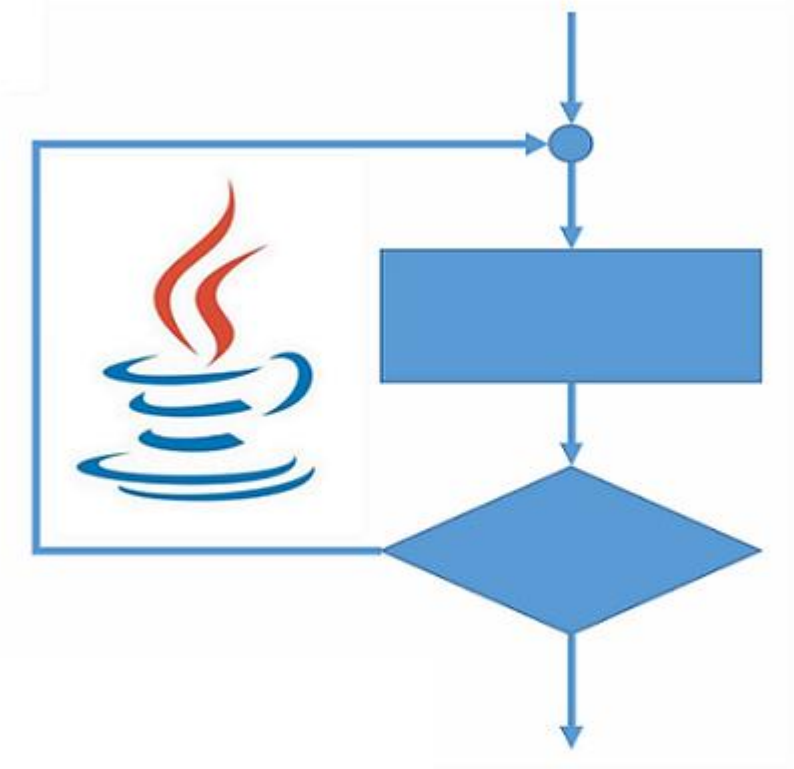
GUSTAVO CORONEL
DESARROLLA SOFTWARE

FUNDAMENTOS DE PROGRAMACIÓN CON JAVA



UNIDAD 06
ESTRUCTURAS REPETITIVAS

Eric Gustavo Coronel Castillo
youtube.com/DesarrollaSoftware
gcoronel@uni.edu.pe

INDICE

BUCLE: WHILE

Esta instrucción de bucle permite repetir la ejecución de un grupo de instrucciones mientras se cumpla cierta condición.

Sintaxis

```
while ( condición ){

   // Bloque de instrucciones

}
```

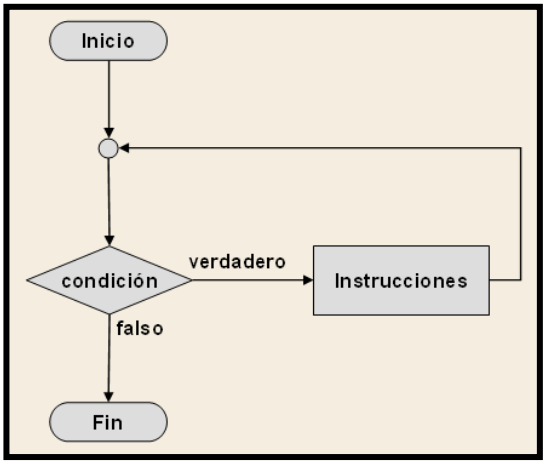


Diagrama de Flujo del bucle while.

Para iniciar el bucle, la primera vez que se evalué la condición debe dar verdadero, caso contrario no existirá ninguna iteración.

Otro aspecto importante es que debe existir alguna instrucción dentro del bucle que cambie la condición, como por ejemplo el valor de un contador, caso contrario estaríamos en un bucle infinito.

Ejemplo 1

El siguiente ejemplo muestra los números pares entre m y n (m < n), los valores de m y n se deben ingresar como parámetros al programa.

```
public class prog0607
{
   public static void main(String[] args)
   {
      int m = Integer.parseInt( args[0] );
      int n = Integer.parseInt( args[1] );

      while(m <= n){
         if( m%2 == 0 )
            System.out.println( m );
         m++;
      }
   }
}
```

El valor de m es modificado (**m++;**) dentro del bloque de instrucciones, de no ser así la condición nunca llegaría a ser falsa, en cuyo caso el bucle no terminaría, sería un bucle infinito.

Ejemplo 2

El siguiente programa recibe como parámetros un mes (número) y un día del mismo mes, considerando solo el presente año (2005), luego calcula los días transcurridos desde el 1ro de Enero hasta ese día del mes ingresado, y el día de la semana al que pertenece.

Debes tener en cuenta que el 1ro de Enero fue sábado.

```
public class prog0608
{
   public static void main(String[] args)
   {
      int mes = Integer.parseInt( args[0] );
      int dia = Integer.parseInt( args[1] );
      int dt = 0; // Acumulador de dias transcurridos
      int k = 0;  // Contador
      int ids = 0;    // Dia de la semana
      String sds = ""; // Nombre del dia de la semana

      while(++k <  mes){
         switch(k){
            case 1: case 3: case 5: case 7:
            case 8: case 10: case 12:
               dt += 31;
               break;
            case 4: case 6: case 9: case 11:
               dt += 30;
               break;
            case 2:
               dt += 28;
               break;
         }
      }

      ids = dt % 7; // Dia de la semana
      switch(ids){
         case 0: sds = "Viernes"; break;
         case 1: sds = "Sabado"; break; // 1 de Enero fue sábado
```

```
        case 2: sds = "Domingo"; break;
        case 3: sds = "Lunes"; break;
        case 4: sds = "Martes"; break;
        case 5: sds = "Miercoles"; break;
        case 6: sds = "Jueves"; break;
      }

      System.out.println( "Dias transcurridos: " + dt );
      System.out.println( "Dia de la semana: " + sds );
    }
}
```

Primero se muestra la fecha del sistema, luego ingresamos al programa el 7 de diciembre, primero el mes y luego el día, y el programa nos muestra el resultado esperado.

Ejemplo 3

El siguiente programa permite visualizar la tabla de multiplicar de un número **n**. El valor de **n** se ingresa como parámetro en la línea de comandos.

```
public class prog0609
{
   public static void main(String[] args)
   {
      int n = Integer.parseInt( args[0] );
      int k = 0; // Contador
      int r;     // Variable auxiliar

      System.out.println( "Tabla de Multiplicar del numero: " + n );
      while(++k <= 12){
         r = n * k;
         System.out.println( k + " * " + n + " = " + r );
      }
      System.out.println( "--- Fin ---" );

   }
}
```

La variable **k** se utiliza como contador, note usted que el incremento de esta variable se realiza en la misma expresión de condición, lo cual es totalmente valido.

BUCLE: DO – WHILE

Esta estructura de bucle permite repetir la ejecución de un grupo de instrucciones mientras se cumpla cierta condición, similar al bucle `while`, la diferencia está en que la condición se evalúa después de la primera iteración, mientras que en el bucle `while` la condición se evalúa antes de ejecutar cualquier iteración.

Sintaxis

```
do{

   // Bloque de instrucciones

} while ( condición );
```

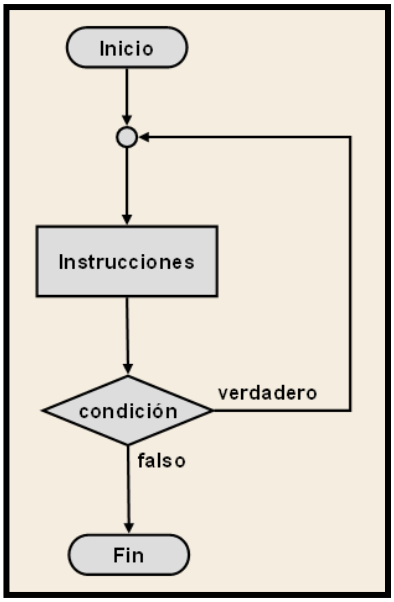


Diagrama de flujo del bucle do - while

Para ejecutar la primera iteración no es necesario verificar la condición.

Ejemplo 4

El siguiente ejemplo imprime los n primeros términos de la serie de Fibonacci, los dos primeros términos son 0 y 1 respectivamente, para calcular un término k (Tk) se utiliza la siguiente formula:

```
Tk = Tk-1 + Tk-2
```

De tal manera que la serie toma la siguiente forma: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . etc.

```
public class prog0610
{
   public static void main(String[] args)
   {
      int n = Integer.parseInt( args[0] );
      int a = 0, b = 1, k = 2, c;

      System.out.println( "Serie de Fibonacci" );
      System.out.print( a + "\t" + b );
      do{
         k++;
         c = a + b;
         System.out.print( "\t" + c );
         a = b;
         b = c;
      }while (k < n);
      System.out.println( "\n--- Fin ---" );

   }
}
```

El valor de **n** se pasa como parámetro en la línea de comando.

BUCLE: FOR

El bucle `for` está basado en tres expresiones, las que dicho sea de paso son opcionales, tal como se aprecia en su sintaxis.

Sintaxis

```
for( ExpInicialización; Condición; ExpFinalización ) {

   // Bloque de Instrucciones

}
```

ExpInicialización Esta sección se ejecuta antes de la primera iteración, normalmente se utiliza para inicializar variables, como por ejemplo contadores.

Condición Es la condición que determina si se inicia la ejecución de la primera iteración, y posteriormente las siguientes iteraciones.

ExpFinalización Esta sección se ejecuta después del bloque de instrucciones propias del bucle.

En la siguiente figura puedes apreciar en forma gráfica como es que funciona el bucle `for`.

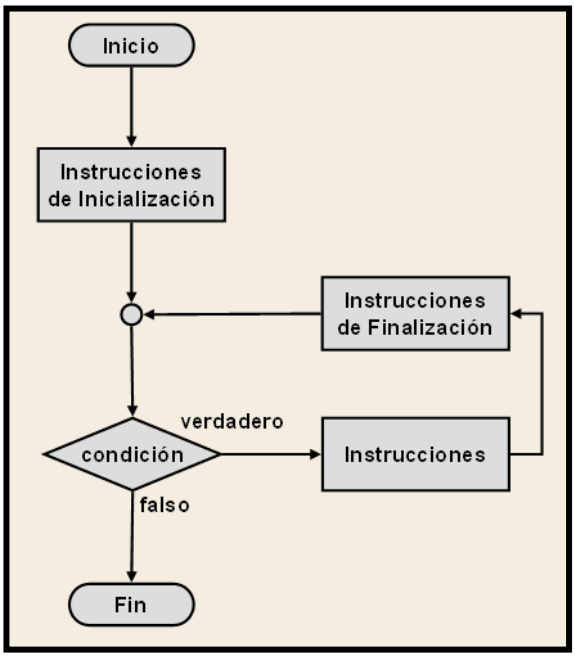


Diagrama de flujo del bucle for.

Ejemplo 5

El siguiente ejemplo permite calcular el factorial de un número. El número se pasa como parámetro en la línea de comando.

```
public class prog0611
{
   public static void main(String[] args)
   {
      int n = Integer.parseInt( args[0] );
      int f = 1;

      for( int k=2; k<=n; k++ ){
         f *= k;
      }

      System.out.println( "n: " + n );
      System.out.println( "Factorial: " + f );

   }

}
```

Ejemplo 6

El siguiente ejemplo permite elevar un número n a una potencia p. La restricción es considerar solo números enteros, y para p solo positivos.

```
public class prog0612
{
   public static void main(String[] args)
   {
      int n = Integer.parseInt( args[0] );
      int p = Integer.parseInt( args[1] );
      int k, r;

      for( k=1, r=1; k<=p; k++ ){
         r *= n;
      }

      System.out.println( "n: " + n );
      System.out.println( "p: " + p );
      System.out.println( "Resultado: " + r );

   }
}
```

Observe que la expresión de inicialización del `for` tiene dos instrucciones separadas por coma (,), de igual forma la expresión de finalización puede tener más de una instrucción.

Bajo estas consideraciones podría calcular el factorial de 5 de la siguiente manera:

```
int n = 5, k, f;
for(k=2, f=1; k<=n; f*=k, k++ );
System.out.println( f );
```

O también podría ser así:

```
int n = 5, k, f;
for(k=2, f=1; k<=n; f*=k++ );
System.out.println( f );
```

Ejemplo 7

En este ejemplo se trata de mostrar la flexibilidad de la instrucción `for`, utilizando solo la condición sería muy similar `while`, pero con `for`.

Este ejemplo calcula el MCD y MCM de dos números enteros.

```
public class prog0613
{

   public static void main(String[] args)
   {
      int n1 = Integer.parseInt( args[0] );
      int n2 = Integer.parseInt( args[1] );
      int n1_aux = n1, n2_aux = n2, mcd, mcm;

      for( ; n1 != n2; )
         if(n1 > n2) n1 -= n2;
         else n2 -= n1;

      mcd = n1;
      mcm = n1_aux * n2_aux / mcd;

      System.out.println( "n1  => " + n1_aux );
      System.out.println( "n2  => " + n2_aux );
      System.out.println( "mcd => " + mcd );
      System.out.println( "mcm => " + mcm );

   }

}
```

BUCLE FOR CON ARREGLOS

Para procesar los elementos de un arreglo podrías utilizar el siguiente bucle:

```
for( k=0; k<lista.length; k++ ){

   // Instrucciones

}
```

En este caso **lista** es el arreglo, por ejemplo:

```
int lista[] = {12,15,45,23,80}, k;
for( k=0; k<lista.length; k++ )
   System.out.println( lista[k] );
```

En el ejemplo anterior se está inicializando **lista** como un arreglo de enteros y luego imprimimos dicha lista.

También puedes utilizar el siguiente formato:

```
for( tipo variable: arreglo ){

   // Instrucciones

}
```

Para imprimir los elementos de la lista del ejemplo anterior, tenemos la siguiente solución:

```
int lista[] = {12,15,45,23,80};
for( int n: lista )
   System.out.println( n );
```

Ejemplo 8

El siguiente ejemplo trata de un programa que recibe a través de la línea de comando una serie de números, y de lo que se trata es de calcular la suma de todos ellos.

```
public class prog0614
{

   public static void main(String[] args)
   {
      int suma = 0;

      for( String dato: args )
         suma += Integer.parseInt( dato );

      System.out.println( "Suma => " + suma );

   }

}
```

BUCLE FOR CON COLECCIONES

También podemos utilizar el bucle for para recorren una colección, el formato es el siguiente:

```
for( tipo variable: colección ) {

   // Instrucciones

}
```

Supondremos que **amigos** es una colección de elementos de tipo String**,** para recorrer sus elementos e imprimirlos la solución sería así:

```
for( String dato: amigos )
   System.out.println( dato );
```

Ejemplo 9

En el siguiente ejemplo creamos una colección de nombre amigos y utilizamos el bucle **for** para recorrer sus elementos y mostrarlos en pantalla.

```
import java.util.ArrayList;

public class prog0615
{

   public static void main(String[] args)
   {
      ArrayList<String> amigos = new ArrayList<String>();

      amigos.add(new String("Sergio"));
      amigos.add(new String("Guino"));
      amigos.add(new String("Ricardo"));
      amigos.add(new String("Julio"));
      amigos.add(new String("Hugo"));

      for( String dato: amigos )
         System.out.println( dato );

   }

}
```

INSTRUCCIONES: BREAK Y CONTINUE

El funcionamiento de estas instrucciones se describe a continuación:

break	Permite salir de una instrucción switch, while, do-while y for.
continue	Permite saltar al inicio de un bucle.

Ejemplo 10

El siguiente programa permite averiguar si un número es primo. Un número es primo cuando es divisible por 1 y por sí mismo.

Por ejemplo 4 no es primo, porque es divisible por 2, 5 si es primo, porque solo es divisible por 1 y por 5.

```
public class prog0616
{
   public static void main(String[] args)
   {
      int n = Integer.parseInt( args[0] );
      String msg = "si es primo"; // Se asume que si es primo

      for( int k = 2; k < n; k++ )
         if( n%k == 0 ) {
            msg = "no es primo";
            break;
         }

      System.out.println( "n: " + n );
      System.out.println( "Resultado: " + msg );
   }
}
```

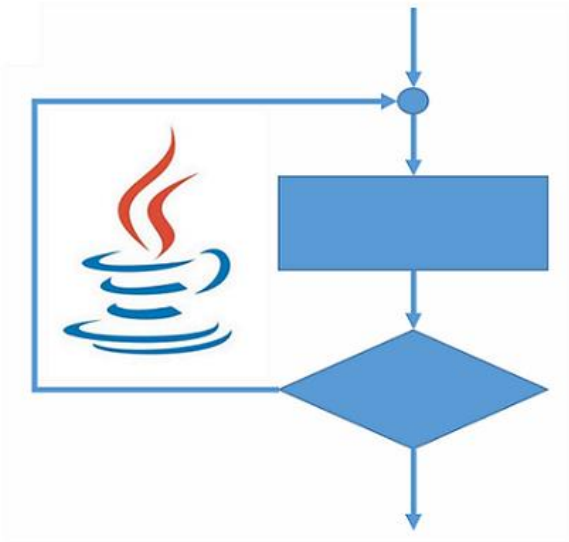
CURSOS VIRTUALES

CUPONES

En esta URL se publican cupones de descuento:

http://gcoronelc.github.io

FUNDAMENTOS DE PROGRAMACIÓN CON JAVA



Tener bases sólidas de programación muchas veces no es fácil, creo que es principalmente por que en algún momento de tu aprendizaje mezclas la entrada de datos con el proceso de los mismos, o mezclas el proceso con la salida o reporte, esto te lleva a utilizar malas prácticas de programación que luego te serán muy difíciles de superar.

En este curso aprenderás las mejores prácticas de programación para que te inicies con éxito en este competitivo mundo del desarrollo de software.

URL del Curso: https://n9.cl/gcoronelc-java-fund

Avance del curso: https://n9.cl/gcoronelc-fp-avance

Cupones de descuento: http://gcoronelc.github.io

JAVA ORIENTADO A OBJETOS

CURSO PROFESIONAL DE JAVA ORIENTADO A OBJETOS

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

En este curso aprenderás a crear software aplicando la Orientación a Objetos, la programación en capas, el uso de patrones de software y Swing.

Cada tema está desarrollado con ejemplos que demuestran los conceptos teóricos y finalizan con un proyecto aplicativo.

URL del Curso: https://bit.ly/2B3ixUW

Avance del curso: https://bit.ly/2RYGXIt

Cupones de descuento: http://gcoronelc.github.io

PROGRAMACIÓN CON JAVA JDBC

En este curso aprenderás a programas bases de datos Oracle con JDBC utilizando los objetos Statement, PreparedStatement, CallableStatement y a programar transacciones correctamente teniendo en cuenta su rendimiento y concurrencia.

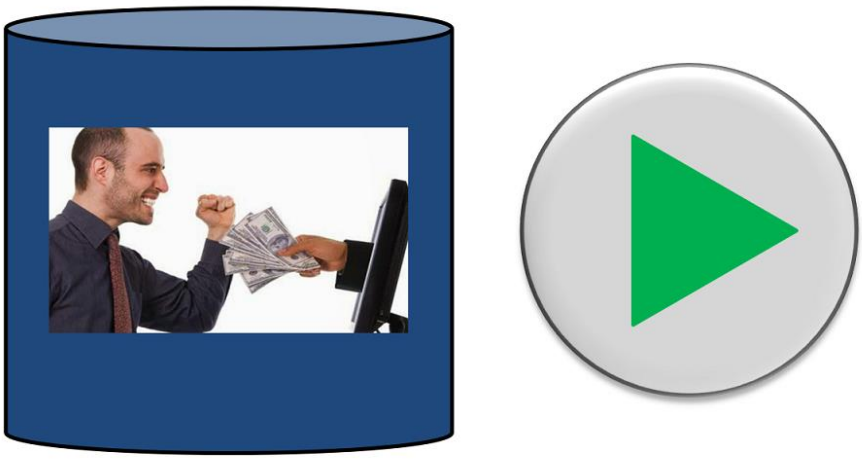
Al final del curso se integra todo lo desarrollado en una aplicación de escritorio.

URL del Curso: https://bit.ly/31apy0O

Avance del curso: https://bit.ly/2vatZOT

Cupones de descuento: http://gcoronelc.github.io

PROGRAMACIÓN CON ORACLE PL/SQL



En este curso aprenderás a programas las bases de datos ORACLE con PL/SQL, de esta manera estarás aprovechando las ventas que brinda este motor de base de datos y mejoraras el rendimiento de tus consultas, transacciones y la concurrencia.

Los procedimientos almacenados que desarrolles con PL/SQL se pueden ejecutarlo de Java, C#, PHP y otros lenguajes de programación.

URL del Curso: https://bit.ly/2YZjfxT

Avance del curso: https://bit.ly/3bciqYb

Cupones de descuento: http://gcoronelc.github.io