/* Ciclos y métodos */

Sesión Conceptual 01

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercitación

- Reflexión

Inicio

{desafío}
latam_

/* Ciclos */

Objetivo

- Conocer los ciclos y sus posibles aplicaciones en la programación
- Leer y transcribir diagramas de flujo con iteraciones a código Java
- Validar una entrada de datos
- Crear un menú de opciones

{desafío} latam_

Introducción a Ciclos

Nos permiten repetir la ejecución de una o más instrucciones

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

Uso de ciclos

- recorrer colecciones
- recorrer espacios de búsqueda
- su posibilidad de uso en algoritmos es infinito

Es importante aprender a utilizarlos y desarrollar habilidades lógicas para entender cuándo utilizarlos.

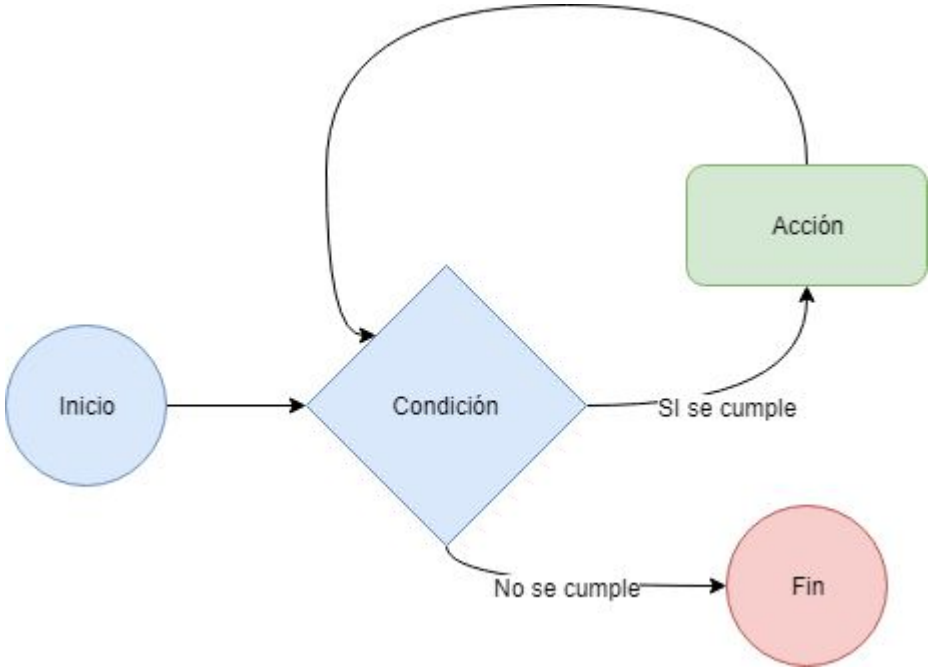
While

Nos permite ejecutar una o más operaciones **mientras** se cumpla una condición

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

`while` paso a paso

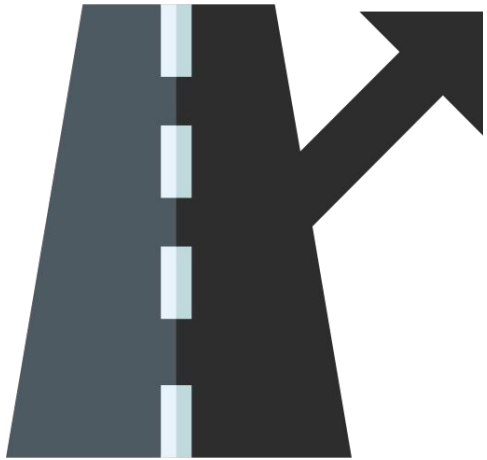
1. Se evalúa la condición
   a. si es true, ingresa al ciclo
2. Se ejecutan, secuencialmente las acciones
3. Se vuelve a evaluar la condición
4. si es true: se vuelve a repetir
   a. si es false: se termina el ciclo



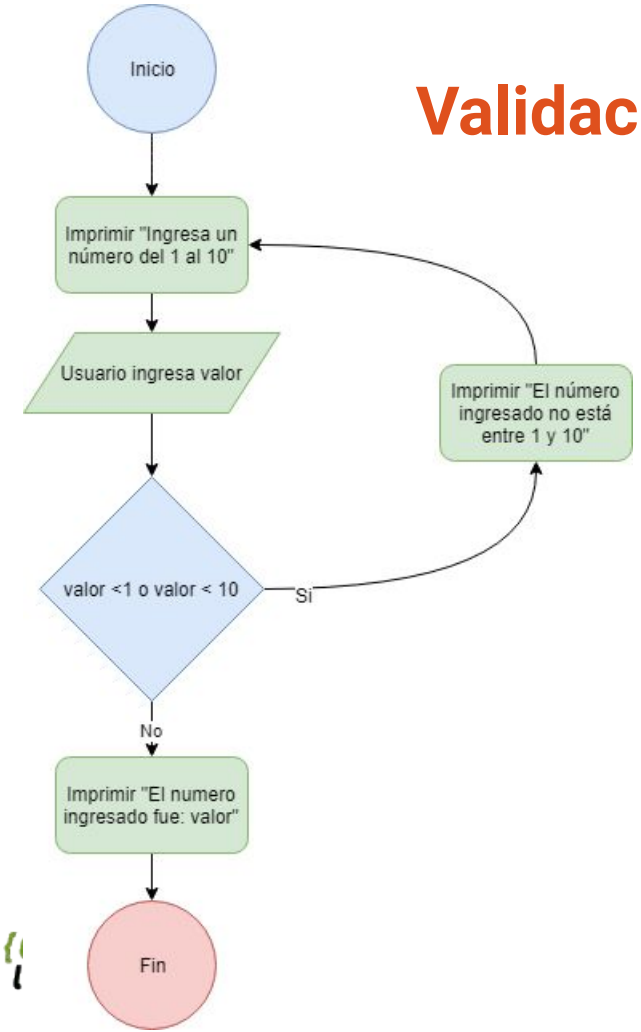
Salida del ciclo

Un algoritmo es una secuencia de pasos **FINITA**

En algún momento de alguna de las instrucciones dentro del ciclo deben lograr que la condición del `while` no se cumpla



Validación de entrada de datos usando `while`



Problema: ingresar un problema 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);
```

¿Por qué es necesario declarar antes del while el ingreso de datos por parte del usuario?

```
Scanner sc = new Scanner(System.in);
System.out.printf("Ingresa un número del 1 al 10: ");

while(num <1 || num > 10) {

      System.out.printf("El numero ingresado no es correcto\n");
      System.out.printf("Ingresa un número del 1 al 10: ");
      int num = sc.nextInt();
}
System.out.printf("El numero ingresado fue: %d \n",num);
```

La variable nun no estará definida, no ingresamos al ciclo y obtendremos un error.

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problems:
      num cannot be resolved to a variable
```

Ejercicio de integración

Problema: impedir al usuario entrar hasta que ingrese el password “password”

```
Scanner sc = new Scanner(System.in);
System.out.printf("Ingrese el password: ");
String password = sc.nextLine();

while(password.compareTo("password")!=0) {
      System.out.printf("El password es incorrecto\n", password);
      System.out.printf("Ingrese el password: ");
      password = sc.nextLine();
}
System.out.printf("El password ingresado es correcto\n");
```

do while

En este caso haremos una acción, hasta que se cumple la condición.

```
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);
```

Ejercicio: El menú

Desarrolla el diagrama de flujo de la solución y luego implementa el algoritmo

se requiere:

- Mostrar el 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.

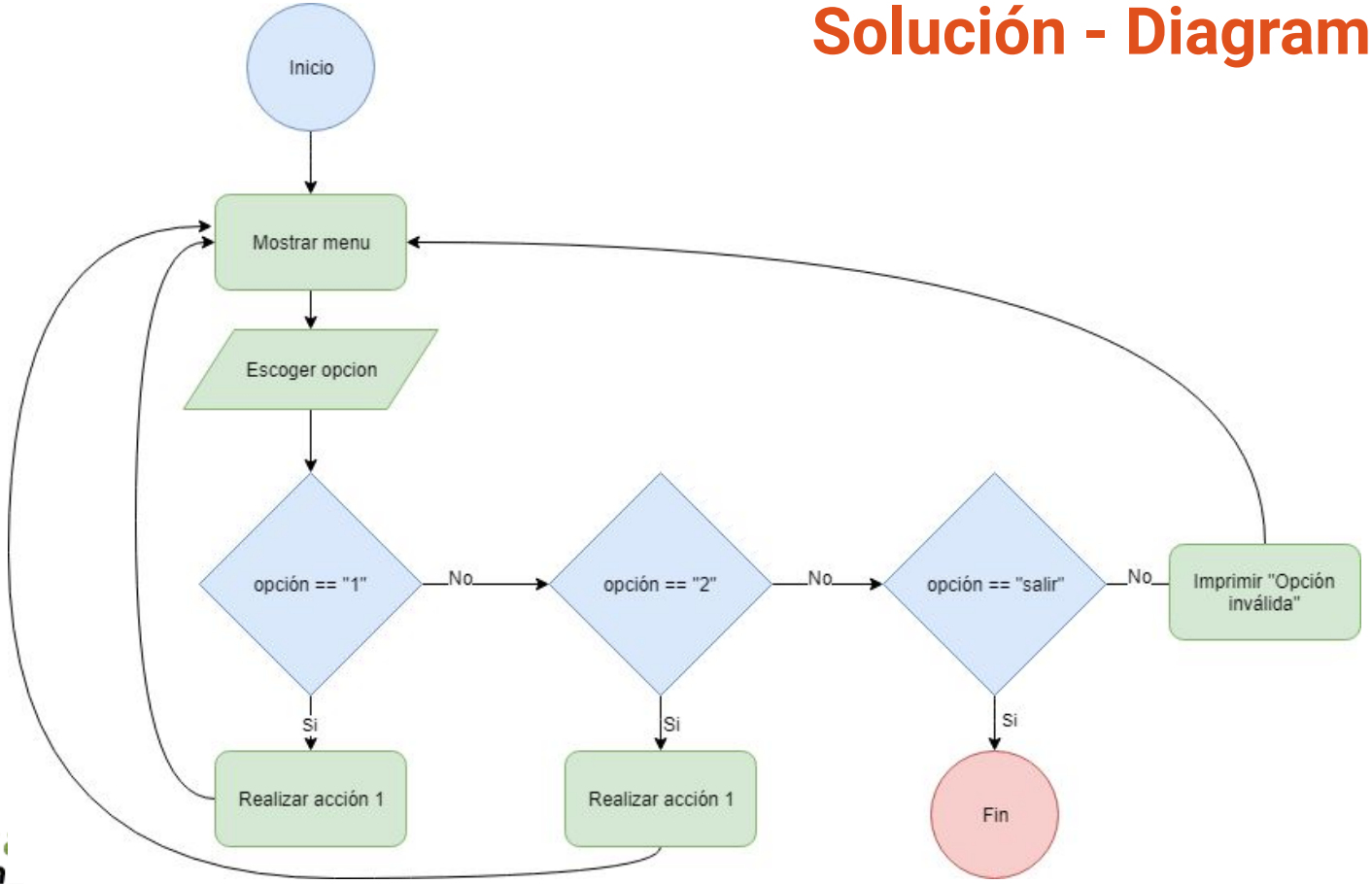
Ejercicio: El menú

Desarrolla el diagrama de flujo de la solución y luego implementa el algoritmo

se requiere:

- Mostrar el 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 - Diagrama de flujo



Solución - Código

```
Scanner sc = new Scanner(System.in);
String opcion  = "";

while(!opcion.equals("salir")) {   //0 son iguales 1.-1 mayor y menor
     System.out.printf("Escoge una opción\n");
     System.out.printf("1 -- Acción 1\n");
     System.out.printf("2 -- Acción 1\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("Opcion inválida\n");
}
```

Quiz

{desafío}
latam_

/* Ciclos y contadores */

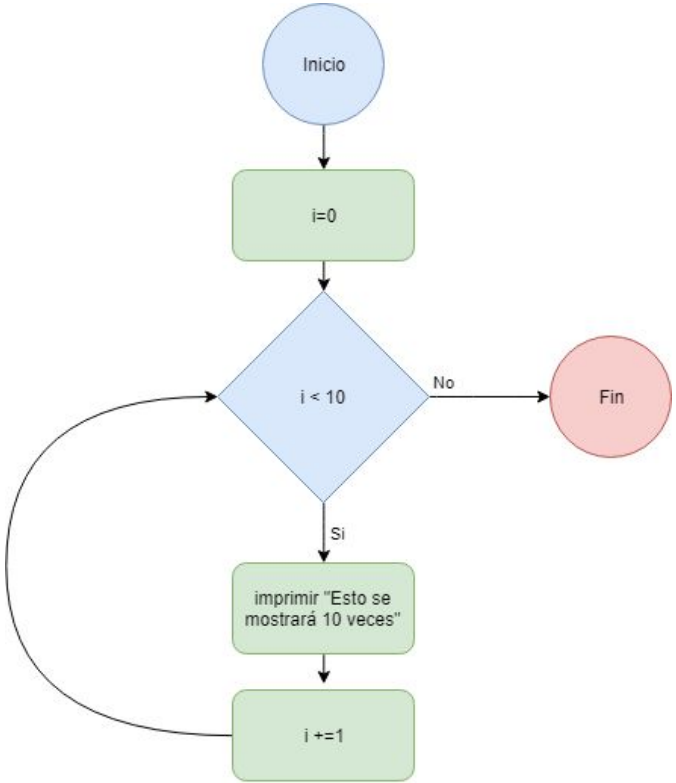
Objetivo

- Conocer el concepto de iteración.
- Contar la cantidad de veces que un programa está dentro de un ciclo.
- Realizar programas donde el usuario ingrese múltiples datos hasta que decida detenerse.
- Resolver problemas que requieren una cantidad determinada de ciclos o iteraciones.

Iterar

Iterar es dar **una vuelta al ciclo.**

Veamos el caso de contar desde cero hasta 10.



Contando con `while`

```
int i = 0;

while (i<10) {
      System.out.printf("Esto se mostrará 10 veces\n");
      i += 1 ;  //IMPORTANTE
}
```

La instrucción puts `System.out.printf("Esto se mostrará 10 veces\n");` se repetirá hasta que la variable i alcance el valor 10. Para entonces, la comparación de la instrucción while se evaluará como false y saldremos del ciclo.

IMPORTANTE: Si no aumentamos el valor de la variable `i` entonces nunca llegará a ser igual o mayor a 10.

¿Qué significa `i+=1;`

+= es un operador de asignación.

```
a = 2; // aquí se asigna el valor 2 a la variable a
a += 2; /*aquí el calor de la variable a aumento en dos, y fue almacenada nuevamente en a*/
a = a+2; //esto equivale a la sentencia anterior.
```

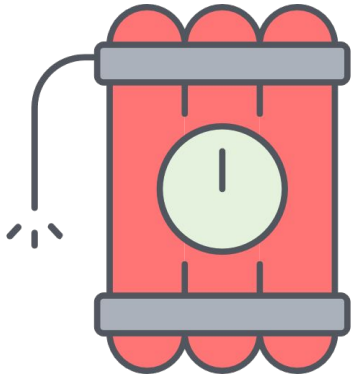
Operadores de asignación

Operador	Nombre	Ejemplo	Resultado
=	Asignación	a = 2	a toma el valor de 2
+=	Incremento y asignación	a += 2	a es incrementado en dos y asignado el valor resultante
-=	Decremento y asignación	a -=2	a es reducido en 2 y asignado el valor resultante
*=	Multiplicación y asignación	a *= 3	a es multiplicado por 3 y asignado el valor resultante
/=	División y asignación	a /=3	a es dividido por 3 y asignado el valor resultante

Ejercicio la bomba de tiempo

Crearemos un algoritmo sencillo que realice una cuenta regresiva de 5 segundos.

Contar de forma regresiva es muy similar, solo debemos comenzar desde el valor correspondiente e ir disminuyendo su valor de a uno en uno.



{desafío} latam_

Ejercicio la bomba de tiempo

```
int i = 5;
while (i > 0)  { //cuando lleguemos a cero terminamos.
      System.out.printf("%d\n",i);
      i -= 1 ;  //en cada iteración descontamos 1
      try {
            TimeUnit.SECONDS.sleep(1);
      } catch (InterruptedException e) {
            e.printStackTrace();
      }
}
```

Para que el programa espere de a 1 segundo, debemos importar `import java.util.concurrent.TimeUnit;`

y utilizar `TimeUnit.SECONDS.sleep(1);` donde el parámetro 1 indica que la espera será de un segundo.

Try Catch

En Java los errores en tiempo de ejecución se denominan excepciones.

Para evitar que el programa se interrumpa, Java maneja estas excepciones con la siguiente estructura

```
try {
	// Instrucciones cuando no hay una excepción
} catch (TypeException ex) {
	// Instrucciones cuando se produce una excepción
} finally {
	// Instrucciones que se ejecutan, tanto si hay como si no hay
excepciones
}
```

En un bloque de código se puede omitir el catch o finally, pero no ambos.

En el ejemplo bomba de tiempo

El método `TimeUnit.SECONDS.sleep(1);`

```
Exception in thread "main" java.lang.Error: Unresolved
compilation problem:
      Unhandled exception type InterruptedException
```

Si tratamos de ejecutar el método sin utilizar try catch, nos tirará el siguiente error
ya que este método requiere control de excepciones, por ello se escribe de la siguiente manera.

```
try {
      TimeUnit.SECONDS.sleep(1);
} catch (InterruptedException e) {
      e.printStackTrace();
}
```

Cuidado con las condiciones de borde

```
while (i >= 0){
      System.out.printf("%d\n",i);
      i -= 1 ;
      try {
            TimeUnit.SECONDS.sleep(1);
      } catch (InterruptedException e) {
            e.printStackTrace();
      }
}
```

```
while (i > 0)  {
      System.out.printf("%d\n",i);
      i -= 1 ;
      try {
            TimeUnit.SECONDS.sleep(1);
      } catch (InterruptedException e) {
            e.printStackTrace();
      }
}
```

Quiz

{desafío}
latam_

/* Ciclos y sumatorias */

{desafío}
latam_

Objetivo

- Conocer / repasar las operaciones de sumatoria
- Crear diagramas de flujo de problemas de sumatorias
- Escribir en Java el código de una sumatoria
- Conocer la diferencia entre un contador y un acumulador.

{desafío} latam_

Introducción a sumatorias

Para resolver una sumatoria, solo necesitamos saber una cosa: **sumar.**

La sumatoria consiste en sumar todos los números de una secuencia.

Por ejemplo:

sumar todos los números entre 1 y 100.

{desafío} latam_

Sumando de 1 a 100

- Resolver esto es muy similar a contar las cien veces,
- En **cada iteración** iremos guardando la suma

1 + 2 + 3 + … + 99 + 100 = ?

Comencemos de la base del ejercicio anterior

```
int i = 0;
while (i < 100){
      i+=1;
}
```

Aca iteramos 100 veces

```
int i = 0;
int suma = 0;
while (i < 100){
      i+=1;
}
```

Creamos una variable para almacenar la suma

Donde realizamos la suma?

```
int i = 0;
int suma = 0;
while (i < 100){
      i+=1;
      suma += i;
}
```

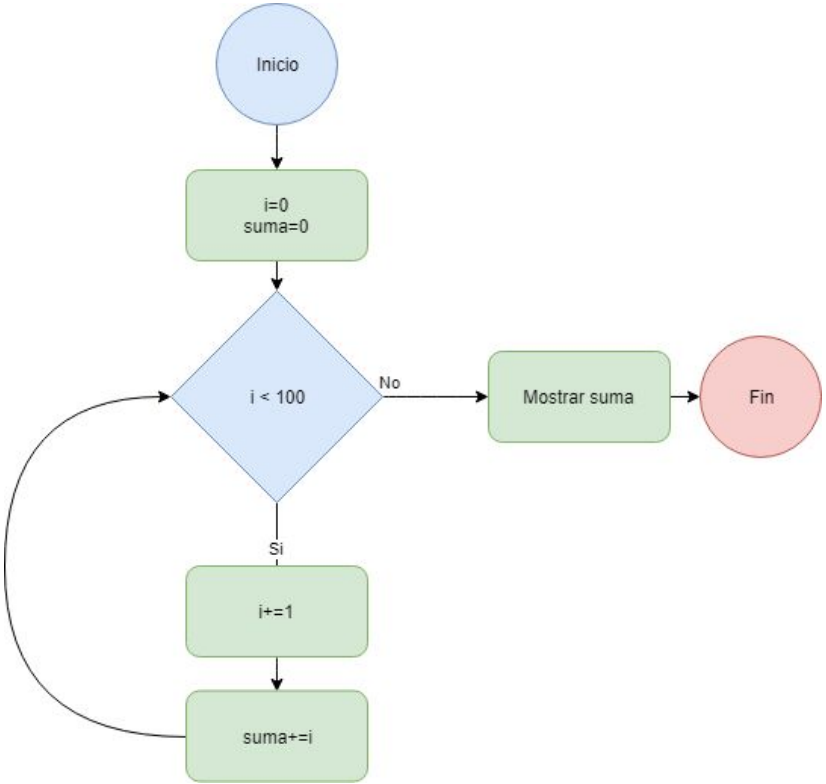
Si queremos sumar del 1 al 100, debemos partir cuando i valga 1 la primera vez

```
int i = 0;
int suma = 0;
while (i < 100){
      suma += i;
      i+=1;
}
```

Aca i vale 0 la primera vez, y contaremos de 0 a 99

{desafío} latam_

Diagrama de flujo al problema sumando de 1 a 100



{desafío}
latam_

Desafíos

- Crear el programa Suma.java, donde el usuario ingresa un número, se muestra la suma de todos los números de 1 hasta ese número.

  si n = 100, el resultado es 5050

- Crear el programa SumaPar.java donde se sumen únicamente los números pares dentro del ciclo entre 1 y un número ingresado por el usuario.

  si n = 100, el resultado debe ser 2550

Solución a desafíos - suma de 1 a n

```java
package ciclos;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;;
public class Ciclos {

	public static void main(String[] args)   {
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int i = 0;
		int suma = 0;
		while (i < n){
			i+=1;
			suma += i;
		}
		System.out.printf("%d\n",suma);
	}
}
```

{desafío} latam_

Suma de los pares de 1 a n

```
package ciclos;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;;
public class Ciclos {

	public static void main(String[] args)   {
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int i = 0;
		int suma = 0;
		while (i < n){
			i+=1;

			if(i%2 == 0) { // es par
				suma += i;
			}
		}
		System.out.printf("%d\n",suma);
	}
}
```

{desafío} latam_

Resumen del capítulo

- Contador: Aumenta de 1 en 1
  - cont = cont + 1
  - cont += 1
- Acumulador: Aumenta en función a valor
  - acu = acu + valor
  - acu += valor

Generador de listas en HTML

Crear un programa donde el usuario ingrese un número, y se genere una lista de HTML con esa cantidad de ítems

```
<ul>
    <li> 1 </li>
    <li> 2 </li>
    <li> 3 </li>
</ul>
```

{desafío} latam_

Solución al generador de listas HTML

Preocupémonos primero de la parte que se repite.

```
package listashtml;
import java.util.Scanner;
public class ListasHTML {

	public static void main(String[] args)   {
		String html = "";
		Scanner sc = new Scanner(System.in);
		int items = sc.nextInt();
		int i = 0;
		while (i<items) {
			i+=1;
			html += "<li> item " + i + "</li>\n";
		}
		System.out.printf(html);
	}
}
```

{desafío} latam_

Solución al generador de listas HTML

Si n=3, obtendremos lo siguiente

```
<li> item 1</li>
<li> item 2</li>
<li> item 3</li>
<li> item 4</li>
<li> item 5</li>
```

```
package listashtml;
import java.util.Scanner;
public class ListasHTML {

	public static void main(String[] args)	{
		String html = "<ul>\n";
		Scanner sc = new Scanner(System.in);
		int items = sc.nextInt();
		int i = 0;


		while (i<items) {
			i+=1;
			html += "<li> item " + i + "</li>\n";
		}
		html += "</ul>\n";
		System.out.printf(html);
	}
}
```

{desafío} latam_

Solución al generador de listas HTML

Ahora debemos agregar el inicio y el final.

```
package listashtml;
import java.util.Scanner;
public class ListasHTML {

	public static void main(String[] args)   {
		String html = "<ul>\n";
		Scanner sc = new Scanner(System.in);
		int items = sc.nextInt();
		int i = 0;
		while (i<items) {
			i+=1;
			html += "<li> item " + i + "</li>\n";
		}
		html += "</ul>\n";
		System.out.printf(html);
	}
}
```

{desafío} latam_

Solución al generador de listas HTML

Ahora agregamos las tabulaciones

```
package listashtml;
import java.util.Scanner;
public class ListasHTML {

	public static void main(String[] args)   {
		String html = "<ul>\n";
		Scanner sc = new Scanner(System.in);
		int items = sc.nextInt();
		int i = 0;
		while (i<items) {
			i+=1;
			html += "\t<li> item " + i + "</li>\n";
		}
		html += "</ul>\n";
		System.out.printf(html);
	}
}
```

{desafío} latam_

Ejercicios sugeridos

Ejercicios

- Promedio.java: El usuario ingresa un número, se muestra la suma de todos los números de 1 hasta ese número dividido por la cantidad de iteraciones.

- PromedioUsario.java: El usuario ingresa un número que indica la cantidad de datos que va ingresar, luego ingresa esa cantidad de datos, el programa al final entrega el promedio.

- SumaPares.java: El usuario ingresa 2 valores, el inicio(n) y término(m). El programa solo suma los números pares entre n a m.

Ciclos con la instrucción `for`

Nos permite iterar en un rango definido, por ejemplo de 1 a 10.

```
int i;
for(i=1;i<=10;i++) {
      System.out.printf("iteración %d\n", i);
}
```

Es convención utilizar i como variable de iteración.

Su sintaxis está dada por:

```
for( variable de iteración ; condición ; incremento){
// bloque de instrucciones
}
```

Incremento

En la sentencia anterior utilizamos i++, lo que incrementa el valor de i en 1.

Equivale a i+=1;

El incremento puede ser de a más de 1.

```
int i;
int suma = 0;
for( i = 2; i<=100;i+=2){
      suma += i;
}
System.out.printf("%d\n",suma);
```

Ventaja `while` vs `for`

While => bucle de iteración: no se sabe exactamente cuántas veces se hará la iteración.

For => bucle de control: se repite el bucle un número determinado de veces

Quiz

{desafío}
latam_

/* Sumatorias y productorias */

Objetivo

- Leer expresiones de sumatorias y productorias
- Escribir código para resolver sumatorias

{desafío} latam_

Introducción

- Frecuente para un programador tener que implementar fórmulas matemáticas.
- Cálculo de negocio, proyecciones de venta, realizar una simulación, entre otros.

$$\sum_{i=5}^{100} i = 5+6+7+...+100$$

i=5100i = 5+6+7+...+100

número inferior: primer valor de la iteración,

número superior, último de la iteración.

En código sería

$$\sum_{i=5}^{100} i = 5+6+7+...+100$$

```
int suma = 0;
int i;
for(i=5;i<=100;i++){
      suma+=i;
}
System.out.printf(suma);
```

En este otro ejemplo

$$\sum_{i=3} 2i = 2*3+2*4+...+2*9$$

la implementación es directa

```
int suma = 0;
int i;
for(i=3;i<=9;i++){
      suma+=2*i;
}
System.out.printf(suma);
```

Productorias

$$\prod_{i=1}^{10} i = 1 * 2 * 3 * \ldots * 10$$

```
producto = 1; //es importante no inicializar el producto en 0
int i;
for(i =1;i<=10;i++){
      producto *=i;
}
System.out.printf(producto);
```

Quiz

{desafío}
latam_

Cierre

{desafío}
latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales