VARIABLES Y ESTRUCTURAS DE CONTROL EN LA PROGRAMACIÓN ORIENTADA A OBJETOS: JAVA

USO DE CICLOS DE REPETICIÓN EN JAVA

Servicio Nacional de Aprendizaje SENA

USO DE CICLOS DE REPETICIÓN EN JAVA

ESTRUCTURA DE CONTENIDOS

USO DE CICLOS DE REPETICIÓN EN JAVA

INTRODUCCIÓN

En este material de formación se estudiará la utilización y la sintaxis de los diferentes ciclos de repetición en el lenguaje Java; se trata del segundo tipo de instrucciones de mayor complejidad, las cuales también contribuyen al control del flujo de los programas, en este caso, relacionadas con la ejecución de órdenes repetitivas comunes en la escritura de este tipo de códigos.

Es necesario tener en cuenta que en las soluciones informáticas se presentan de manera frecuente cierto tipo de operaciones que deben ser resueltas en múltiples iteraciones, dependiendo de los valores almacenados por ciertas variables y de una comparación, normalmente lógica, que se realiza en torno a dichas variables; la ejecución de estas iteraciones puede ser ejecutada utilizando diferentes líneas de código; sin embargo, ello haría la codificación poco funcional y mal estructurada, por lo tanto, es necesario que el programador cuente con la habilidad requerida para manejar los ciclos de repetición.

1 CICLOS DE REPETICIÓN

Las estructuras de control denominadas ciclos de repetición o también conocidas como bucles, consisten en instrucciones cuyo objetivo principal es evitar la escritura repetitiva de líneas de código y controlar por medio de condiciones analizadas, la cantidad de veces que debe ser ejecutado un bloque de sentencias.

Existen tres tipos de ciclos de repetición que son el **for**, el **while** y el **do while**, los cuales brindan diferentes funcionalidades al programador; a continuación, se observa cada uno de ellos con su respectiva sintaxis y ejemplificación.

1.1 For

El ciclo de repetición **for** permite la ejecución de una sentencia o bloque de sentencias un número determinado de veces; se utiliza normalmente en aquellos casos en los que, desde el comienzo, se conoce cuantas veces se desea repetir la ejecución del bloque de instrucciones.

USO DE CICLOS DE REPETICIÓN EN JAVA

La sintaxis del ciclo **for** es la siguiente:

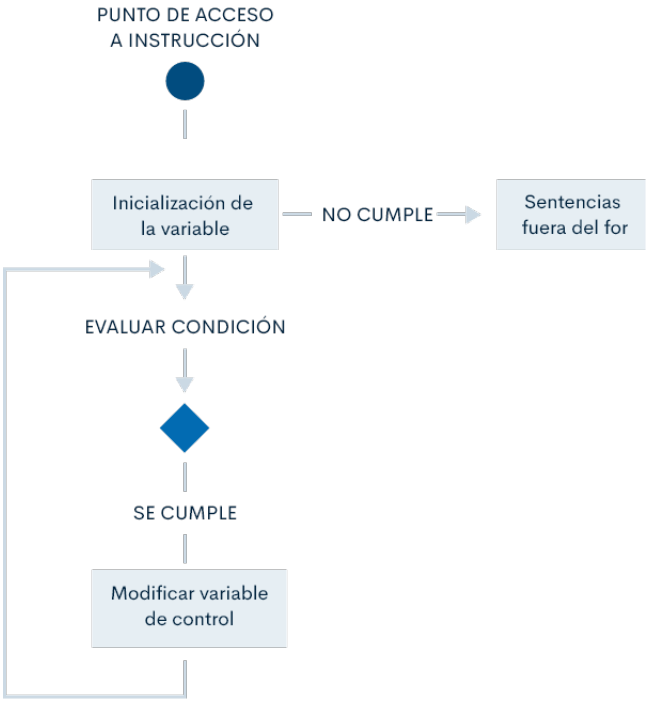
```
for (inicializacionVariable;condición;incremento) {
   sentencia1...
   sentencia2...
   sentenciaN
}
```

En la primera parte del **for** se inicializa la variable que va a controlar el ciclo y con la que se indica cuál va a ser el valor con que dicho ciclo comienza.

En la segunda parte se analiza la condición que debe cumplirse para que el bloque de sentencias continúe ejecutándose; en el momento que la condición deja de cumplirse, el bloque de sentencias no se ejecuta más.

Finalmente está la parte de modificación de la variable de control, que es la que indica si el valor de la variable inicializada debe aumentar o disminuir y con qué frecuencia, de uno en uno, de dos en dos, entre otros; este elemento es de gran importancia, ya que si el valor de la variable de control no cambia, el análisis de la condición siempre va a devolver el mismo resultado, lo cual llevaría a la ejecución de un ciclo infinito que en la mayoría de los casos ocupa recursos del sistema y lo bloquea.

Se ilustra a continuación el proceso efectuado cuando se utiliza un bucle **for**:



USO DE CICLOS DE REPETICIÓN EN JAVA

» **Ejemplos**

a Se requiere escribir cinco veces el texto: Ciclos de Repetición.

```
Source  History
1  public class Main
2  {
3      public static void main(String[] args)
4      {
5          for (int i=0;i<5;i++)
6          {
7              System.out.println("Ciclos de Repetición");
8          }
9      }
10 }
```

```
Salida - Aplicacion1 (run)
run:
Ciclos de Repetición
Ciclos de Repetición
Ciclos de Repetición
Ciclos de Repetición
Ciclos de Repetición
BUILD SUCCESSFUL (total time: 4 seconds)
```

Código y ejecución ejemplo a – for

El ejemplo anterior puede realizarse sin el uso del **for** de la siguiente manera:

```
Source  History
1  public class Main
2  {
3      public static void main(String[] args)
4      {
5          {
6              System.out.println("Ciclos de Repetición");
7              System.out.println("Ciclos de Repetición");
8              System.out.println("Ciclos de Repetición");
9              System.out.println("Ciclos de Repetición");
10             System.out.println("Ciclos de Repetición");
11         }
12     }
13 }
```

Código ejemplo a sin utilizar for

El resultado devuelto sería exactamente igual, nótese la mejora en la estructura del código, evitando la escritura repetitiva de instrucciones.

USO DE CICLOS DE REPETICIÓN EN JAVA

b. Se requiere mostrar los números pares entre el 100 y el 120.

```
Source  History

1   public class Main
2   {
3       public static void main(String[] args)
4       {
5           for (int i=100;i<=120;i+=2)
6           {
7               System.out.println(i);
8           }
9       }
10  }
```

```
Salida - Aplicacion1 (run)

run:
100
102
104
106
108
110
112
114
116
118
120
BUILD SUCCESSFUL (total time: 0 seconds)
```

Código y ejecución ejemplo b – for

Dado que el ejercicio solicita los números pares desde el 100, ese es el valor de inicialización de la variable del **for**, teniendo en cuenta que únicamente los solicita hasta el 120, se usa dicho valor como finalización del for y como son números pares, el incremento debe ir de 2 en 2, para lo cual se utiliza i+=2 que es lo mismo que decir i = i + 2.

USO DE CICLOS DE REPETICIÓN EN JAVA

c. Se desea mostrar en pantalla un conteo regresivo del diez al cero.

```
Source  History
1  public class Main
2  {
3      public static void main(String[] args)
4      {
5          for (int i=10;i>=0;i--)
6          {
7              System.out.println(i);
8          }
9      }
10 }
```

```
Salida - Aplicacion1 (run)
run:
10
9
8
7
6
5
4
3
2
1
0
BUILD SUCCESSFUL (total time: 1 second)
```

Código ejemplo c – for

Dado que el conteo es regresivo, la única diferencia consiste en que la variable en la tercera parte del **for**, en lugar de ser incrementada, se le efectúa decremento utilizando el operador - -.

USO DE CICLOS DE REPETICIÓN EN JAVA

d. Se desea simular el funcionamiento de un reloj por medio de ciclos for anidados.

```
Source  History
1   public class Main
2   {
3       public static void main(String[] args)
4       {
5           for (int a=0;a<24;a++)
6           {
7               for (int b=0;b<60;b++)
8               {
9                   for (int c=0;c<60;c++)
10                  {
11                      System.out.println(" "+a+": "+b+": "+c);
12                  }
13              }
14          }
15      }
16  }
```

```
Salida - Aplicacion1 (run)
   23: 59: 50
   23: 59: 51
   23: 59: 52
   23: 59: 53
   23: 59: 54
   23: 59: 55
   23: 59: 56
   23: 59: 57
   23: 59: 58
   23: 59: 59
BUILD SUCCESSFUL (total time: 29 seconds)
```

Código ejemplo d – for

Lo que ocurre con el **for** anidado es que se efectúa un conteo de 0 a 59 para las variables **b** y **c** que están simulando los minutos y los segundos, y de 0 a 23 para la variable a que maneja las horas; de este modo se van mostrando dichas horas, desde las 0:0:0 hasta las 23:59:59.

De acuerdo con la ubicación de los **for**, se observan primero las horas, luego los minutos y finalmente los segundos.

1.2 While

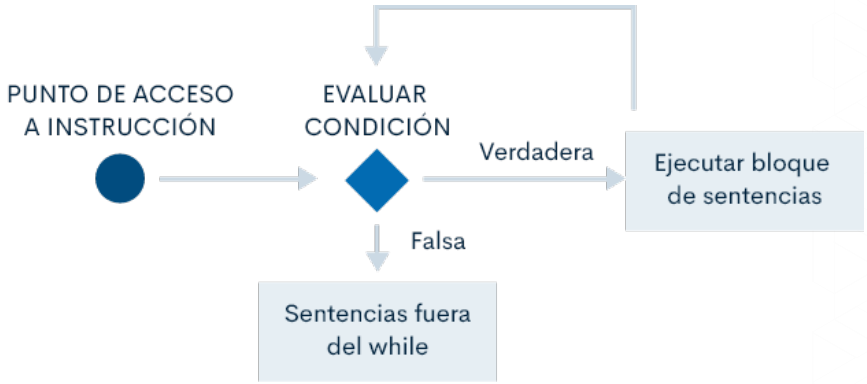
El bucle **while** permite la ejecución de un bloque de instrucciones siempre y cuando la condición analizada sea verdadera, si en el análisis inicial la condición es falsa, el bloque de instrucciones no se ejecuta nunca. Es importante que dentro de dicho bloque de instrucciones se incluya una que haga variar el resultado de la condición para que esta pueda llegar a ser verdadera y se ejecuten las instrucciones, o bien, pueda convertirse en falsa y que el ciclo finalice.

La sintaxis del ciclo **while** es la siguiente:

```
while (condición) {
   sentencia1...
   sentencia2...
   sentenciaN
}
```

Se puede apreciar que la condición adquiere un valor booleano que, al ser verdadero, se ingresa a las líneas de código contenidas en las llaves y se ejecutan, y si es falso, pasa a la primera instrucción que se encuentre por fuera del **while**.

Se ilustra a continuación, el proceso efectuado cuando se utiliza un bucle **while**



» **Ejemplos**

a. Se requiere un programa que pida al usuario un número entero e imprima todos los números existentes entre 0 y dicho número.

USO DE CICLOS DE REPETICIÓN EN JAVA

```java
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        int a=0, numero;

        Scanner s = new Scanner(System.in);
        System.out.println("Introduzca un número entero: ");
        numero = s.nextInt();

        while(a<=numero)
        {
            System.out.println(a);
            a++;
        }
    }
}
```

Código ejemplo a – while

Se declara la variable a que hace las veces de variable de control en el **while** y la variable número que será el dato ingresado por el usuario; el programa entonces imprime los números desde 0, que es el valor de inicialización que se le dio a a, hasta el número indicado por el usuario.

b. Se necesita encontrar el área de una esfera solicitando al usuario que ingrese el valor del radio; en caso de que el valor ingresado sea inferior o igual a cero, se debe devolver un mensaje y pedir nuevamente el dato.

USO DE CICLOS DE REPETICIÓN EN JAVA

```java
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        double radio, area;

        Scanner s = new Scanner(System.in);
        System.out.println("Introduzca el radio de la esfera: ");
        radio = s.nextDouble();

        while(radio<=0)
        {
            System.out.println("El radio debe ser mayor que cero");
            System.out.println("Introduzca el radio de la esfera: ");
            radio = s.nextDouble();
        }
        area = 4*Math.PI*Math.pow(radio,2);
        System.out.println("El área de la esfera es: " + area);

    }
}
```

Código ejemplo b – while

Se declaran las variables radio y área, la primera es el valor ingresado por el usuario y la segunda para efectuar el cálculo cuando el valor del radio cumpla con lo indicado.

Si se ingresa como radio el valor 0, el programa pide nuevamente el dato, en caso contrario, el programa calcula el área; la ejecución se observa en la siguiente figura:

```
Salida - Aplicacion1 (run)
run:
Introduzca el radio de la esfera:
0
El radio debe ser mayor que cero
Introduzca el radio de la esfera:
0
El radio debe ser mayor que cero
Introduzca el radio de la esfera:
0
El radio debe ser mayor que cero
Introduzca el radio de la esfera:
2
El área de la esfera es: 50.26548245743669
BUILD SUCCESSFUL (total time: 13 seconds)
```

Ejecución código ejemplo b – while

c. Se requiere un programa que pida al usuario la medida en centímetros del lado de un cuadrado y calcule su área, siempre y cuando el valor ingresado sea mayor que cero; en caso de que dicha condición no se cumpla, ya no se solicitarán más valores y se debe mostrar la cantidad de áreas que han sido calculadas.

USO DE CICLOS DE REPETICIÓN EN JAVA

```java
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        int contador=0, lado, area;

        Scanner s = new Scanner(System.in);
        System.out.println("Introduzca el lado del cuadrado: ");
        lado = s.nextInt();

        while(lado>0)
        {
            area = lado*lado;
            System.out.println("El área del cuadrado es: " + area);
            contador ++;
            System.out.println("Introduzca el lado del cuadrado: ");
            lado = s.nextInt();
        }
        System.out.println("Se ha calculado el área de " + contador + " cuadrados");
    }
}
```

Código ejemplo c – while

Se declaran las variables contador, lado y área; la primera va a guardar la cantidad de veces que el área sea calculada, la segunda almacena el valor ingresado por el usuario y la tercera para efectuar el cálculo respectivo, cada que el lado cumpla con lo indicado.

En el **while** se indica que siempre que el valor ingresado por el usuario sea mayor a cero, se realice el cálculo, en caso contrario, el programa ejecuta la instrucción que hay inmediatamente después de terminar el **while** e imprime la cantidad de áreas que han sido calculadas. En la siguiente figura se puede observar la ejecución de este ejemplo.

```
Salida - Aplicacion1 (run) #2
run:
Introduzca el lado del cuadrado:
3
El área del cuadrado es: 9
Introduzca el lado del cuadrado:
5
El área del cuadrado es: 25
Introduzca el lado del cuadrado:
11
El área del cuadrado es: 121
Introduzca el lado del cuadrado:
0
Se ha calculado el área de 3 cuadrados
BUILD SUCCESSFUL (total time: 19 seconds)
```

Ejecución código ejemplo c – while

d. Se requiere un programa que solicite al usuario un número entero y muestre en pantalla la tabla de multiplicar de dicho número.

USO DE CICLOS DE REPETICIÓN EN JAVA

```java
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        int contador=1, resultado, numero;

        Scanner s = new Scanner(System.in);
        System.out.println("Introduzca la tabla deseada: ");
        numero = s.nextInt();

        while(contador<=10)
        {
            resultado = numero*contador;
            System.out.println(numero + "x" + contador + "=" + resultado);
            contador ++;
        }
    }
}
```

Código ejemplo d – while

Se declaran las variables contador, resultado y número. La primera es la variable de control del **while** que indica que van a mostrarse los datos de la tabla desde el 1 hasta el 10, la segunda para efectuar el cálculo respectivo, y la tercera, recibe el valor ingresado por el usuario que determina la tabla que va a ser mostrada.

En el **while** se da la instrucción que se realice el cálculo hasta 10, y posteriormente se muestra la tabla. En la siguiente figura se puede observar la ejecución de este ejemplo.

```
Salida - Aplicacion1 (run) #2
run:
Introduzca la tabla deseada:
3
3x1=3
3x2=6
3x3=9
3x4=12
3x5=15
3x6=18
3x7=21
3x8=24
3x9=27
3x10=30
BUILD SUCCESSFUL (total time: 5 seconds)
```

Ejecución código ejemplo d – while

1.3 Do while

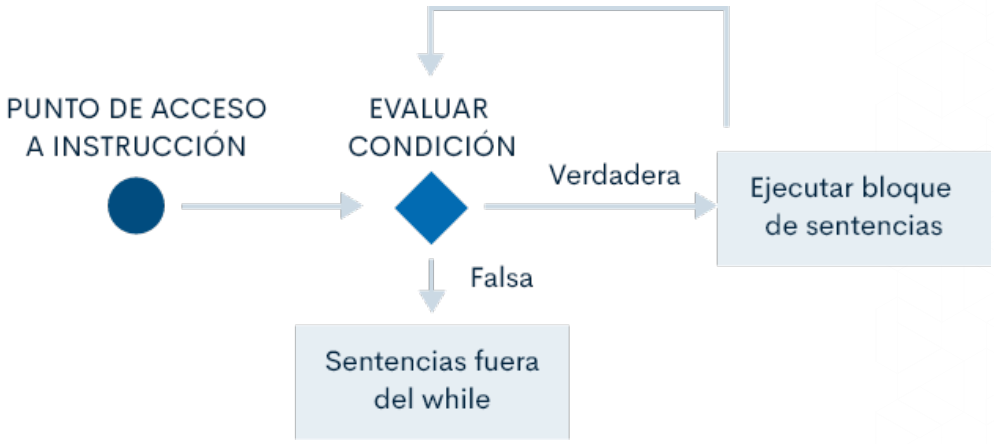
El ciclo **do while** también es utilizado para ejecutar repetidamente unas instrucciones, difiere del **while** en el hecho de que con **do while** el bloque de instrucciones siempre va a ser ejecutado al menos una vez, dado que primero se lleva a cabo dicha ejecución y posteriormente se evalúa la condición; si se cumple, continúan ejecutándose las instrucciones hasta que no sea así, y si no se cumple, no se ejecutan más.

La sintaxis del ciclo **do while** es la siguiente:

```
do {
   sentencia1...
   sentencia2...
   sentenciaN
} while (condicion);
```

La condición adquiere un valor booleano que, al ser verdadero, se continúan ejecutando las sentencias que hay dentro de las llaves, y si es falso, pasa a la primera instrucción que se encuentre por fuera del **do while**.

Se ilustra a continuación, el proceso efectuado cuando se utiliza un bucle **do while**.



USO DE CICLOS DE REPETICIÓN EN JAVA

» Ejemplos

a. Solicitar al usuario que digite un número del 1 al 10 y mostrarlo en pantalla; se debe validar que el número ingresado sí se encuentra dentro del rango indicado, en caso de que no lo esté, se pide nuevamente.

```
Source  History
1  import java.util.Scanner;
2
3  public class Main
4  {
5      public static void main(String[] args)
6      {
7          int numero;
8          Scanner s = new Scanner(System.in);
9
10         do
11         {
12             System.out.println("Digite un número entre 1 y 10: ");
13             numero = s.nextInt();
14         } while (numero < 0 || numero > 10);
15
16         System.out.println("El número ingresado fue: " + numero);
17     }
18 }
```

```
Salida - Aplicacion1 (run)
run:
Digite un número entre 1 y 10:
-5
Digite un número entre 1 y 10:
21
Digite un número entre 1 y 10:
17
Digite un número entre 1 y 10:
5
El número ingresado fue: 5
```

Solución ejemplo a - do while

Se declara la variable número que recibe el dato ingresado por el usuario, el do indica que se pida dicho dato al usuario, y en el **while** se valida si cumple con el rango requerido; cuando no lo cumple, se pide nuevamente el número, y cuando lo cumple, se muestra el número ingresado.

USO DE CICLOS DE REPETICIÓN EN JAVA

b. Se requiere un programa que solicite al usuario el ingreso de varios números y que calcule el promedio de estos cuando el número ingresado sea 0, dicho número no debe ser tenido en cuenta en la cantidad de datos para promediar.

```java
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        //variable cantidad para almacenar la cantidad de números ingresados
        int cantidad=0;
        /*total para almacenar la suma
          valor recibe los datos que ingresa el usuario
          promedio almacena el promedio calculado
        */
        double total=0, valor, promedio;
        Scanner s = new Scanner(System.in);

        do
        {
            /*en el do se pide al usuario el ingreso del número y se almacena
              éste en la variable valor
            */
            System.out.println("Ingrese un número-Presione 0 para terminar: ");
            valor = s.nextDouble();
            /*se incluye un if que valida si el valor es diferente de 0 y si
             lo es, realiza la sumatoria e incrementa la cantidad
            */
            if (valor!=0)
            {
                total = total + valor;
                cantidad ++;
            }
        /*en el while se indica que todo lo anterior se realiza. únicamente
          mientras el valor ingresado sea diferente de cero
        */
        } while (valor!=0);
        /*ya por fuera del do while con otro if, se evalúa que la cantidad de
        números ingresados sea diferente de cero
        */
        if (cantidad!=0)
        {
            //se calcula el promedio y se muestra en pantalla
            promedio = total / cantidad;
            System.out.println("El promedio es: ");
            System.out.println(promedio);
        //con el else, si la cantidad de números fue cero, se muestra el mensaje
        } else {
            System.out.println("No se realizó el ingreso de ningún valor");
        }
    }
}
```

Código ejemplo b – do while

USO DE CICLOS DE REPETICIÓN EN JAVA

En la siguiente figura se observa la ejecución del ejemplo anterior.

```
Salida - Aplicacion1 (run)
run:
Ingrese un número - Presione 0 para terminar:
6
Ingrese un número - Presione 0 para terminar:
5
Ingrese un número - Presione 0 para terminar:
4
Ingrese un número - Presione 0 para terminar:
8
Ingrese un número - Presione 0 para terminar:
9
Ingrese un número - Presione 0 para terminar:
0
El promedio es:
6.4
BUILD SUCCESSFUL (total time: 12 seconds)
```

Ejecución código ejemplo b – while

c. Se necesita un programa que solicite las notas de estudiantes hasta que el valor ingresado sea 0, y posterior a ello muestre en pantalla cuántos obtuvieron una nota igual o superior a 6, y cuántos menor.

Se declara la variable nota que recibe las notas ingresadas por el usuario y se utilizan dos contadores llamados cant1 y cant2 para almacenar el total de estudiantes que obtuvieron notas inferiores y superiores a 6.

En el do se solicita el ingreso de cada nota y por medio de un if se suma en cada contador respectivo la nota que haya sido ingresada, y en el **while** se indica que el ciclo debe terminar cuando se ingrese como nota el valor cero.

Finalmente se muestran los resultados que indican cuántos estudiantes obtuvieron notas superiores e inferiores a 6.

En la siguiente figura se observa el código de este ejemplo y luego su correspondiente ejecución.

```
Source  History
1  import java.util.Scanner;
2  public class Main
3  {
4      public static void main(String[] args)
5      {
6         int nota=0, cant1=0, cant2=0;
7         Scanner s = new Scanner(System.in);
8
9         do
10        {
11            System.out.println("Ingrese la nota del estudiante: ");
12            nota = s.nextInt();
13
14            if(nota>=6)
15            {
16               cant1 += 1;
17            } else if(nota>0){
18               cant2 += 1;
19            }
20        } while (nota != 0);
21        System.out.println("Los estudiantes con nota mayor o igual que 6 son: " +
22                cant1);
23        System.out.println("Los estudiantes con nota menor que 6 son: " +
24                cant2);
25    }
26 }
```

Código ejemplo c – do while

USO DE CICLOS DE REPETICIÓN EN JAVA

```
Salida - Aplicacion1 (run)
run:
Ingrese la nota del estudiante:
5
Ingrese la nota del estudiante:
7
Ingrese la nota del estudiante:
9
Ingrese la nota del estudiante:
1
Ingrese la nota del estudiante:
10
Ingrese la nota del estudiante:
0
Los estudiantes con nota mayor o igual que 6 son: 3
Los estudiantes con nota menor que 6 son: 2
BUILD SUCCESSFUL (total time: 5 seconds)
```

Ejecución ejemplo c – do while

d. Se desea un programa que solicite al usuario la contraseña de acceso y únicamente le dé la bienvenida al sistema cuando esta sea correcta.

```
Source  History
1  import java.util.Scanner;
2  public class Main
3  {
4      public static void main(String[] args)
5      {
6          int clave;
7          Scanner s = new Scanner(System.in);
8
9          do
10         {
11             System.out.println("Digite su clave: ");
12             clave = s.nextInt();
13         } while (clave != 3520);
14         System.out.println("Bienvenido - Datos Correctos - Ingreso en Proceso");
15     }
16 }
Salida - Aplicacion1 (run)
run:
Digite su clave:
1017
Digite su clave:
0317
Digite su clave:
3520
Bienvenido - Datos Correctos - Ingreso en Proceso
BUILD SUCCESSFUL (total time: 12 seconds)
```

Solución ejemplo d - do while

GLOSARIO

Bucle: Nombre con el cual también se conocen los ciclos de repetición.

Ciclo infinito: Ciclo de repetición cuya variable de control no cambió de estado y por lo tanto el ciclo nunca finalizó.

Valor booleano: Tipo de valor que toman algunas variables, definido como falso o verdadero.

Variable de control: Valor que garantiza que cada ciclo de repetición pueda tener un final.

REFERENCIAS BIBLIOGRÁFICAS

NetBeans. (2019). NetBeans IDE 8.2 Download. Recuperado de https://netbeans.org/downloads/8.2/

CRÉDITOS

Equipo Contenido Instruccional		
» **Gloria Matilde Lee Mejía**	Responsable equipo	Centro de Comercio y Servicios – Regional Tolima
» **Rafael Nelftalí Lizcano Reyes**	Asesor pedagógico	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Lucero Montes Arenas**	Gestora de Desarrollo de Programas	Centro para la Formación Cafetera (Caldas)
» **Julio Alexander Rodriguez Del Castillo**	E-pedagogo instruccional	Centro Atención Sector Agropecuario Regional Risaralda
» **Rachman Bustillo Martínez**	Evaluador de contenido	Centro Atención Sector Agropecuario Regional Risaralda
» **Natalia Andrea Bueno Pizarro**	Diseñadora instruccional	Centro de Diseño y Metrología - Regional Distrito Capital
Equipo Diseño y Desarrollo		
» **Francisco José Lizcano Reyes**	Responsable Equipo	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Daniel Ricardo Mutis Gómez**	Diagramación web	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Yazmin Rocio Figueroa Pacheco**	Construcción documentos digitales	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Edgar Mauricio Cortes García**	Desarrollo front-end	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Luis Gabriel Urueta Álvarez**	Desarrollo de actividades didácticas	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» **Leyson Fabian Castaño Pérez**	Integración de recursos y pruebas	Centro Industrial Del Diseño y La Manufactura - Regional Santander

Equipo de Gestores de Repositorio

» **Kely Alejandra Quiros Duarte**	Administrador repositorio de contenidos y gestores de repositorio.	Centro de comercio y servicios – Regional Tolima

Recursos gráficos

Fotografías y vectores tomados de www.shutterstock.com y www.freepik.com