Chapter 4 Loop

Increment and decrement

shortcuts to increase or decrease a variable's value by 1

Shorthand	Equivalent longer version
variable++;	**variable** = **variable** + 1;
variable--;	**variable** = **variable** - 1;

```
int x = 2;
x++;                          // x = x + 1;
                              // x now stores 3

double gpa = 2.5;
gpa--;                        // gpa = gpa - 1;
                              // gpa now stores 1.5
```

Modify-and-assign

shortcuts to modify a variable's value

Shorthand	Equivalent longer version
variable += **value;**	**variable** = **variable** + **value;**
variable -= **value;**	**variable** = **variable** - **value;**
variable *= **value;**	**variable** = **variable** * **value;**
variable /= **value;**	**variable** = **variable** / **value;**
variable %= **value;**	**variable** = **variable** % **value;**

```
x += 3;                  // x = x + 3;

gpa -= 0.5;              // gpa = gpa - 0.5;

number *= 2;             // number = number * 2;
```

Loops

- A ***loop*** is a program construct that repeatedly executes the loop's statements (known as the ***loop body***) while the loop's expression is true; when false, execution proceeds past the loop.

- Each time through a loop's statements is called an ***iteration***.

`while` loops

Motivations

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a hundred times:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?

Opening Problem

Problem:

100 times

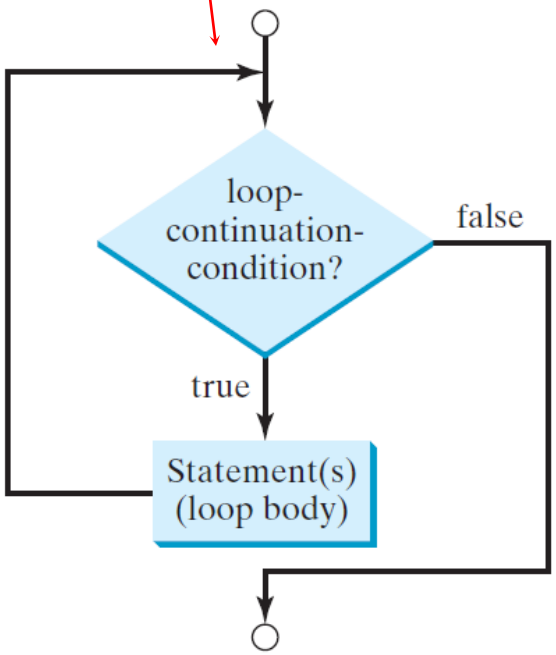
```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
                                    ...
                                    ...
                                    ...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

Introducing while Loops

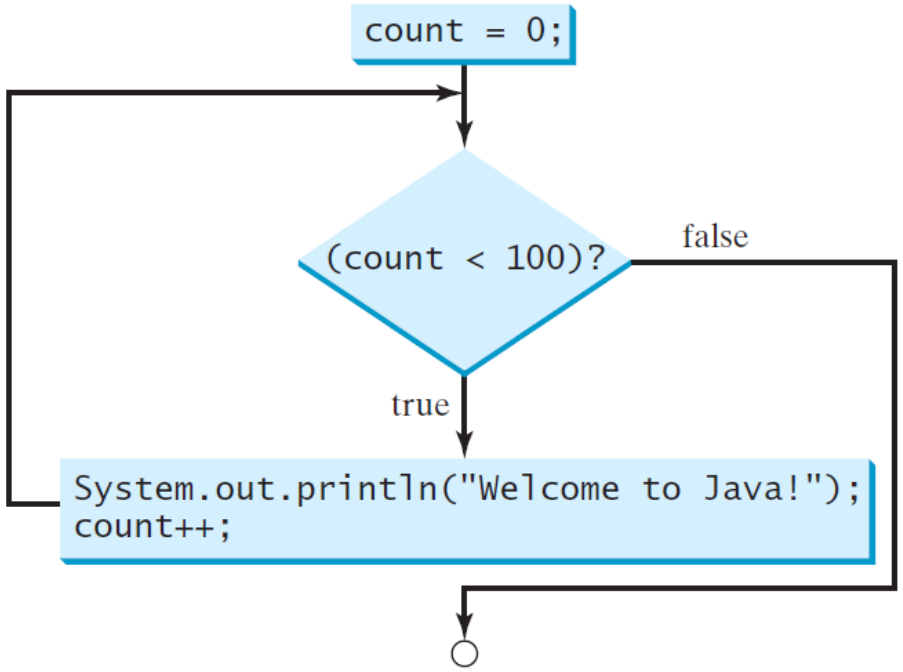
```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java");
  count++;
}
```

`while` Loop Flow Chart

```
while (loop-continuation-condition) {
  // loop-body;
  Statement(s);
}
```




```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```



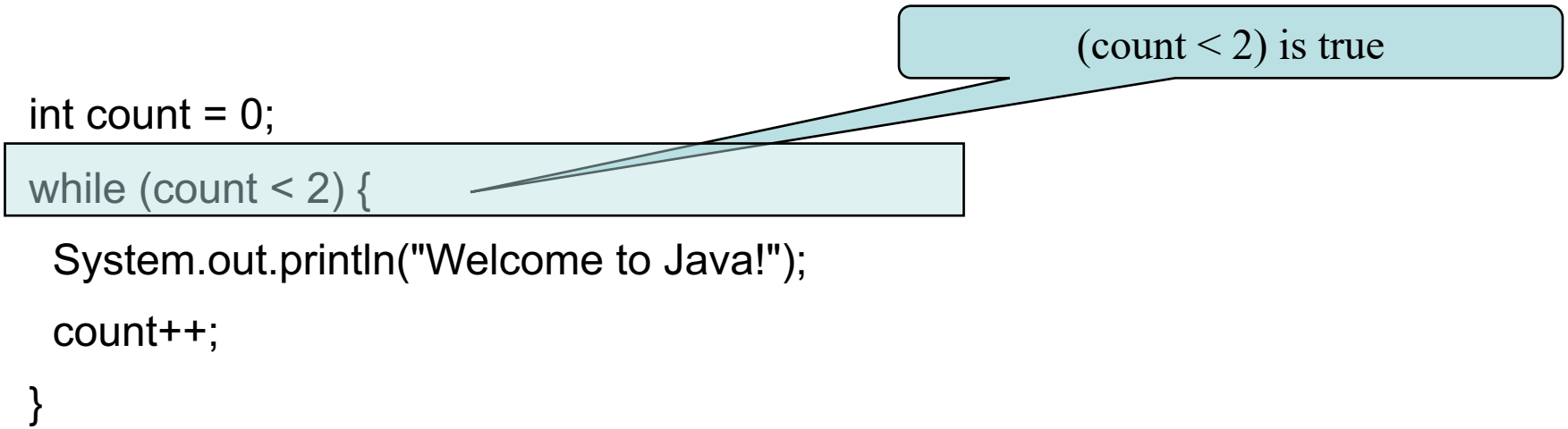
Trace while Loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

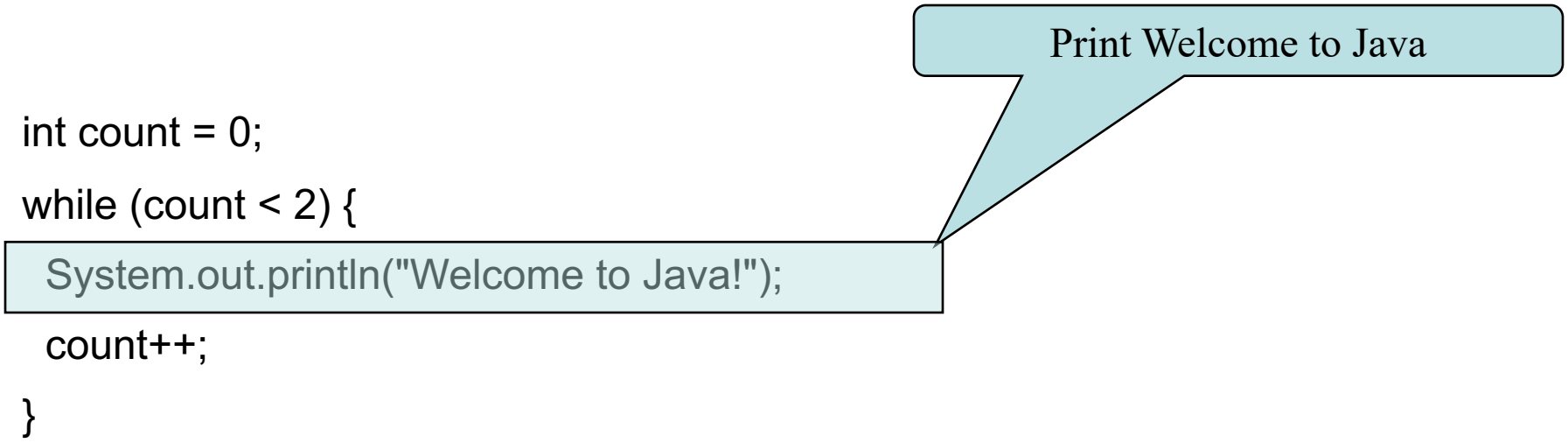
Initialize count

animation

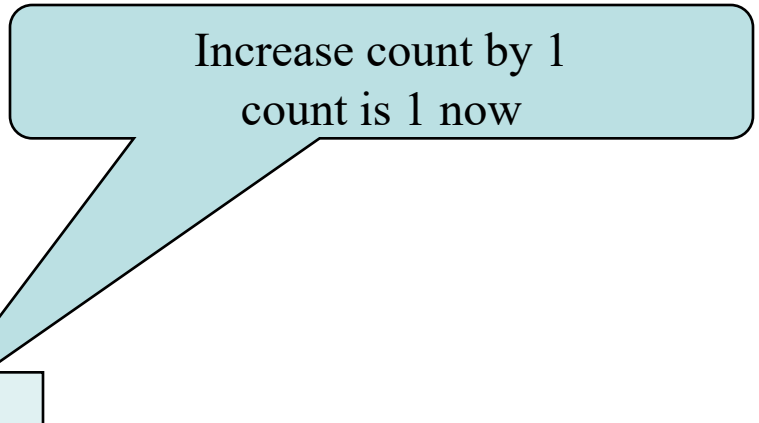
Trace while Loop, cont.



Trace while Loop, cont.



Trace while Loop, cont.




```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

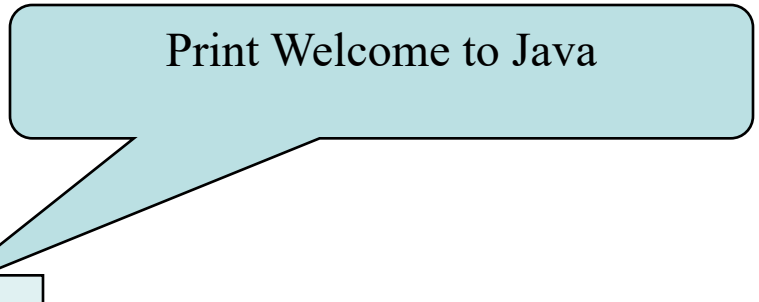
animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

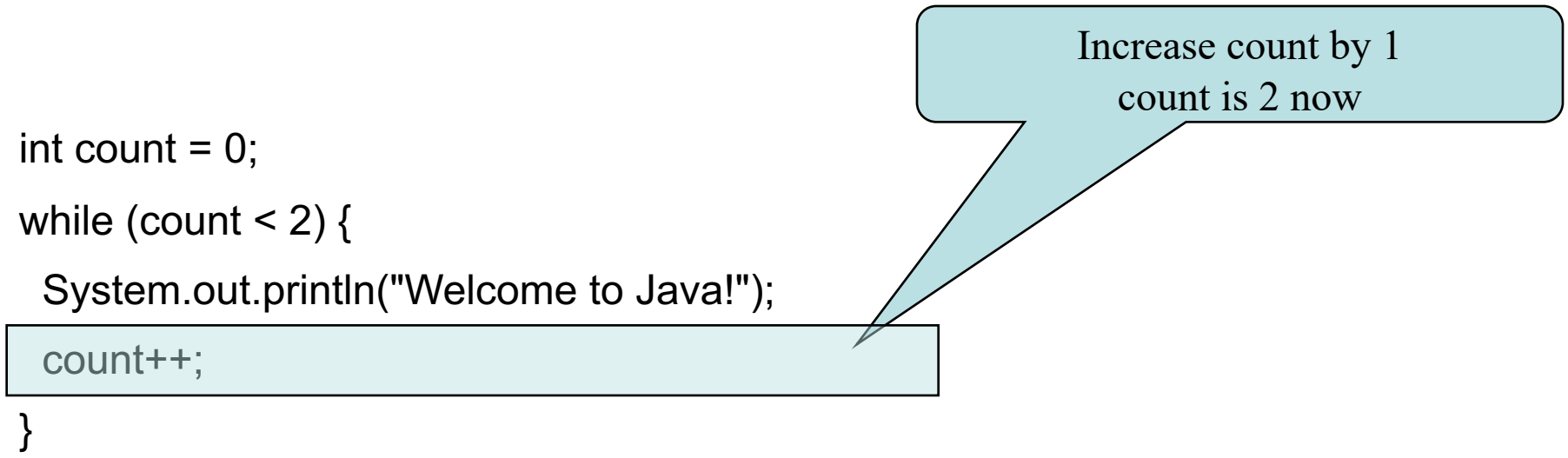
(count < 2) is still true since count is 1

Trace while Loop, cont.




```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Trace while Loop, cont.



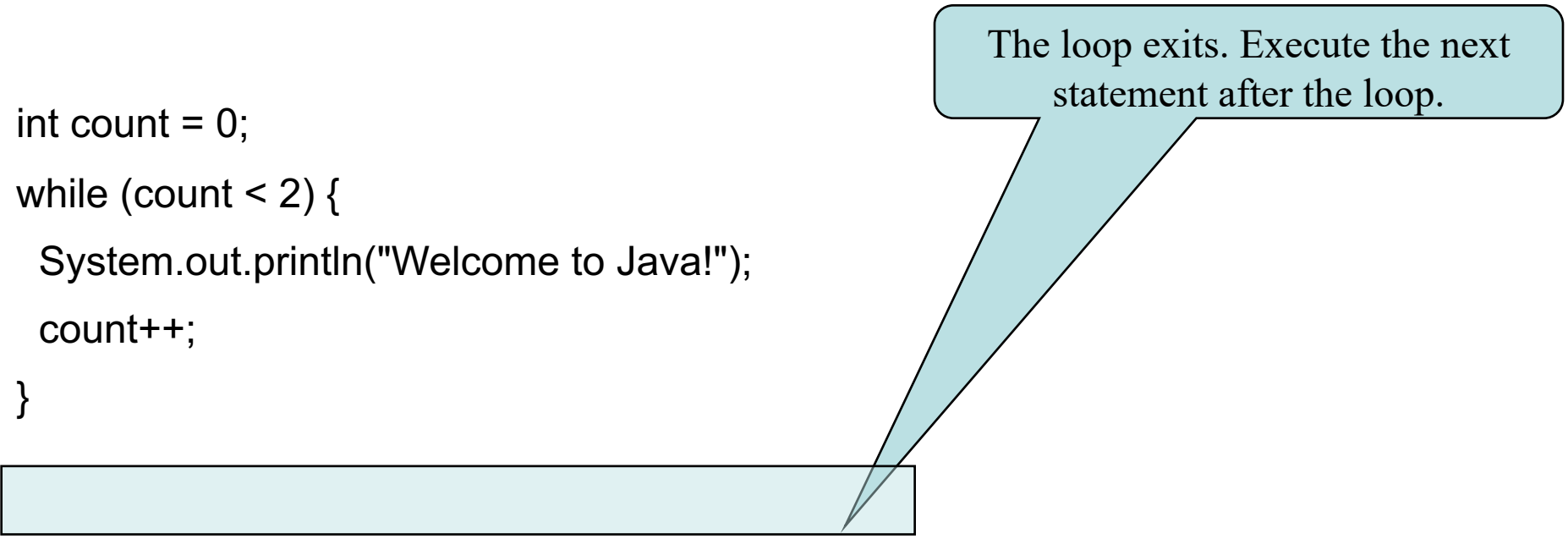
Trace while Loop, cont.

(count < 2) is false since count is 2 now

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Trace while Loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```



Categories of loops

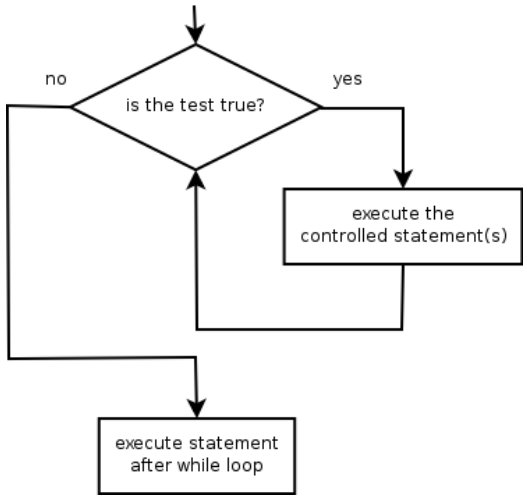
- **definite loop**: Executes a known number of times.
  - The `for` loops:
    - Print "hello" 10 times.
    - Find all the prime numbers up to an integer *n*.
    - Print each odd number between 5 and 127.

- **indefinite loop**: One where the number of times its body repeats is not known in advance.
    - Prompt the user until they type a non-negative number.
    - Print random numbers until a prime number is printed.
    - Repeat until the user has types "q" to quit.

The `while` loop

- **`while` loop**: Repeatedly executes its body as long as a logical test is true.

```
while (test) {
    statement(s);
}
```



- Example:

```
int num = 1;                          // initialization
while (num <= 200) {                  // test
    System.out.print(num + " ");
    num = num * 2;                    // update
}

// output:  1 2 4 8 16 32 64 128
```

Example `while` loop

```
// finds the first factor of 91, other than 1
int n = 91;
int factor = 2;
while (n % factor != 0) {
    factor++;
}
System.out.println("First factor is " + factor);

// output:  First factor is 7
```

- `while` is better than `for` because we don't know how many times we will need to increment to find the factor.

Sentinel values

- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.

- Example: Write a program that prompts the user for numbers until the user types 0, then outputs their sum.
  - (In this case, 0 is the sentinel value.)

```
Enter a number (0 to quit): 10
Enter a number (0 to quit): 20
Enter a number (0 to quit): 30
Enter a number (0 to quit): 0
The sum is 60
```

Flawed sentinel solution

- What's wrong with this solution?

```
Scanner console = new Scanner(System.in);
int sum = 0;
int number = 1;    // "dummy value", anything but 0

while (number != 0) {
    System.out.print("Enter a number (0 to quit): ");
    number = console.nextInt();
    sum = sum + number;
}

System.out.println("The total is " + sum);
```

Changing the sentinel value

- Modify your program to use a sentinel value of -1.
  - Example log of execution:

```
Enter a number (-1 to quit): 15
Enter a number (-1 to quit): 25
Enter a number (-1 to quit): 10
Enter a number (-1 to quit): 30
Enter a number (-1 to quit): -1
The total is 80
```

Changing the sentinel value

- To see the problem, change the sentinel's value to -1:

```
Scanner console = new Scanner(System.in);
int sum = 0;
int number = 1;  // "dummy value", anything but -1

while (number != -1) {
    System.out.print("Enter a number (-1 to quit): ");
    number = console.nextInt();
    sum = sum + number;
}

System.out.println("The total is " + sum);
```

- Now the solution produces the wrong output. Why?

```
The total was 79
```

The problem with our code

- Our code uses a pattern like this:

  sum = 0.
  while (input is not the sentinel) {
  prompt for input; read input.
  add input to the sum.
  }

- On the last pass, the sentinel -1 is added to the sum:

  prompt for input; read input (-1).
  add input (-1) to the sum.

Correct sentinel code

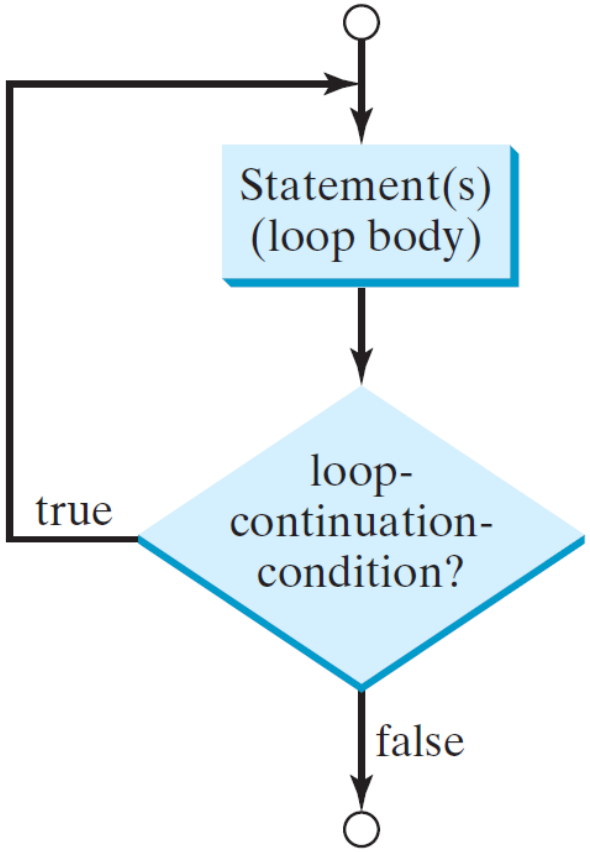
```
Scanner console = new Scanner(System.in);
int sum = 0;

// pull one prompt/read ("post") out of the loop
System.out.print("Enter a number (-1 to quit): ");
int number = console.nextInt();

while (number != -1) {
    sum = sum + number;     // moved to top of loop
    System.out.print("Enter a number (-1 to quit): ");
    number = console.nextInt();
}

System.out.println("The total is " + sum);
```

Sentinel as a constant

```
public static final int SENTINEL = -1;
...
Scanner console = new Scanner(System.in);
int sum = 0;

// pull one prompt/read ("post") out of the loop
System.out.print("Enter a number (" + SENTINEL +
                 " to quit): ");
int number = console.nextInt();

while (number != SENTINEL) {
    sum = sum + number;      // moved to top of loop
    System.out.print("Enter a number (" + SENTINEL +
                     " to quit): ");
    number = console.nextInt();
}

System.out.println("The total is " + sum);
```

do-while Loop

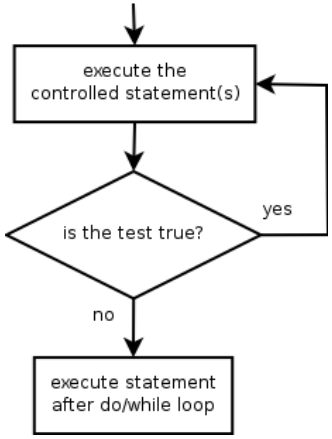
```
do {
  // Loop body;
  Statement(s);
} while (loop-continuation-condition);
```



The `do/while` loop

- **`do/while` loop**: Performs its test at the *end* of each repetition.
  - Guarantees that the loop's `{}` body will run at least once.

```
do {
    statement(s);
} while (test);
```




```
// Example: prompt until correct password is typed
String phrase;
do {
    System.out.print("Type your password: ");
    phrase = console.next();
} while (!phrase.equals("abracadabra"));
```

`do/while` question

- Modify the `Dice` program to use `do/while`.

```
2 + 4 = 6
3 + 5 = 8
5 + 6 = 11
1 + 1 = 2
4 + 3 = 7
You won after 5 tries!
```

- Is `do/while` a good fit for our past `Sentinel` program?

do/while answer

```
// Rolls two dice until a sum of 7 is reached.
import java.util.*;

public class Dice {
    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;
        int sum;

        do {
            int roll1 = rand.nextInt(6) + 1;   // one roll
            int roll2 = rand.nextInt(6) + 1;
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        } while (sum != 7);

        System.out.println("You won after " + tries + " tries!");
    }
}
```

Exercise

- **CHALLENGEACTIVITY**
- 4.3.3: While loop: Insect growth.

Given positive integer numInsects, write a while loop that prints that number doubled without reaching 200. Follow each number with a space. After the loop, print a newline.

Ex: If numInsects = 16, print:16 32 64 128

Anwser

```
import java.util.Scanner;

public class InsectGrowth {
   public static void main (String [] args) {
      Scanner scnr = new Scanner(System.in);
      int numInsects;

      numInsects = scnr.nextInt(); // Must be >= 1

      while (numInsects < 200) {
       System.out.print(numInsects);
       System.out.print(" ");
       numInsects = numInsects * 2;
}
System.out.println("");

   }
}
```