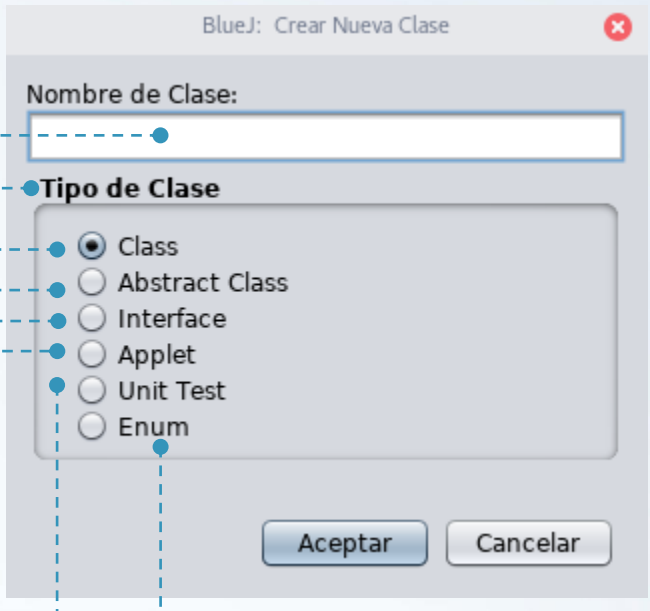
Class:
Define una Clase básica

Abstract Class:
Corresponde a una Clase que puede o no implementar algo. Esta clase no se puede declarar y solo las clases que la extiendan pueden utilizar sus métodos sobrescribiéndolos.

Interface:
Corresponde a una Generalización de Clases Abstractas. En estas clases solo se declaran métodos, pero no se implementan

Applet:
Corresponde a un pequeño programa en java que es embebido y puede ejecutarse sobre tecnología web.

Unit Test:
Corresponde a una clase especial que sirve para hacer test para otra clase y sus métodos.



Enum:
Es una clase especial que contiene constantes predefinidas

Tipos de Datos en Java

A continuación se presenta los tipos de datos básicos para programar en Java.

Tipo Numérico

boolean:
Este tipo de dato solo se puede definir con un valor true (verdadero) o false (falso)

byte:
Esta tipo de dato se define como un numero entero de 8 bits. (-128 a 127)

short:
Esta tipo de dato se define como un numero entero de 16 bits. (-32768 a 32767)

int:
Esta tipo de dato se define como un numero entero de 32 bits. ($2x10^9$)

long:
Esta tipo de dato se define como un numero entero de 64 bits.

float:
Esta tipo de dato se define como un numero real de 32 bits.

double:
Esta tipo de dato se define como un numero real de 64 bits.

```
boolean a;
byte    b;
short   c;
int     d;
long    e;
float   f;
double  g;
```

Clases Contenedoras (wrapper class):
Estas son clases que contienen los respectivos tipos de datos y sirven para ocupar constantes y métodos de cada respectiva clase.

```
Boolean h;
Byte    i;
Short   j;
Integer k;
Long    l;
Float   m;
Double  n;
```

Tipos de Datos en Java

Tipo caracter

char:
Corresponde a un carácter Unicode de 16 bit.

Character:
Corresponde a la clase contenedora de char.

```
char letra;
Character caracter;
```

String:
Corresponde a la clase que almacena cadenas de caracteres, estas cadenas se pueden concatenar con el operador + (suma).

```
String cadena;
```

Declarar y asignar variables en Java

La declaración de una variable se compone de 3 parte:

Tipo de dato — Nombre de la Variable — Valor (Opcional)

```
String cadena;
String frase= "hola mundo";
```

Declaración simple.

Declaración múltiple.

Declaración e inicialización.

Declaración mixta.

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

    boolean a;
    String cadena;

    int    b, d, numero;

    double  g=0;
    String frase= "hola mundo";

    byte h, i=0, e;
    char letra, letra0='e';

}
```

Operadores en Java

Operadores Matemáticos

Suma
Resta
Multiplicación
División
Modulo

```
public static void main (String[] args){
    double suma, resta, multi, div, modulo;
    int a= 4, b=7;

    suma=   a + b;
    resta=  a - b;
    multi=  a * b;
    div=    a / b;
    modulo= a % b;

    System.out.println("Suma:            "+suma);
    System.out.println("resta:           "+resta);
    System.out.println("multiplicación:  "+multi);
    System.out.println("División:        "+div);
    System.out.println("Modulo:          "+modulo);

}
```

Salidas

```
Suma:           11.0
resta:          -3.0
multiplicación: 28.0
División:       0.0
Modulo:         4.0
```

Operadores de Comparación

Igual
No Igual
Menor que
Mayor que
Menor o igual que
Mayor o igual que

```
public static void main (String[] args){
    boolean c;
    int a=3, b=2;

    c = a==b;
    System.out.println("a y b son iguales:        " + c);

    c = a!=b;
    System.out.println("a y b no son iguales:     " + c);

    c = a<b;
    System.out.println("a es menor que b:         " + c);

    c = a>b;
    System.out.println("a es mayor que b:         " + c);

    c = a<=b;
    System.out.println("a es menor o igual que b: " + c);

    c = a>=b;
    System.out.println("a es mayor o igual que b: " + c);

}
```

Salidas

```
a y b son iguales:        false
a y b no son iguales:     true
a es menor que b:         false
a es mayor que b:         true
a es menor o igual que b: false
a es mayor o igual que b: true
```

Operadores en Java

Operadores Lógicos

AND

OR

NOT

```
public static void main (String[] args){
    boolean t=true, f=false;
    boolean c;

    c= t && f;
    System.out.println("verdadero Y falso: "+ c);

    c= t || f;
    System.out.println("verdadero o falso: "+ c);

    c= !t;
    System.out.println("no verdadero:      "+ c);

    c= !f;
    System.out.println("no falso:          "+ c);

}
```

Salida

```
verdadero Y falso: false
verdadero o falso: true
no verdadero:      false
no falso:          true
```

Operadores Abreviados

variable++

Añade 1 después de utilizar la variable.

++variable

Añade 1 antes de utilizar la variable.

Variable - -

Añade 1 después de utilizar la variable.

--variable

Añade 1 antes de utilizar la variable.

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

    int a=0, b=0, c=0, d=0;

    System.out.println("Suma 1 a 'a': "+ (a++));
    System.out.println("Suma 1 a 'b': "+ (++b));

    System.out.println("Resta 1 a 'c': "+ (c--));
    System.out.println("Resta 1 a 'd': "+ (--d));

}
```

Salidas

```
Suma 1 a 'a': 0
Suma 1 a 'b': 1
Resta 1 a 'c': 0
Resta 1 a 'd': -1
```

Suma abreviada

a=a+10;

Resta abreviada

b=b+10;

Multiplicación abreviada

c=c+10;

División abreviada

d=d+10;

División abreviada

c=c+10;

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

    double a=5, b=5, c=5, d=5, e=5;

    System.out.println("Suma Abreviada:           "+ (a+=10));

    System.out.println("Resta Abreviada:          "+ (b-=10));

    System.out.println("Multiplicación Abreviada: "+ (c*=10));

    System.out.println("División Abreviada:       "+ (d/=10));

    System.out.println("Modulo Abreviada:         "+ (e%=10));

}
```

Salidas

```
Suma Abreviada:           15.0
Resta Abreviada:          -5.0
Multiplicación Abreviada: 50.0
División Abreviada:       0.5
Modulo Abreviada:         5.0
```

Estructuras de Control en Java

If - else

Si (if) la condición lógica es verdadera se ejecuta el primer set de instrucciones, sino (else) se ejecuta el segundo set de instrucciones.

Forma 1

```
//...
// Instrucciones
if(/*condición lógica*/ x<y){
    //...
    // Instrucciones
    //...
}else{
    //...
    // Instrucciones
    //...
}
// Instrucciones
//...
```

Instrucciones previas

Pueden ser cualquier tipo de instrucciones o estructuras de control;

Condiciones lógicas:

Operaciones que requieren el uso de operadores que entreguen un resultado de tipo boolean (true o false);

Instrucciones posteriores

Pueden ser cualquier tipo de instrucciones o estructuras de control;

Forma 2 (abreviada)

```
//...
// Instrucciones
if(/*condición lógica*/)// Instrucción ;
else
// Instrucciones;
//...
```

Instrucción

Esta forma permite solo una instrucción;

Forma 3 (múltiple)

```
//...
// Instrucciones
if(/*condición lógica*/){
    /*Instrucciones*/
}else if(/*condición lógica*/){
    /*Instrucciones*/
}else if(/*condición lógica*/){
    /*Instrucciones*/
}else if(/*condición lógica*/){
    /*Instrucciones*/
}else if(/*condición lógica*/){
}//...
//else if(/*condición lógica*/){
///*instrucciones*/
//}else if(/*condición lógica*/){
//}
else{
    /*instrucciones*/
}
// Instruccionees;
//...
```

Esta forma permite múltiples condiciones if una posterior a la otra permitiendo utilizar diferentes condiciones lógicas;

while

```
//...
// Instrucciones
while(/*condición lógica*/){
    //...
    // Instrucciones ;
    //...
 }
// Instrucciones;
//...
```

While (mientras)

Mientras la condición lógica es verdadera, se ejecutan las condiciones dentro de la estructura de control, verificando la condición lógica al inicio de cada ciclo lógico. Es siempre necesario revisar la condición lógica dentro del set de instrucciones, sino se generara un loop infinito;

do - while

```
//...
// Instrucciones;
do{
    //...
    // Instrucciones;
    //...
}while(/*condición lógica*/);
// Instrucciones;
//...
```

Do - While (hacer - mientras)

Ejecuta un grupo de instrucciones delimitadas por la estructura de control, verificando la condición lógica al final de cada ciclo. Es siempre necesario revisar la condición lógica dentro del set de instrucciones, sino se generara un loop infinito;

for

```
//...
// Instrucciones;
for (/*variable*/;/*condición lógica*/;/*modificador*/){
    //...
    // Instrucciones;
    //...
}
// Instrucciones;
//...
```

For (para)

Consiste de una estructura cíclica que ejecuta un set de instrucciones mientras la condición lógica sea verdadera, esta estructura permite añadir un contador en la que se puede basar la condición lógico. La condición lógica debe ser falsa en algún momento si no se generara un loop infinito

Ejemplo

```
for (int i=0;i<100;i++){
    System.out.println("Numero: "+i);
}
```

Imprime por consola 100 veces el valor del contador “i”

switch

```
//...
// Instrucciones;
switch(x){
    case 0:/*Instrucciones*/;break;
    case 1:/*Instrucciones*/;break;
    case 2:/*Instrucciones*/;break;
  //case 3:/*Instrucciones*/;break;
  //case 4:/*Instrucciones*/;break;
  //...
}
// Instrucciones;
//...
```

Switch

Permite el uso de variados set de instrucciones basado en opciones: las opciones pueden ser de tipo int, char o string;

```
//...
// Instrucciones;
switch(x){
    case 'a':
                System.out.println("hola Venus");
                System.out.println("Opcion: "+ x);
                break;
    case 'b':
                System.out.println("hola Tierra");
                System.out.println("Opcion: "+ x);
                break;
    case 'c':
                System.out.println("hola Marte");
                System.out.println("Opcion: "+ x);
                break;
  //case 'd':/*Instrucciones*/;break;
  //case 'e':/*Instrucciones*/;break;
  //...
}
// Instrucciones;
//...
```

break

Este operador cancela un set de instrucciones de una estructura de control

Para este caso el operador break, permite finalizar el switch y que no continúe con los casos restantes.

try - catch

Esta estructura permite controlar las validaciones que puedan generar errores por excepciones que detengan el funcionamiento de un programa.

```
try {
    //...
    //Instrucciones
    //...
} catch (/*Excepcion a capturar*/) {
    //...
    //Instrucciones
    //...
}
```

función con excepciones

Instrucciones a ejecutar luego de que se capture la excepción

```
try {
    int entero;
    entero= in.nextInt();
    System.out.println(entero);
} catch (Exception excepcion) {
    System.out.println("se produjo un error");
}
```

Declaración de la excepción

Operadores y Estructura de Java

Permisos Clases, Funciones y Variables

Public

Permite acceder desde la clase, package, subclase y el mundo.

```
public class Canvas
{
```

Protected

Permite acceder desde la clase, package y subclase.

```
protected void suma(){
```

Sin control de acceso

Permite acceder desde la clase y package

```
void resta(){
```

Private

Permite acceder solo desde la clase.

```
private int lado;
```

Static

Static

- Este operador aplicado a un método , permite que uno pueda acceder de forma directa a el, de modo que no es necesario crear un objeto de la clase para ocupar este método.
- Aplicado a una variable implica que es posible acceder a esta variable desde cualquier parte de la clase y no puede existir una copia de esta a lo largo de la clase.
- Aplicado este operador a una clase se utiliza cuando se declara para declarar una clase dentro de otra general, y que no se pueda acceder desde esa clase a otras clases del mismo grupo de clases.

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

Final

Final

- Este operador aplicado a una clase, evita que este este posea una Subclase
- Este operador aplicado a un método implica que no podrá ser sobrescrito por subclases.
- Este operador aplicado a una variable, implica que esta variable no podrá ser modificada por nadie, convirtiéndose en una constante.

```
final int sumar(int a){
    return (a+10)
}
```

Estructura básica de una clase

Librerías :

Se declaran TODAS las librerías que se utilizaran en la clase.

Declaración de clase:

Se declara la clase con el mismo nombre del archivo ".java"

Variables de la clase

Se declaran las variables globales de la clase de ser necesarias.

Constructor

Corresponde al método que se ejecuta de forma automática cuando un objeto es creado (una clase puede tener de o a n constructores distintos).

Métodos

Se declaran tantos métodos como sea necesario para cumplir con las funcionalidades del objeto.

```
import java.awt.*;
import java.util.*;

public class Cuadrado extends Rectangulo
{
    private int lado;

    public Cuadrado(){

        lado = 50;
        super.changelado1(lado);
        super.changelado2(lado);
    }

    protected void suma(){
        ++this.lado
    }

    void resta(){
        ++this.lado
    }
}
```

Operadores de Clases

this

Hace referencia al objeto que se esta utilizando.

```
void resta(){
    this.lado -= 10;
}
```

extend

Este operador se utiliza para señalar que la clase es “hija” de una clase. Este operador permite luego ocupar el los métodos de la clase padre a través de la operados **super**.

```
public class Cuadrado extends Rectangulo
{
    private int lado;

    public Cuadrado(){

        lado = 50;
        super.changelado1(lado);
        super.changelado2(lado);

    }
```

Super

Este operador se utiliza para ocupar los métodos de la clase padre desde un método una clase hijo

```
public Cuadrado(){

    lado = 50;
    super.changelado1(lado);
    super.changelado2(lado);

}
```

Cast (forzar tipo)

Convierte un tipo de dato en otro, para que pueda ser utilizado, si el dato se convierte a otro de menor precisión. este se trunca.

Double -> Entero

```
public static void main(String[] args){
    double decimal = 2.98000456;
    int entero;

    System.out.println("Decimal: "+decimal);
    entero = (int)decimal;
    System.out.println("Entero:  "+entero);

}
```

Salida

```
Decimal: 2.98000456
Entero:  2
```

Clases y Métodos de uso Frecuente

Principales métodos de la clase Object

En java, todas las clases creadas y las variables, heredan de la clase Object, por lo que se pueden ocupar sus métodos (sin ocupar el operador super)..

equals()

Permite comparar 2 objetos, se utiliza generalmente para la comparación de Strings dado que el operador ==, solo es para datos de tipo numérico

toString()

Permite convertir un atributo de un objeto en Strings.

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

    String  frase1="palabra", frase2="palabra";
    Double  numero= 98.934;

    if (frase1.equals(frase2)){

        frase1= numero.toString();
        System.out.println(frase1);

    }else{
        System.out.println("no son iguales");
    }

}
```

Salida

```
98.934
6
```

String

String, es una clase que permite la creación de arreglos de caracteres de forma simple.

Declarar un String

Asignar un dato a un String

Concatenar un String

El operador de suma (+) es el encargado de hacer la concatenación de Strings

length()

Permite conocer el largo de un String

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

    String  frase;

    frase= "hola mundo, Introducción a la Programación";
    System.out.println(frase);

    frase = frase + ", concatenar un String";
    System.out.println(frase);

    System.out.println(frase.length());

}
```

Salida

```
hola mundo, Introducción a la Programación
hola mundo, Introducción a la Programación, concatenar un String
64
```

Scanner

Scanner es el método por el cual es posible obtener datos por teclado u otras fuentes como un archivo.

Librería Scanner

Permite crear elementos de la clase Scanner.

Conjunto Librerías util

Contiene las librerías para las entrada por teclado entre otras.

nextByte()

Permite recibir por teclado variables de tipo byte.

nextShort()

Permite recibir por teclado variables de tipo short.

nextInt()

Permite recibir por teclado variables de tipo int.

nextLong()

Permite recibir por teclado variables de tipo long.

nextFloat()

Permite recibir por teclado variables de tipo float.

nextDouble()

Permite recibir por teclado variables de tipo double.

next()

Permite recibir por teclado variables de tipo String .

nextLine()

Permite recibir por teclado variables de tipo String, a diferencia de next(), este método captura toda una frase, incluyendo espacios.

```
import java.util.Scanner;
import java.util.*;

public class Principal
{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        byte a;
        a= in.nextByte();
        System.out.println(a);

        short b;
        b= in.nextShort();
        System.out.println(b);

        int c;
        c= in.nextInt();
        System.out.println(c);

        long d;
        d = in.nextLong();
        System.out.println(d);

        float e;
        e = in.nextFloat();
        System.out.println(e);

        double f;
        f = in.nextDouble();
        System.out.println(f);

        String g;
        g=in.next();
        System.out.println(g);

        String h;
        h=in.nextLine();
        System.out.println(h);


    }
}
```

Math

Corresponde a una que contiene métodos para realizar operaciones matemáticas básicas además de contener constantes y proporcionar el método random(), para generar números aleatorios.

Librería Math

Permite crear elementos de la clase Math.

Constante euler

Crea constante doublé con el valor de euler

Constante Pi

Crea constante doublé con el valor de Pi

```
import java.lang.Math;

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

        System.out.println("Constante de euler: "+Math.E);

        System.out.println("Constante de pi:    "+Math.PI);

    }
}
```

Salida

```
Constante de euler: 2.718281828459045
Constante de pi:    3.141592653589793
```

Funciones trigonométricas

Math posee una serie de funciones trigonométricas incluyendo los inversos.

Funciones raíz

Funciones potencia.

```
public static void main(String[] args){
    double coseno;
    coseno = Math.cos(5.0);
    System.out.println("Coseno: "+coseno);

    double raiz;
    raiz = Math.sqrt(16.0);
    System.out.println("Raiz Cuadrada: "+raiz);

    double potencia;
    potencia = Math.pow(5.0, 3);
    System.out.println("Potencia: "+potencia);

}
```

Salida

```
Coseno: 0.28366218546322625
Raiz Cuadrada: 4.0
Potencia: 125.0
```

Números Aleatorios

Librería Math

```
import java.lang.Math;

public class Principal
{
    public static void main(String[] args){
        double aleatorio;
        for (int i=0;i<5;i++){
            aleatorio = Math.random();
            System.out.println("aleatorio: "+aleatorio);
        }
    }
}
```

Math. random

Permite crear números (double) aleatorio entre 0 y 1

```
aleatorio: 0.7921891383492613
aleatorio: 0.26223555440616897
aleatorio: 0.940786642637996
aleatorio: 0.8507893635670716
aleatorio: 0.6258218620449154
```

Clase Random

Permite crear números aleatorios entre 0 y el numero que se entrega.

```
import java.util.Random;

public class Principal
{
    public static void main(String[] args){
        double aleatorio;
        Random rNum= new Random();
        for (int i=0;i<5;i++){
            aleatorio = rNum.nextInt(10);
            System.out.println("aleatorio: "+aleatorio);
        }
    }
}
```

```
aleatorio: 9.0
aleatorio: 8.0
aleatorio: 2.0
aleatorio: 9.0
```

Semilla Aleatoria

Crea números aleatorios generados a partir de una semilla.

```
import java.util.Date;
import java.util.Random;

public class Principal
{
    public static void main(String[] args){
        double aleatorio;
        Date semilla= new Date();

        Random rNum= new Random(semilla.getTime());

        for (int i=0;i<5;i++){
            aleatorio = rNum.nextInt(900);
            System.out.println("aleatorio: "+aleatorio);
        }
    }
}
```

```
aleatorio: 600.0
aleatorio: 784.0
aleatorio: 336.0
aleatorio: 431.0
aleatorio: 598.0
```

Arreglos

Los arreglos son contenedores de objetos (objetos, Strings, enteros, caracteres, arreglos, etc.) de largo fijo, que almacenan valores de un solo tipo.

Declaración de un tipo de arreglo y si nombre.

Declaración de largo

Asignar valores a un arreglo

```
public static void main(String[] args){
    int[] arreglo;
    arreglo = new int[10];

    for(int i=0;i<10;i++){

        arreglo[i]= i;
        System.out.print("[ "+ arreglo[i]+" ]");

    }
}
```

Los espacios de memoria de un arreglo van de 0 á n-1, donde "n" es el largo

```
[ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ][ 8 ][ 9 ]
```

Se asigna el valor **"i"** en la posición **"i"** del arreglo.

```
arreglo[i]= i;
```

Es posible crear arreglos en donde el tamaño inicial no sea asignado de inmediato

Asignación por teclado del largo del arreglo

```
public static void main(String[] args){
    int[] arreglo;
    int largo;

    System.out.println("Ingrese el largo del arreglo");
    Scanner n = new Scanner(System.in);
    largo = n.nextInt();

    arreglo = new int[largo];

    for(int i=0;i<largo;i++){

        arreglo[i]= i;
        System.out.print("[ "+ arreglo[i]+" ]");

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

Generación de un arreglo de largo 4

Generación de un arreglo de largo 9

```
Ingrese el largo del arreglo
4
[ 0 ][ 1 ][ 2 ][ 3 ]
Ingrese el largo del arreglo
9
[ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ][ 8 ]
```

Arreglos

Asignación Directa

Permite crear un arreglo con valores fijos.

length

Devuelve un entero con el largo del arreglo este .

```
public static void main(String[] args){
    int[] arreglo= {9, 0, 6, 7, 10, 1001, 1};
    int largo;

    for(int i=0;i<arreglo.length;i++){

        System.out.print("[ "+ arreglo[i]+" ]");

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

Salida

```
[ 9 ][ 0 ][ 6 ][ 7 ][ 10 ][ 1001 ][ 1 ]
```

Clase Arrays

Por medio de esta clase se pueden ocupar métodos muy útiles para el manejo de arreglos

Librerias

Librería Principal

Librería Opcional

```
import java.util.Arrays;
import java.lang.reflect.Array;


public class Princial
{
    public static void main(String[] args){
        int[] arreglo = new int[10];
        Arrays.fill(arreglo,1);

        for(int i=0;i<arreglo.length;i++){
            System.out.print("[ "+ arreglo[i]+" ]");
        }

        System.out.println("");

    }
}
```

fill

Llena el arreglo con un valor determinado (útil.Arrays)

```
[ 1 ][ 1 ][ 1 ][ 1 ][ 1 ][ 1 ][ 1 ][ 1 ][ 1 ][ 1 ]
```

equals()

Permite comparar 2 arreglos (útil.Arrays)

```
public static void main(String[] args){
    int[] arreglo1 = new int[10];
    int[] arreglo2 = new int[10];

    Arrays.fill(arreglo1,1);
    Arrays.fill(arreglo2,1);

    if (Arrays.equals(arreglo1, arreglo2)){
        System.out.println("son iguales");
    }else {
        System.out.println("no son iguales");
    }

    System.out.println("");

}
```

```
son iguales
```

Clase Arrays

Librerias

Librería alternativa con funciones complementarias

```
import java.lang.reflect.Array;

public class Principal
{
    public static void main(String[] args){
        int[] arreglo1 = new int[10];
        int[] arreglo2 = new int[10];
        int k=0;

        for (int i=0;i<10;i++){
            arreglo1[i]=i*i;
            System.out.print("[ "+ arreglo1[i]+ " ]");
        }
        System.out.println("");

        k = Array.getInt(arreglo1,4);
        System.out.println(""+ k);

    }
}
```

getInt()

Entrga el valor entero de un arreglo en el índice indicado.

```
[ 0 ][ 1 ][ 4 ][ 9 ][ 16 ][ 25 ][ 36 ][ 49 ][ 64 ][ 81 ]
16
```

setInt()

Asigna un valor a un elemento de un arreglo en el índice indicado.

```
public static void main(String[] args){
    int[] arreglo1 = new int[10];
    int k=0;

    for (int i=0;i<10;i++){
        arreglo1[i]=i*i;
        System.out.print("[ "+ arreglo1[i]+ " ]");
    }
    System.out.println("");

    Array.setInt(arreglo1, 4,80 );

    for (int i=0;i<10;i++){
        System.out.print("[ "+ arreglo1[i]+ " ]");
    }
    System.out.println("");
}
```

```
[ 0 ][ 1 ][ 4 ][ 9 ][ 16 ][ 25 ][ 36 ][ 49 ][ 64 ][ 81 ]
[ 0 ][ 1 ][ 4 ][ 9 ][ 80 ][ 25 ][ 36 ][ 49 ][ 64 ][ 81 ]
```

DEPARTAMENTO DE COMPUTACIÓN E INFORMÁTICA

Documentación Online

Documentación completa de Librerías y Clases de Java

http://docs.oracle.com/javase/8/docs/api/index.html

Java 8 Docs

Posee una base completa de las clases y sus respectivos métodos y relaciones con otras clases.

http://docs.oracle.com/javase/7/docs/api/index.html

Java 7 Docs

Posee una base completa de las clases y sus respectivos métodos y relaciones con otras clases.

Tutoriales Oracle

http://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html

Tutoriales

Pagina Principal de tutoriales de Java.

http://docs.oracle.com/javase/tutorial/getStarted/index.html

Iniciando en Java

Contiene tutoriales de conceptos de inicio.

http://docs.oracle.com/javase/tutorial/java/index.html

Aprendiendo Java

Contiene resúmenes de diversos conceptos básicos de java

http://docs.oracle.com/javase/tutorial/essential/index.html

Clases Esenciales

Contiene ejemplos y documentación de las principales clases de Java.

UNIVERSIDAD
DE LA FRONTERA

Guía de Referencia Rápida

UNIVERSIDAD DE LA FRONTERA
FACULTAD DE INGENIERÍA Y CIENCIAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INFORMÁTICA

Avda Francisco Salazar 01145
Temuco – Chile / casilla 54-D
Fono (56) 45 2325000 /2744219

dci.ufro.cl

Departamento de
Computación e Informática