Ciclos

Ciclos 1
- ¿Qué aprenderás? 2
- Introducción 2
- Uso de ciclos 3
- Estructuras Iterativas 3
  - Bloque While 3
    - While paso a paso 4
    - Salida del ciclo 4
    - Validación de entrada de datos utilizando while 5
    - Ejercicio de Integración 6
  - Bloque DO While 6
  - Bloque FOR 7
- Ejercicio guiado: Menú 8
  - Solución 8

¡Comencemos!

¿Qué aprenderás?

- Clasificar los ciclos y sus posibles aplicaciones en la programación.
- Construir programas Java con iteraciones a partir de diagramas de flujo.

Introducción

Los ciclos son estructuras de control que nos permiten repetir la ejecución de una o más instrucciones.

```
Mientras se cumpla una condición:
    Instrucción 1
    Instrucción 2
    Instrucción 3
```

El uso de estructuras de control, como los ciclos, es la clave para crear programas avanzados.

¡Vamos con todo!

Uso de ciclos

Los posibles usos de ciclos en algoritmos son infinitos, nos permiten recorrer colecciones de datos o espacios de búsqueda.

- Si necesitamos buscar una palabra en un diccionario, debemos recorrer palabra por palabra hasta encontrar la que buscamos.
- Si necesitamos resolver un acertijo, debemos repasar cada una de las pistas hasta lograr obtener el resultado.
- Si necesitamos sumar el monto total de una factura, debemos sumar el costo de ítem a ítem para obtener el monto total.

En esta unidad estudiaremos distintos problemas que se resuelven con ciclos y que, si bien no siempre son utilizados en la industria, nos ayudarán a desarrollar las habilidades lógicas que necesitamos para ser buenos programadores y cuándo utilizarlos correctamente.

Estructuras Iterativas

En esta unidad veremos el uso de distintas estructuras de control que se pueden utilizar en el mundo de Java, las cuáles son las siguientes:

- While
- Do While
- For

Comenzaremos estudiando la sentencia while.

Bloque While

La instrucción while nos permite ejecutar una o más operaciones mientras se cumpla una condición. Su sintaxis es la siguiente:

```
while(condición){
    //Código que se ejecuta
}
```

While paso a paso

1. Se evalúa la condición; si es `true`, ingresa al ciclo.
2. Se ejecutan, secuencialmente, las instrucciones definidas dentro del ciclo.
3. Una vez ejecutadas todas las instrucciones se vuelve a evaluar la condición:

    a. Si se evalúa como `true` : vuelve a repetir.
    b. Si se evalúa como `false` : sale del ciclo.

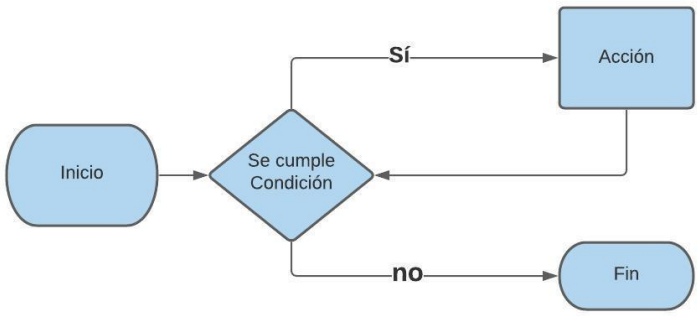

Imagen 1. Diagrama de flujo ciclo While.
Fuente: Desafío Latam.

Salida del ciclo

Anteriormente aprendimos que un algoritmo es una secuencia de pasos finita para resolver un problema. El ciclo while termina cuando alguna de las instrucciones no cumple la condición definida inicialmente.

Validación de entrada de datos utilizando while

Un ejemplo común para comenzar a estudiar los ciclos es validar la entrada de un dato, es decir, que este dato cumpla un criterio. Podemos, por ejemplo, validar que el usuario ingrese un número entre 1 y 10:

```
Scanner sc = new Scanner(System.in);
System.out.printf("Ingresa un número del 1 al 10: ");
int num = sc.nextInt();
while(num <1 || num > 10) {
    System.out.printf("El número no está entre 1 y 10\n");
    System.out.printf("Ingresa un número del 1 al 10: ");
    num = sc.nextInt();
}
System.out.printf("El número ingresado fue: %d \n",num);
```

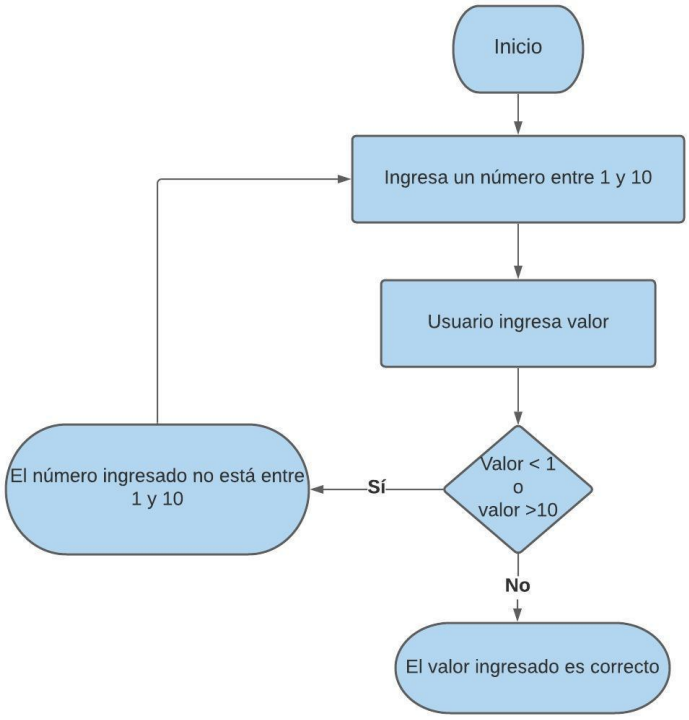

Imagen 2. Diagrama de flujos ingresando un valor entre 1 y 10.
Fuente: Desafío Latam.

Ejercicio de Integración

Podríamos utilizar la misma idea de validación para impedir al usuario entrar, hasta que ingrese la contraseña correcta, para este ejemplo la contraseña será "password".

```
Scanner sc = new Scanner(System.in);
System.out.printf("Ingrese la contraseña: ");
String password = sc.nextLine();
while(password.compareTo("password")!=0) {
    System.out.printf("La contraseña es incorrecta\n", password);
    System.out.printf("Ingrese la contraseña: ");
    password = sc.nextLine();
}
System.out.printf("La contraseña ingresada es correcta\n");
```

Bloque DO While

También existe la sentencia do while, que funciona al inverso del while. En este caso haremos una o más acciones, mientras se cumpla la condición. Las instrucciones se ejecutarán al menos una vez.

```
do{
    //acciones
} while (condición);
```

En el ejemplo de pedir un número del 1 al 10, quedaría de la siguiente manera:

```
int num;
do {
    System.out.printf("ingrese un numero entre 1 y 10:");
    num = sc.nextInt();
}while(num <1 || num > 10);
System.out.printf("El número ingresado es: %d\n",num);
```

Bloque FOR

Este bloque iterativo permite repetir una serie de instrucciones hasta que se cumpla un número determinado de repeticiones. Por lo tanto, el bloque `for` se compone por tres partes:

- Inicio o variable inicializadora.
- Condición, contar mientras la variable inicio sea “menor” o “menor igual a”…
- Cada cuantos pasos se avanza la variable inicializadora.

Lo sintaxis del ciclo `for` es la siguiente:

```
/*
   i: Variable inicializadora
   i<=x: Condición, el ciclo se repetira hasta que i sea mayor a x
   i++: Indica que i se incrementará de 1 en 1
*/
for(int i = 0;i<=x;i++) {
    System.out.println("Instrucciones");
}
```

Existen varias formas de aplicar el bucle `for`, algunas de estas son:

```
//Repite hasta 1 > 10
for(int i = 0;i<=10;i++) {
    System.out.println("Instrucciones 1 en 1");
}
//Repite hasta i > 10, pero salta de dos en dos
for(int i = 0;i<=10;i+=2) {
    System.out.println("Instrucciones 2 en 2");
}
//Repite hasta que i sea menor que cero
for(int i = 10;i>=0;i--) {
    System.out.println("Instrucciones 1 en 1 en reversa");
}
```

Ejercicio guiado: Menú

Podemos implementar de forma sencilla un menú de opciones para el usuario. La lógica es similar a la de la validación de entrada.

- Se muestra un texto con opciones.
- El usuario tiene que ingresar una opción válida -> validación de entrada.
- Si el usuario ingresa la opción `1`, mostramos un texto.
- Si el usuario ingresa la opción `2`, mostramos otro texto.
- Si el usuario ingresa la opción "`salir`" terminamos el programa.

Solución

1. Diagrama de flujo

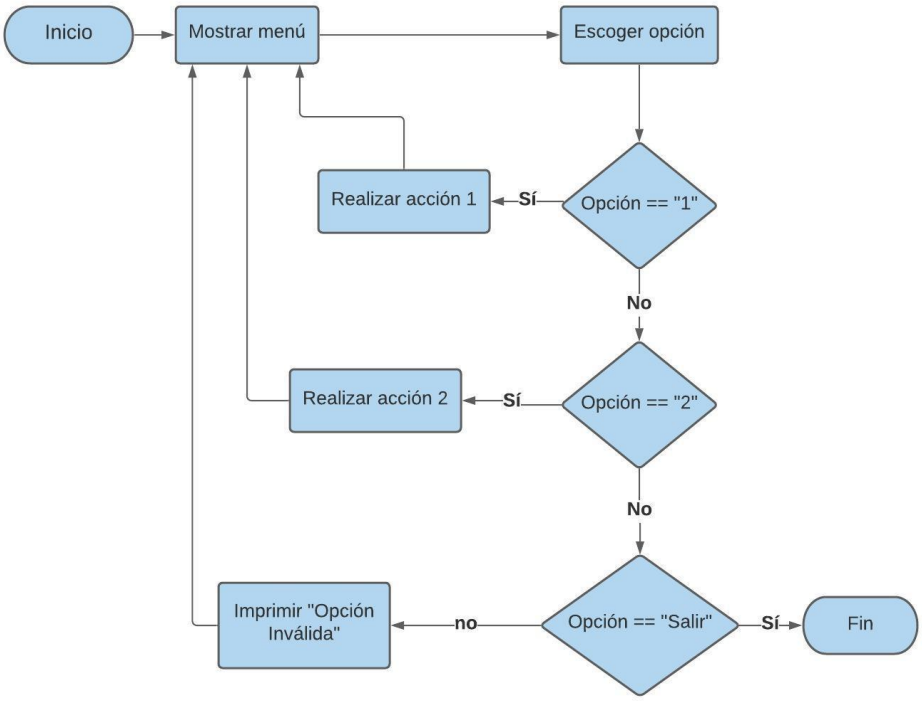

Imagen 3. Diagrama de flujo Solución Menú.
Fuente: Desafío Latam.

2. Código

```
Scanner sc = new Scanner(System.in);
String opcion = "";
//.equals devuelve verdadero (True) si son iguales, y falso (False) si los valores a comparar no son iguales
while(!opcion.equals("salir")) {
    System.out.printf("Escoge una opción\n");
    System.out.printf("1 -- Acción 1\n");
    System.out.printf("2 -- Acción 2\n");
    System.out.printf("Escribe 'salir' para terminar el programa\n\n");
    System.out.printf("Ingrese una opción:");
    opcion = sc.nextLine();
    if(opcion.equals("1")) {
        System.out.printf("Realizando acción 1\n");
    } else if(opcion.equals("2")) {
        System.out.printf("Realizando acción 2\n");
    } else if(opcion.equals("salir")) {
        System.out.printf("Saliendo...\n");
    } else {
        System.out.printf("Opción inválida\n");
    }
}
```