Módulo 6

Ciclos

Ciclos o Loops

Loops

Un bucle / ciclo / loop, es una sección de código diseñada para repetirse. Por lo general, cuenta con tres secciones:

1. Cuerpo [Body]: ¿Qué quiero repetir?
2. Variable[s] de Control [Control variable]: ¿Qué variable va a controlar la repetición?
3. Condición[es] de Salida [Exit condition]: ¿Cuántas veces lo queremos repetir?

```java
public class Print5 {
    public static void main(String[] args) {
        System.out.println("0");
        System.out.println("1");
        System.out.println("2");
        System.out.println("3");
        System.out.println("4");
    }
}
```

Ambos programas son equivalentes, pero ¿si quisiéramos imprimir 100 números? ¿Qué modificaciones haríamos a cada uno?

```java
public class Loop5 {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println(i);
            i++;
        }
    }
}
```

While

Ciclos

Un ciclo while se va a repetir mientras la expresión boleana entre paréntesis bolean_condition condición se cumpla.

Utilizamos la siguiente sintaxis:

```
while(boolean_condition){
  //
  //code block to be executed
  //
}
```

Ejemplos de condiciones:

```
i < 10
keepGoing == true
residuo != 0
```

Ejemplo: Identifica cada componente del ciclo:

- Control variable
- Body
- Exit condition

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

El ciclo completo abarca desde la línea 5 a 9.

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

Variable de control: bits.

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

Cuerpo del ciclo.

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

Condición: repetir mientras bits sea menor o igual a 8.

```
01  public class SampleLoop {
02      public static void main(String[] args) {
03          int bits = 1;
04          int maxValues = 1;
05          while (bits <= 8) {
06              maxValues *= 2;
07              System.out.println(bits + " bit(s) -> " + maxValues + " v
08              bits++;
09          }
10      }
11  }
```

Condición de salida: bits mayor que 8

Prueba de Escritorio

	bits	maxValues	out	bits <= 8	
initial values -->	1	1		TRUE	<-- Enters while loop
	2	2	1 bit(s) -> 2 values	TRUE	
	3	4	2 bit(s) -> 4 values	TRUE	
	4	8	3 bit(s) -> 8 values	TRUE	
	5	16	4 bit(s) -> 16 values	TRUE	
	6	32	5 bit(s) -> 32 values	TRUE	
	7	64	6 bit(s) -> 64 values	TRUE	
	8	128	7 bit(s) -> 128 values	TRUE	
	9	256	8 bit(s) -> 256 values	TRUE	<-- Exit condition is met

Do-While

Do-While

El do-while es tipo de ciclo, muy similar al while, con la excepción de que el procesamiento se va a ejecutar **por lo menos una vez**.

La sintaxis es la siguiente:

```
01  do {
02    //
03    //code block to be executed
04    //
05  } while (boolean_condition);
```

```java
boolean invalidInput = true;
int i;
Scanner keyboard = new Scanner(System.in);
do {
    System.out.print("Type a number between 1 and 10: ");
    i = keyboard.nextInt();
    if (i >= 1 && i <= 10){
        invalidInput = false;
    } else {
        System.out.println("Error. Try again!");
    }
} while(invalidInput == true);

System.out.println("Success: " + i + " is a valid number");

keyboard.close();
```

```
01          boolean invalidInput = true;
02          int i;
03          Scanner keyboard = new Scanner(System.in);
04          do {
05              System.out.print("Type a number between 1 and 10: ");
06              i = keyboard.nextInt();
07              if (i >= 1 && i <= 10){
08                  invalidInput = false;
09              } else {
10                  System.out.println("Error. Try again!");
11              }
12          } while(invalidInput == true);
13
14          System.out.println("Success: " + i + " is a valid number");
15
16          keyboard.close();
```

El ciclo abarca de la línea 4 a la 12.

```
01          boolean invalidInput = true;
02          int i;
03          Scanner keyboard = new Scanner(System.in);
04          do {
05              System.out.print("Type a number between 1 and 10: ");
06              i = keyboard.nextInt();
07              if (i >= 1 && i <= 10){
08                  invalidInput = false;
09              } else {
10                  System.out.println("Error. Try again!");
11              }
12          } while(invalidInput == true);
13
14          System.out.println("Success: " + i + " is a valid number");
15
16          keyboard.close();
```

El cuerpo del ciclo de la 5-11. Estas instrucciones se repetirán.

```
01          boolean invalidInput = true;
02          int i;
03          Scanner keyboard = new Scanner(System.in);
04          do {
05              System.out.print("Type a number between 1 and 10: ");
06              i = keyboard.nextInt();
07              if (i >= 1 && i <= 10){
08                  invalidInput = false;
09              } else {
10                  System.out.println("Error. Try again!");
11              }
12          } while(invalidInput == true);
13
14          System.out.println("Success: " + i + " is a valid number");
15
16          keyboard.close();
```

Condicion: Repetir mientras invalidInput almacene un valor de "true".

```
01          boolean invalidInput = true;
02          int i;
03          Scanner keyboard = new Scanner(System.in);
04          do {
05              System.out.print("Type a number between 1 and 10: ");
06              i = keyboard.nextInt();
07              if (i >= 1 && i <= 10){
08                  invalidInput = false;
09              } else {
10                  System.out.println("Error. Try again!");
11              }
12          } while(invalidInput == true);
13
14          System.out.println("Success: " + i + " is a valid number");
15
16          keyboard.close();
```

Condición de salida: Detener la repetición cuando invalidInput almacene el valor boleano "false".

```
Type a number between 1 and 10: 20
Error. Try again!
Type a number between 1 and 10: -15
Error. Try again!
Type a number between 1 and 10: 33
Error. Try again!
Type a number between 1 and 10: 5
Success: 5 is a valid number
```

For

For

Los ciclos for generalmente son utilizados para repetir una serie de instrucciones una cantidad de veces fija.

La sintaxis es la siguiente:

```
01  for(initialization; boolean_condition; update){
02    //
03    // code block to be executed
04    //
05  }
```

```
01                 System.out.println(i);
02             }
03         }
```

```
01                 System.out.println(i);
02             }
03         }
```

Ciclo completo

```
01                System.out.println(i);
02            }
03        }
```

int i = 9 <-- Inicialización. Se declara una variable que sólo existe dentro del ciclo.

```
01                System.out.println(i);
02            }
03        }
```

i>=0 <-- Condición. Mientras esta condición se cumpla,el ciclo for continuará ejecutándose.

```
01                 System.out.println(i);
02             }
03         }
```

i = i-2 <-- Al final de cada ciclo, está operación se ejecutará, reduciendo el valor de "i" en "2".

Parrot Salute

En el zoológico de Monterrey hay un cotorro muy educado que saluda a cada grupo de personas que pasan. Lee del teclado el tamaño de un grupo, e imprime un saludo por cada persona.

Toma en cuenta lo siguiente:

- Los grupos no pueden ser de menos de 0 personas.
- Los grupos no pueden ser de más de 10 personas.

```java
        Scanner keyboard = new Scanner(System.in);

        System.out.print("How big is your group?: ");
        int groupSize = keyboard.nextInt();

        while (groupSize <= 0 || groupSize >= 11){
            System.out.print("Hmm that doesn't seem right. How big is your group: ");
            groupSize = keyboard.nextInt();
        }

        String salute = "";
        for(int i = 0; i < groupSize; i++){
            salute = salute + "Hi";
            if (i < (groupSize-1)){
                salute = salute + "-";
            }
        }

        System.out.println(salute);
```

Errores

Errores en Java

Un programa de Java puede tener tres tipos de errores:

1. Sintaxis
2. Ejecución (Runtime)
3. Lógica

Errores de Sintaxis

Son errores en la estructura del código fuente. Pueden ser palabras mal escritas, llaves no cerradas, paréntesis incompletos, etc. Generalmente estos errores son detectados por el compilador.

```
01  System.out.println(Hola mundo); //Text should be between "
02
03  int i //Missing ;
04
05  int j = 1.56; //int variables cannot hold decimal values
06
07  String s1;
08  System.out.println(s1); //String has not been initialized
```

Errores de Ejecución (Runtime errors)

Estos errores surgen durante la ejecución de un programa, en donde algun dato o operación generan algun problema.

```
01  int a = 1;
02  int b = 0;
03  int division = a / b; //Divide by zero exception!
04
05  int i = Scanner.nextInt(); //User types "a"
06
07  String s1 = "abcdef";
08  char c = s1.charAt(s1.length()); //String index out of range: 6
```

Errores de Lógica

Estos errores se originan por defectos inyectados por el programador. El programa no funciona de la manera esperada en alguno o todos los casos.

En los ciclos, hay errores como:

1. Off by one
2. Infinite loops
3. Wrong boolean expressions

Estos errores son más difíciles de encontrar, pues requiere que entendamos la intensión, el código y el programa que estamos evaluando.

```java
// off by one... prints only "hol"
String s1 = "hola";
int len = s1.length() - 1;
for (int i = 0; i < len; i++) {
  System.out.print(s1.charAt(i));
}
```

```java
//i is never updated, so this loops infinitely
while(i < 10){
  System.out.println(i);
}

// wrong condition i<0, causing an infinite loop
for (int i = 10; i>0; i++){
  System.out.println(i);
}
```