



# ORACLE

## Academy

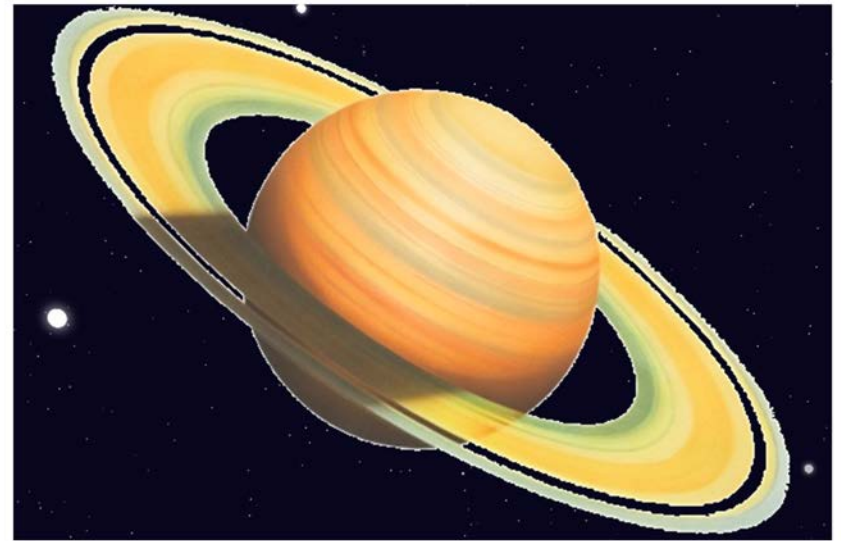


# Java Foundations

6-1

Bucles for

**ORACLE**  
Academy



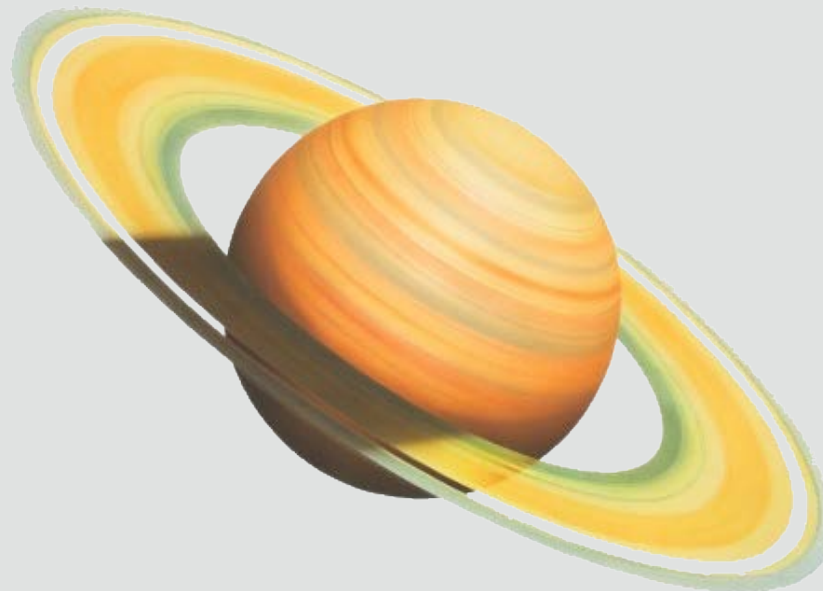
# Objetivos:

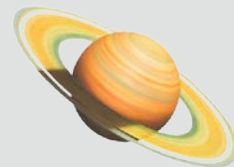
- En esta lección se abordan los siguientes objetivos:
  - Entender los componentes del bucle for estándar
  - Entender la creación y el uso de un bucle for
  - Describir el ámbito de la variable
  - Describir las técnicas de depuración
  - Explicar cómo se producen los bucles infinitos en Java



# Misión a los anillos de Saturno

- Vamos a lanzar un cohete
- Su misión es estudiar los anillos de Saturno
- ¿Tiene alguna idea acerca de cómo programar un temporizador de cuenta atrás?





# La cuenta atrás

- Contar hacia atrás desde 10 necesita 10 líneas de código

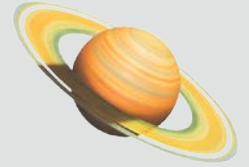
```
System.out.println("Countdown to Launch: ");  
System.out.println(10);  
System.out.println(9);  
System.out.println(8);  
System.out.println(7);  
System.out.println(6);  
System.out.println(5);  
System.out.println(4);  
System.out.println(3);  
System.out.println(2);  
System.out.println(1);  
System.out.println("Blast Off!");
```



# La cuenta atrás

- Contar hacia atrás desde 100 necesitaría 100 líneas de código
- Esto sería difícil y tedioso de programar
- ¿Hay alguna forma más práctica de escribir este programa?
- ¿Puede el código adaptarse fácilmente a cualquier valor de inicio?

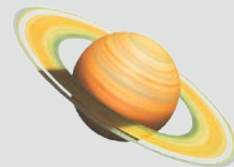




# La cuenta atrás

```
System.out.println("Countdown to Launch: ");  
System.out.println(100);  
System.out.println(99);  
System.out.println(98);  
System.out.println(97);  
System.out.println(96);  
System.out.println(95);  
...  
...  
...  
...  
...  
...  
System.out.println(2);  
System.out.println(1);  
System.out.println("Blast Off!");
```





# ¿Pueden las variables ayudar?

- Las variables son relativamente útiles
- Sin embargo, todavía tenemos que copiar y pegar las mismas líneas de código hasta 0 impresiones

```
System.out.println("Countdown to Launch: ");
```

```
int i = 10;
```

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

```
i--;
```

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

```
i--;
```

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

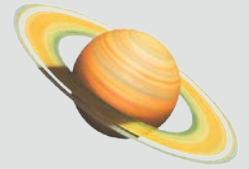
```
i--;
```

```
...
```

```
System.out.println("Blast Off!");
```








# Código de repetición

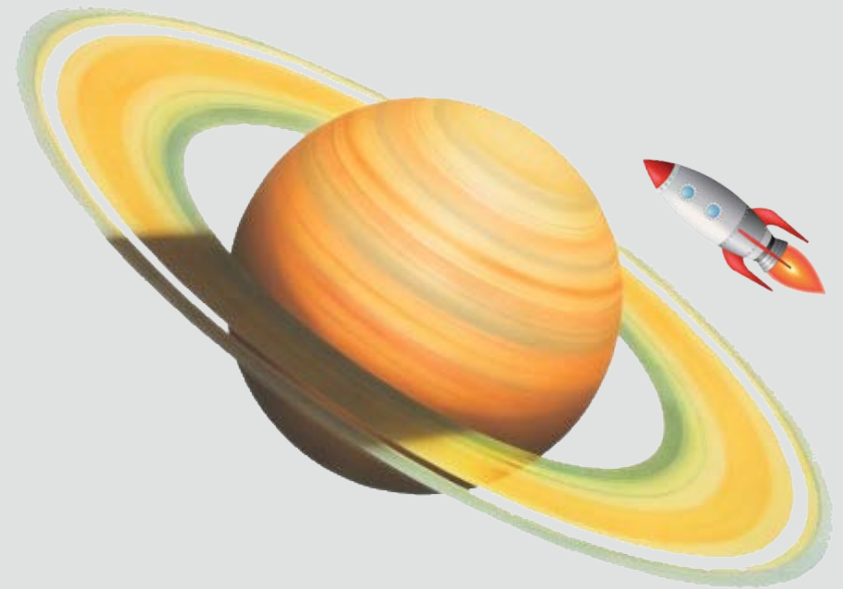
- ¿Podemos hacer que las mismas líneas de código se repitan un número variable de veces?
- Las líneas 7-10 muestran el bloque de código que deseamos repetir
- Recuerde la naturaleza línea a línea de los programas:
  - Cuando el programa alcanza la línea 10...
  - Deseamos volver a la línea 7

```
5  int i = 10;
6
7  {
8      System.out.println(i);
9      i--;
10 }
```



# Sentencias de bucle

- Las sentencias de bucle se utilizan para repetir líneas de código
- Java proporciona tres tipos de bucles:
  - for
  - while
  - do-while



# Comportamiento de repetición



```
while (!areWeThereYet) {  
  
    read book;  
    argue with sibling;  
    ask, "Are we there yet?";  
  
}  
  
Woohoo!;  
Get out of car;
```



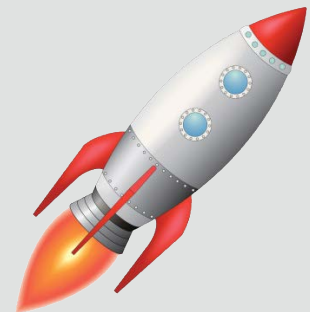
# Bucles

- Los bucles se usan en los programas para la ejecución repetida de una o más sentencias hasta que se alcanza la condición de terminación
  - Hasta que una expresión es false
    - o
  - Un número específico de veces:
    - Deseo imprimir los números del 1 al 10
    - Deseo calcular la suma de los números en un rango determinado
- Un bucle for se ejecuta un número determinado de veces
  - Los bucles for también se denominan bucles definidos

# Lo que sabemos

- En el escenario de la cuenta atrás, esto es lo que sabemos:

Lo que sabemos	Nombre técnico	Código
Cuando se inicia el bucle...	Expresión de inicialización	<code>int i = 10;</code>
Continúe el bucle si...	Expresión de condición	<code>i &gt;= 0;</code>
Después de cada bucle...	Expresión de actualización	<code>i--;</code>
Código para repetir	Sentencias de código	<code>System.out.println(i);</code>



# Visión general del bucle for

- Sintaxis: **Encabezado**

```
for(initialization; condition; update){  
    Code statement(s)  
    Code statement(s) } Body  
} //end for
```

- La expresión de inicialización inicializa el bucle Lo ejecuta una sola vez, conforme empieza el bucle
- Cuando la expresión de condición se evalúa como false, el bucle termina
- Se llama a la expresión de actualización después de cada iteración a través del bucle Esta expresión puede aumentar o disminuir un valor
- Cada expresión se debe separar por un punto y coma (;)



# Expresión de inicialización

- Se realiza una vez conforme empieza el bucle
- Indica al compilador la variable (denominada contador de bucle) que se utiliza en el bucle
- Puede comenzar en cualquier valor, no solo en 10

```
System.out.println("Countdown to Launch: ");
```

```
for(int i = 10; i >= 0; i--) {  
    System.out.println(i);  
}//end for
```

```
System.out.println("Blast Off!");
```

# Expresión de condición

- El bucle continúa siempre que esta expresión sea true
- Utiliza operadores de comparación:
  - (==, !=, <, >, <=, >=)

```
System.out.println("Countdown to Launch: ");
```

```
for(int i = 10; i >= 0; i--) {  
    System.out.println(i);  
} //end for
```

```
System.out.println("Blast Off!");
```

# Expresión de actualización

- Esta sentencia se ejecuta después de cada iteración del bucle for
- Se utiliza para actualizar el contador de bucles

```
System.out.println("Countdown to Launch: ");  
  
for(int i = 10; i >= 0; i--) {  
    System.out.println(i);  
}//end for  
  
System.out.println("Blast Off!");
```

# Ejercicio 1, parte 1

- Cree un nuevo proyecto y agréguele el archivo `Countdown.java`
- Defina un punto de ruptura en `Countdown.java` y observe...
  - Cómo el bucle `for` afecta a la ejecución de código
  - Cómo el valor de `i` cambia

Name	Type	Value
Static		
args	String[]	#71(length=0)
i	int	5

¿Cuál es el valor de `i` cuando el bucle `for` termina?

## Ejercicio 1, parte 2

- ¿Se puede modificar el código para empezar a contar desde 0 hasta 5?
- ¿Se puede modificar el código para contar todos los números pares de 0 a 20?

# ¿Necesito la expresión de actualización?

- ¿Qué pasaría si escribiera mi bucle así?

```
for(int i = 10; i >= 0; ) {  
    System.out.println(i);  
    i--;  
}//end for
```

- Eso también funciona
- Pero puede que no desee el código de esta forma, ya que los bucles se pueden volver más complicados



# Omisión de las expresiones en el bucle for

- Todas las expresiones en la cabecera son opcionales
- Pero existen riesgos cuando se omite una expresión:
  - Sin inicialización:
    - La inicialización no se realiza
    - Puede que no haya ningún contador de bucles
  - Sin condición:
    - La condición de bucle siempre se considera que sea true
    - El bucle es un bucle infinito
  - Sin actualización:
    - No se realiza ningún incremento en la operación
    - El contador de bucles mantiene el mismo valor

# Omisión de todas las expresiones en el bucle for

- Examine el siguiente código:
  - Se pueden omitir las tres expresiones en el bucle for
  - El bucle se repite infinitamente

```
for(;;){  
    System.out.println("Welcome to Java");  
}//end for
```

## Ejercicio 2

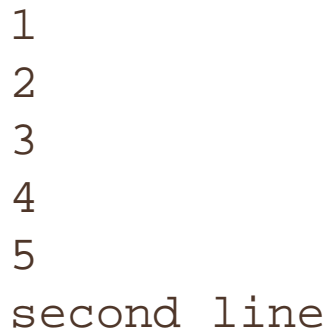
- Agregue el archivo `InfiniteLoop.java` al proyecto creado para el ejercicio 1
- Ejecute `InfiniteLoop.java` y observe la salida
- Modifique el bucle `for` en `InfiniteLoop.java` para imprimir "**Hello**" cinco veces

# Sentencias múltiples en un cuerpo del bucle

- Para ejecutar varias sentencias en un cuerpo...
- Incluya las sentencias dentro de un par de corchetes angulares
- De lo contrario, solo la primera sentencia del cuerpo se ejecuta

```
for(int i = 1; i <= 5; i++)  
System.out.println(i);  
System.out.println("second line");
```

- Resultado:



```
1  
2  
3  
4  
5  
second line
```

# Un uso del bucle for

- El bucle for proporciona un medio compacto para iterar sobre un rango de valores
- Repetición sin el bucle for:

```
//Prints the square of 1 through 5
System.out.println("1 squared = " + 1 * 1);
System.out.println("2 squared = " + 2 * 2);
System.out.println("3 squared = " + 3 * 3);
System.out.println("4 squared = " + 4 * 4);
System.out.println("5 squared = " + 5 * 5);
```

- Repetición con el bucle for:

```
for(int i = 1; i <= 5; i++){
    System.out.println("i squared = " + i * i);
} //end for
```

# i es el contador de bucles

- Todos los ejemplos que hemos visto se basan en el contador de bucles

```
for(int i = 1; i <= 5; i++){  
    System.out.println("i squared = " + i * i);  
} //end for
```

- i puede:
  - Imprimirse
  - Cambiar sus valores
  - Utilizarse en los cálculos
- Esta opción es ideal para:
  - Recuento
  - Calcular valores de forma rápida



# Descripción del ámbito de las variables

- Pero `i` solo existe en el bucle `for`
  - Esto se conoce como el ámbito de `i`
  - `i` ya no existe cuando el bucle `for` termina
  - Si `i` se utiliza para calcular los valores, nunca obtendremos esos valores fuera del bucle `for`
- ¿No se ha dado cuenta de que `i` desaparece al depurar `Countdown.java`?

```
for(int i = 1; i <= 5; i++){  
    System.out.println("i squared = " + i * i);  
}//end for
```

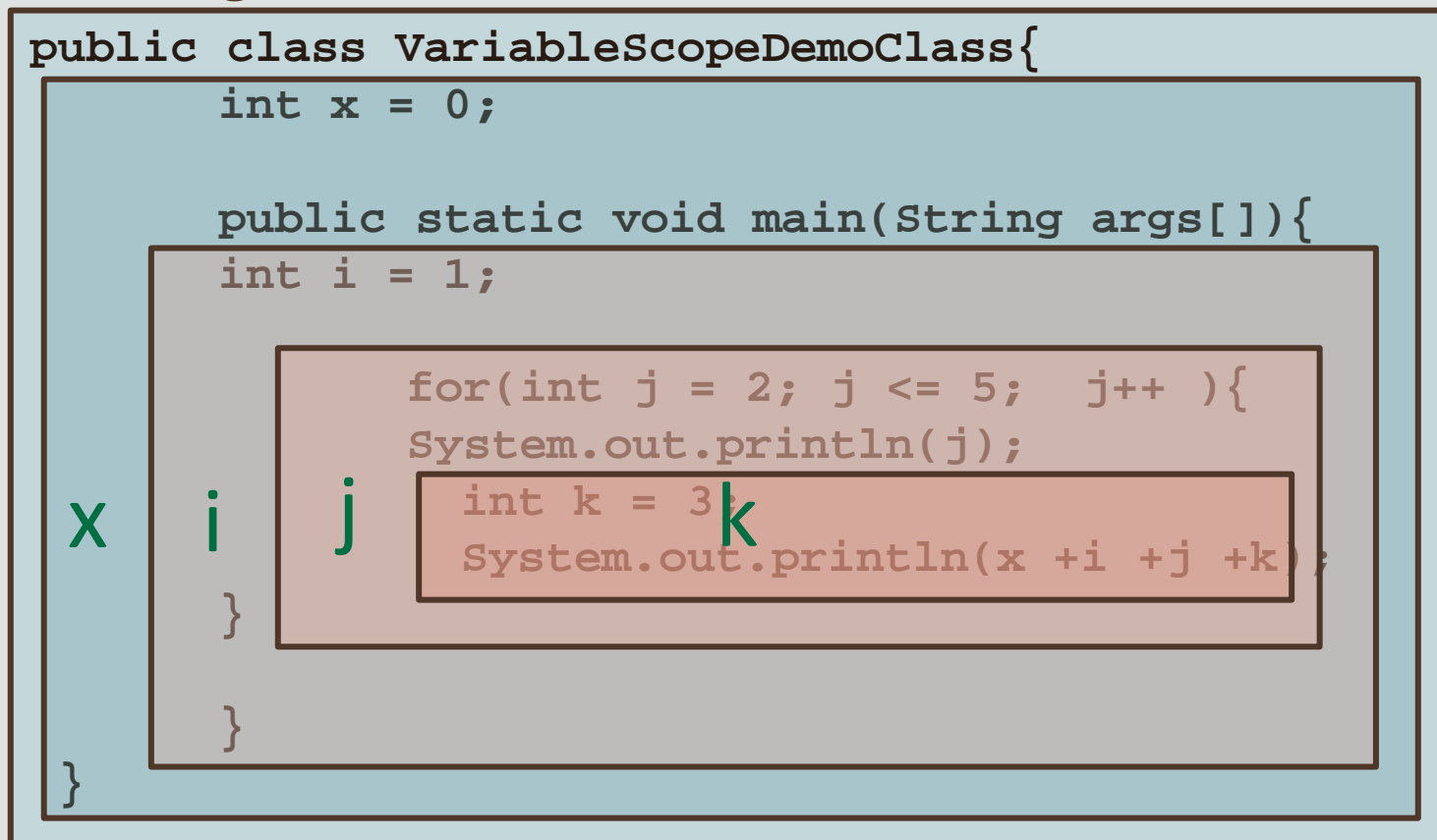
# Ámbito de las variables: Ejemplo

- La variable `i` declarada en el bucle `for` es una variable local y no se puede acceder a ella fuera del bucle
- El error del compilador se genera en la línea 8

```
1 public class VariableScopeDemo {
2
3     public static void main(String args[]){
4
5         for(int i = 0; i <= 5; i++ ){
6             System.out.println("i: " +i);
7         }//end for
8     → System.out.println("i: " +i);
9     }//end method main
10 }//end class VariableScopeDemo
```

# Animación del ámbito

- Las variables no pueden existir antes o fuera de su bloque de código



## Otro uso de los bucles

- Supongamos que necesita encontrar la suma de cuatro números

```
import java.util.Scanner;
public class Add4Integers {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("This program adds four numbers.");
        System.out.println("Type each number, followed by Enter.");
        int n1 = in.nextInt();
        int n2 = in.nextInt();
        int n3 = in.nextInt();
        int n4 = in.nextInt();
        int total = n1 + n2 + n3 + n4;
        System.out.println("The total is " + total + ".");
    } //end method main
} //end class Add4Integers
```

## Otro uso de los bucles

- Este enfoque es complejo de programar si desea agregar 100 valores

```
int n1 = in.nextInt();  
int n2 = in.nextInt();  
int n3 = in.nextInt();  
int n4 = in.nextInt();  
  
...  
int n100 = in.nextInt();  
int total = n1 + n2 + n3 + n4 +... + n100;
```

- ¿Puede un bucle for hacer que este programa sea más corto?
- ¿Puede un bucle for ayudar a averiguar la suma de un número variable de enteros?

# Uso de opciones de ámbito con los bucles for

- Esto se puede resolver con...
  - Un bucle for con variables de diferentes ámbitos

```
import java.util.Scanner;
public class PracticeCode {
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int N = 100;
    int total = 0;
    System.out.println("This program adds " + N + " numbers.");
    for(int i = 0; i < N; i++){
        System.out.println("Enter your next number:");
        int value = in.nextInt();
        total += value;
    } //end for
    System.out.println("The total is " + total + ".");
} //end method main
```



# Animación del ámbito

```
import java.util.Scanner;
public class PracticeCode {
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int N = 100;
    int total = 0;
    System.out.println("This program adds " + N + " numbers.");
    for(int i = 0; i < N; i++){
        System.out.println("Enter your next number:");
        int value = in.nextInt();
        total += value;
    } //end for
    System.out.println("The total is " + total + ".");
} //end method main
```

N

total

i

value

## Ejercicio 3

- Agregue el archivo `ScopeTest.java` al proyecto creado para el ejercicio 1
- `ScopeTest.java` se ha interrumpido
- ¿Puede solucionarlo?
- Debe aparecer la siguiente salida:  
-64 32 16 8 4 2 1  
-0 1 2 3 4 5  
-5 4 3 2 1 0  
-2 4 8 16 32 64

# Variable ya definida

- i se crea antes que el bucle for
- Otra i no puede existir en el mismo ámbito
- Una de estas variables necesita un nombre diferente

```
public static void main(String[] args) {
```

```
    int i = 0;
```

i

```
    for(int i = 64; i > 0; i=i/2 ){  
        System.out.print(i + " ");  
    }
```

```
}
```

# Fuera del ámbito

- j no puede existir fuera del ámbito en el que se ha creado
- Se puede crear otra j si los ámbitos no se superponen

```
public static void main(String[] args) {  
    for(int j = 0; j<=5; j++){  
        j    System.out.print(j + " ");  
    }  
  
    for(int j = 5; j>=0; j--){  
        j    System.out.print(j + " ");  
    }  
  
    for(int k = 2; k<=64; k=k*2){  
        k    System.out.print(j + " ");  
    }  
}
```

# ¿Necesito la expresión de inicialización?

- ¿Qué pasaría si escribiera mi bucle así?

```
int i = 10;  
for(; i >= 0; i--){  
    System.out.println(i);  
}//end for
```

- Eso también funciona
  - Pero i existe fuera del ámbito del bucle for
  - Si i solo está destinado a ser un contador de bucles, la variable está desperdiciando memoria
  - Mantenga el ámbito reducido (tan pequeño como sea posible)
  - Las variables perdidas complican el código y aumentan el potencial de errores

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Entender los componentes del bucle for estándar
  - Entender la creación y el uso de un bucle for
  - Describir el ámbito de la variable
  - Describir las técnicas de depuración
  - Explicar cómo se producen los bucles infinitos en Java





# ORACLE

## Academy

