Taller de preparación para la OCI - PUC

While

1. Introducción

En este capítulo se tratará el manejo de ciclos utilizando `while`. Este es un elemento fundamental en cualquier lenguaje de programación.

Cuando estamos programando nos gustaría que algunas instrucciones se ejecuten una y otra vez **mientras** se cumplen un conjunto de condiciones. Por ejemplo, pensemos en que tenemos que caminar para llegar al colegio. Mientras aún no haya llegado al colegio, debo seguir caminando, suponiendo que no nos gustaría quedar detenidos camino a la escuela y llegar atrasados.

En esta parte del curso aprenderemos a dar instrucciones en nuestros programas que permitan repetir bloques de código. Hay que destacar que ahora debemos ser muy cuidadosos, ya que a veces nuestros programas se pueden quedar dentro de un ciclo hasta el infinito.

2. While

En Java el `while` va acompañado de una condición booleana y un bloque de código. El bloque de código se ejecutará mientras la condición booleana se cumple. Normalmente en el bloque de código se actualizan las variables que participan de la condición booleana. La estructura es la siguiente:

```
1  while (<condición>) {
2    // bloque de código que se ejecuta
3    // mientras se cumpla la condición
4  }
```

Partamos con un ejemplo básico. Definiremos una variable entera $n > 0$ e imprimiremos todos los números entre 0 y n, incluyendo ambos números.

Ejemplo 2.1. Dada una variable entera n, imprima los números entre 0 y n incluyendo ambos números.

```
1  int n = 10;
2
3  int contador = 0;
4  while (contador <= n) {
5    System.out.println("Vamos en el número: " + contador);
6    contador = contador + 1;
7  }
8
9  System.out.println("Terminó el programa!");
```

En el ejemplo 2.1 partimos definiendo la variable `n` hasta donde queremos llegar. Luego iniciamos un contador en 0. Luego al entrar al `while` comprobamos que `contador` sea menor o igual que `n`. Como esto se cumple, se ejecuta el código al interior del `while`. Este código imprime el valor del contador y luego lo incrementa en 1. Después de ejecutar este bloque de código se vuelve a comprobar si `contador` es menor o igual que `n`. Como `contador` aumenta en cada iteración, llegará un momento en que debemos salir del `while`, porque **ya no se va a cumplir la condición** `contador <= n`.

Es importante destacar que dentro del `while` podemos usar bloques más complejos, como por ejemplo bloques de código que usan `if - else`. Veamos un ejemplo:

Ejemplo 2.2. Dada una variable entera n, imprima los números pares entre 0 y n incluyendo ambos números.

```
1  int n = 10;
2
3  int contador = 0;
4  while (contador <= n) {
5    if (contador%2 == 0) {
6      System.out.println("Encontré un par! Su valor es: " + contador);
7    }
8    contador = contador + 1;
9  }
10
11 System.out.println("Terminó el programa!");
```

De este ejemplo destacamos que el incremento del contador se hace **fuera** del `if`, porque queremos que el contador **siempre** incremente, aunque el número no sea par.

Por último como ya dijimos anteriormente, hay que tener cuidado, porque un programa puede quedar en un loop por siempre. Por ejemplo este programa:

```
1  while (true) {
2    System.out.println("Hola");
3  }
```

Imprimirá `"Hola"` por siempre.

3. Break

En Java tenemos la instrucción `break`. Esta instrucción nos permite salir de un ciclo. Cuando el programa está dentro de un `while` y lee un `break`, sale inmediatamente del ciclo. Por ejemplo, veamos el ejemplo que imprime los números desde el 0 hasta el n, pero que utiliza `break` para salir del ciclo.

Ejemplo 3.1. Dada una variable entera n, imprima los números entre 0 y n incluyendo ambos números. Utilice la instrucción `break` para salir del ciclo.

```
1  int n = 10;
2
3  int contador = 0;
4  while (true) {
5    System.out.println("Vamos en el número: " + contador);
6    contador = contador + 1;
7    if (contador == n) {
8      break;
9    }
10 }
11
12 System.out.println("Terminó el programa!");
```

Notamos que el ejemplo 3.1 parte con un `while(true)`, que sabemos que se debería ejecutar para siempre. Sin embargo, este programa contiene un `if` que en cada iteración comprueba si el contador es igual a `n`. En el momento que esto sucede, el programa entra al bloque de código contenido por el `if` y leerá la instrucción `break`. En ese momento el programa saldrá del ciclo e imprimirá `"Terminó el programa!"`

4. Ejemplos

A continuación presentamos algunos ejemplos resueltos.

Ejemplo 4.1. Haga un programa en el que se parta definiendo una variable `n` que representa un entero tal que $n > 0$. Para cada número entre 0 y n (incluyéndolos) debe imprimir si el número es par o impar.

```
1  public class Main {
2    public static void main(String[] args) {
3
4      int n = 10;
5
6      int contador = 0;
7      while (contador <= n) {
8        if (contador%2 == 0) {
9          System.out.println("Encontré un número par: " + contador);
10       }
11       else {
12         System.out.println("Encontré un número impar: " + contador);
13       }
14     }
15
16   }
17 }
```

Ejemplo 4.2. En este ejemplo veremos cómo utilizar `while` para validar un input. Haga un programa que reciba dos números a y b. Luego debe recibir un tercer dígito c que representa la operación a realizar entre a y b. Para sumar c debe ser 1, para restar c debe ser 2, para multiplicar c debe ser 3 y para dividir c debe ser 4. Debe validar que c tenga alguno de esos valores.

```
1  import java.util.Scanner;
2
3  public class Main {
4    public static void main(String[] args) {
5
6    Scanner scanner = new Scanner(System.in);
7    System.out.println("Ingrese a: ");
8    double a = scanner.nextDouble();
9
10   System.out.println("Ingrese b: ");
11   double b = scanner.nextDouble();
12
13   System.out.println("Ingrese la operación: ");
14   int c = scanner.nextInt();
15
16   while (c<1 || c>4) {
```

```
17      System.out.println("Operación inválida!\nIngrese de nuevo: ");
18      c = scanner.nextInt();
19    }
20
21    if (c==1) {
22      System.out.println("Resultado: " + (a+b));
23    }
24    else if (c==2) {
25      System.out.println("Resultado: " + (a-b));
26    }
27    else if (c==3) {
28      System.out.println("Resultado: " + (a*b));
29    }
30    else if (c==4) {
31      System.out.println("Resultado: " + (a/b));
32    }
33
34    }
35  }
```

En la solución notamos que el programa entrará al `while` en el caso de que `c` sea menor que 1 o mayor que 4. En ese caso, se pedirá al usuario que vuelva a ingresar `c`. Mientras `c` no esté en el rango deseado el número se seguirá pidiendo.

Ejemplo 4.3. En este ejemplo veremos un programa que recibirá valores hasta que el usuario lo desee. Pida al usuario números positivos. El programa debe parar cuando el usuario ingresa algún valor negativo. El programa debe mostrar el promedio de todos los números positivos ingresados.

```
1   import java.util.Scanner;
2
3   public class Main {
4     public static void main(String[] args) {
5
6     Scanner scanner = new Scanner(System.in);
7     double numero = 0;
8     double total = 0;
9     double contador = 0;
10
11    while (true) {
12      System.out.println("Ingrese número: ");
13      numero = scanner.nextDouble();
14      if (numero >= 0) {
15        total = total + numero;
16        contador += 1;
17      }
18      else {
19        break;
20      }
21    }
22    double promedio = total / contador;
23    System.out.println("El promedio de los números es: " + promedio);
24
25    }
26  }
```

En el ejemplo partimos inicializando las variables:

- `numero`: representa el número que va a ir ingresando el usuario en cada iteración.
- `total`: llevará registro de la suma total de todos los números ingresados.
- `contador`: llevará el registro de cuantos números hemos ingresado.

Se pedirán valores al usuario hasta ingresar un número negativo, porque el programa leerá el `break` cuando el número ingresado sea negativo. En ese momento el programa saldrá del ciclo. Hasta entonces el total y el contador se habrán actualizado en cada iteración. Al finalizar el *loop* mostramos el resultado del total sumado dividido en el total de números ingresados.

Ejemplo 4.4. En este ejemplo veremos un programa que recibirá un input más complejo. Se hará un programa que sume varios pares de números. Se debe pedir al usuario que ingrese un número n que represente cuantos pares de números desea sumar. Se debe pedir al usuario n veces dos números a y b. Para cada par de números a y b se debe imprimir la suma.

```
1  import java.util.Scanner;
2
3  public class Main {
4    public static void main(String[] args) {
5
6    Scanner scanner = new Scanner(System.in);
7
8    System.out.println("Cuantos pares de números desea sumar? ");
9    int n = scanner.nextInt();
10
11   int contador = 0;
12
13   int a;
14   int b;
15   while (contador < n) {
16     System.out.println("Ingrese a: ");
17     a = scanner.nextInt();
18     System.out.println("Ingrese b: ");
19     b = scanner.nextInt();
20     int suma = a+b;
21     System.out.println("La suma vale: " + suma);
22
23     contador += 1;
24   }
25   }
26 }
```