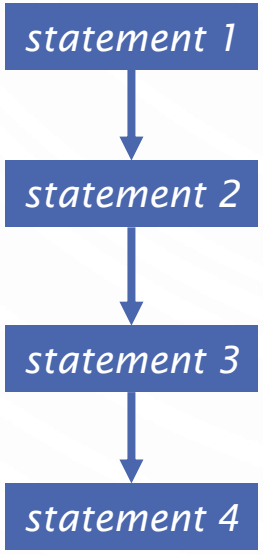
CHAPTER 5
LOOPS

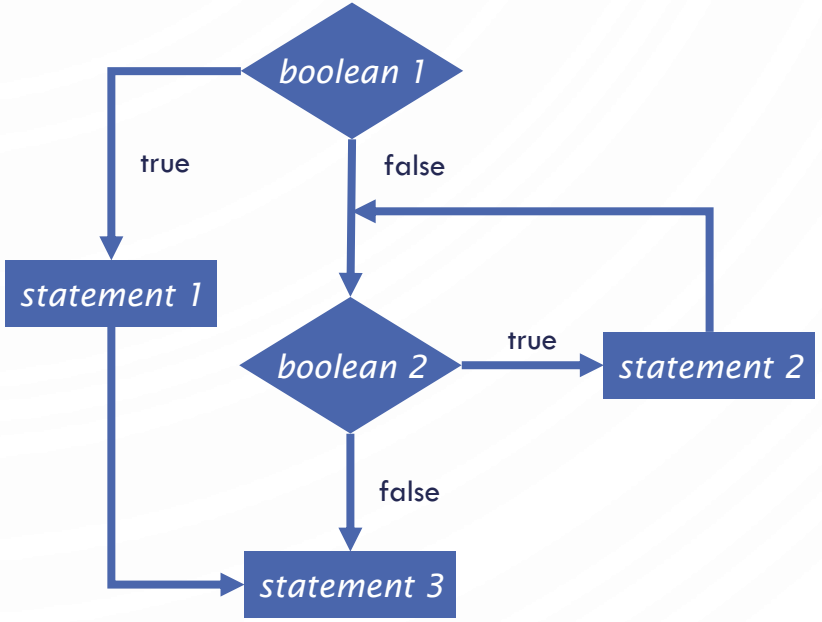
ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO JAVA PROGRAMMING, LIANG (PEARSON 2014)

CONTROL FLOW

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph control flow.



straight-line control flow



control flow with conditionals and 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?

- How about altering our guessing game program to allow 20 tries?

OPENING PROBLEM

```
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!");
```

100 times

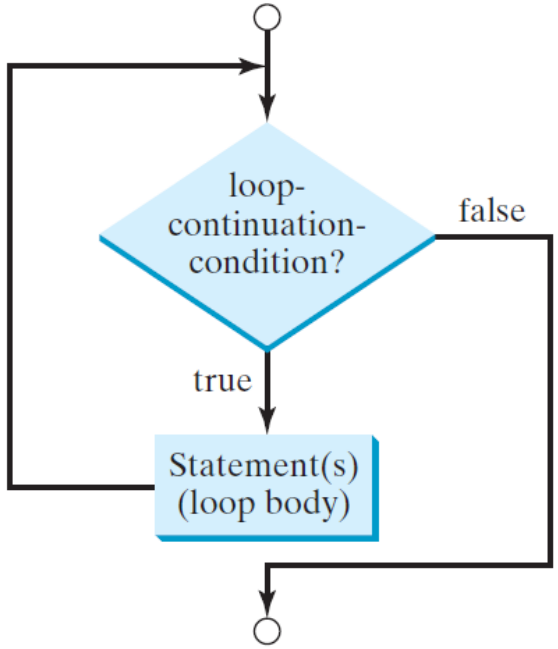
THE WHILE LOOP

INTRODUCING WHILE LOOPS

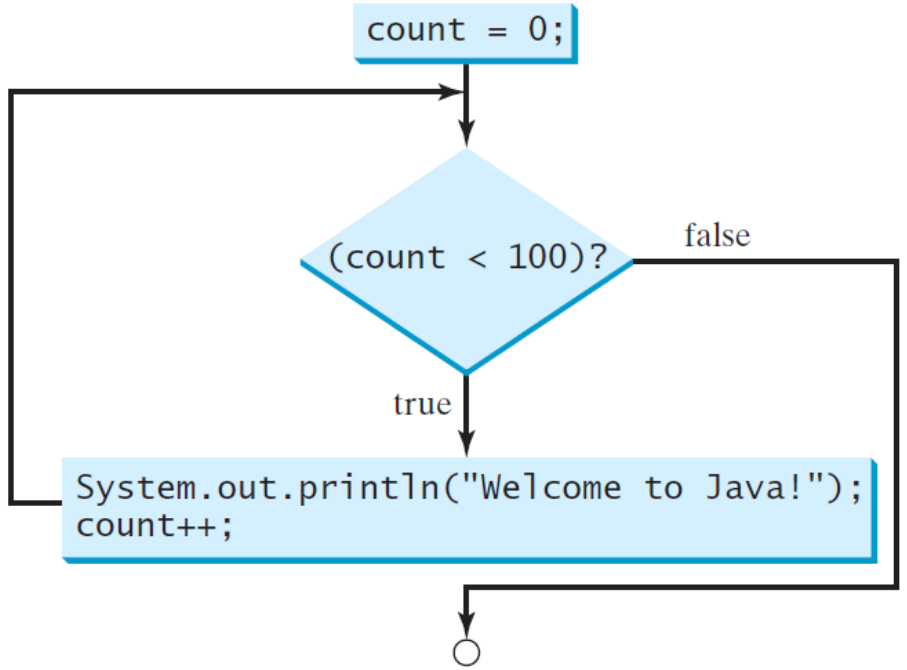
```
1.int count = 0;
2.while(count < 100) {
3.   System.out.println("Welcome to Java");
4.   count++;
5.}
```

WHILE LOOP FLOW CHART

```
1. while (loop-continuation-condition) {
2.   // loop-body;
3.   Statement(s);
4. }
```

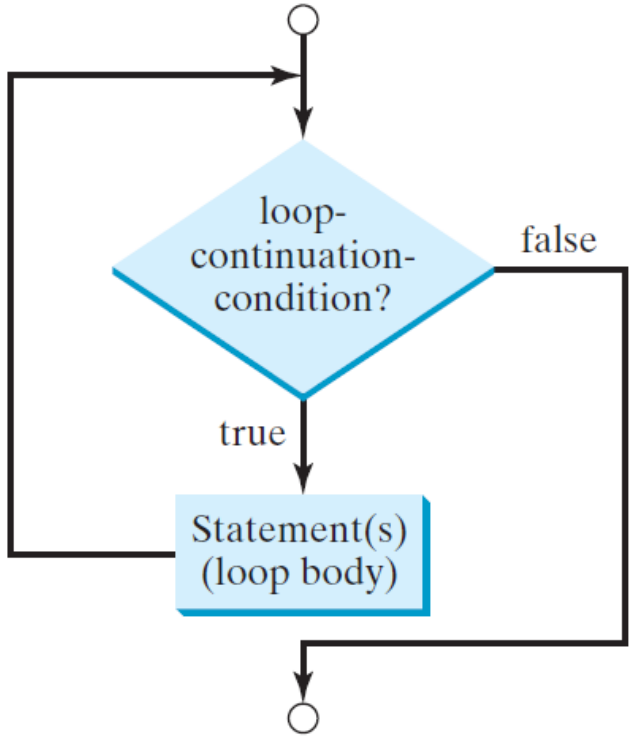



```
1. int count = 0;
2. while(count < 100) {
3.   System.out.println("Welcome to Java!");
4.   count++;
5. }
```



WHILE LOOP IN PSEUDOCODE

```
1. while loop-continuation-condition do
2.    loop-body
```



TRACING WHILE LOOPS

```
1. int count = 0;
2. while(count < 2) {
3.   System.out.println("Welcome to Java");
4.   count++;
5. }
```

Initialize Count

Memory	
Count	0

TRACING WHILE LOOPS

```
1. int count = 0;
2. while(count < 2) {
3.    System.out.println("Welcome to Java");
4.    count++;
5. }
```

Count < 2 is true

Memory	
Count	0

TRACING WHILE LOOPS

```
1.int count = 0;
2.while(count < 2) {
3.    System.out.println("Welcome to Java");
4.    count++;
5.}
```

Output

Memory	
Count	0

TRACING WHILE LOOPS

```
1.int count = 0;
2.while(count < 2) {
3.   System.out.println("Welcome to Java");
4.   count++;
5.}
```

Increment count

Memory	
Count	1

TRACING WHILE LOOPS

```
1. int count = 0;
2. while(count < 2) {
3.    System.out.println("Welcome to Java");
4.    count++;
5. }
```

Count < 2 is true

Memory	
Count	1

TRACING WHILE LOOPS

```
1.int count = 0;
2.while(count < 2) {
3.    System.out.println("Welcome to Java");
4.    count++;
5.}
```

Output

Memory

Count	1

TRACING WHILE LOOPS

```
1.int count = 0;
2.while(count < 2) {
3.   System.out.println("Welcome to Java");
4.   count++;
5.}
```

Increment count

Memory	
Count	2

TRACING WHILE LOOPS

```
1. int count = 0;
2. while(count < 2) {
3.    System.out.println("Welcome to Java");
4.    count++;
5. }
```

Count < 2 is false

Memory	
Count	2

TRACING WHILE LOOPS

```
1.int count = 0;
2.while(count < 2) {
3.   System.out.println("Welcome to Java");
4.   count++;
5.}
```

Memory	
Count	2

EXAMPLES – WITH A PARTNER

- What does are the values of n and m after this:

```
int n = 1234567;
int m = 0;
while(n != 0) {
    m = (10*m) + (n % 10);
    n /= 10;
}
```

- Show the trace of the program at each step

QUESTION

- What is wrong with the following code?
- What happens?
- Fix it and explain what the code outputs

```
1. int i = 0;
2. while (i <= N)
3.     System.out.println(i);
4.     i = i + 5;
```

ACTIVITY

- Write an algorithm (in pseudocode! NOT JAVA) to compute the number of digits an integer has.
  - Example: input – 34567 output – 5
- Bonus: modify your algorithm to compute the number of “digits” for any base, e.g., binary, octal, or hexadecimal

EXERCISE – FIX THE GUESSING GAME

- Lets fix our guessing game program to allow 20 correct guess. We will protect against bad input!
- Program this together
- If you get lost program is on following slides (split into multiple slides)

EXERCISE – FIX THE GUESSING GAME

```
1.  import java.util.Scanner;
2.  import java.util.Random;
3.  public class GuessingGame {
4.    public static void main(String[] args) {
5.      Scanner in = new Scanner(System.in);
6.      Random rand = new Random();
7.      int r = rand.nextInt(100) + 1;
8.      int guess = -1;
9.      int guesses = 0;
10.     //… continue on next slide
11.   }
12. }
```

EXERCISE – FIX THE GUESSING GAME

```
10.     while(guess != r && guesses < 20) {
11.       //… Continue on next slide
12.     }
13.
14.     if(guesses >= 20)
15.       System.out.println("You lost. " +
16.         "Out of guesses. " +
17.         "The correct number is " + r + ".");
```

EXERCISE – FIX THE GUESSING GAME

```
11.         System.out.print("Enter a guess: ");
12.         if(in.hasNextInt()) {
13.           //… Continued on next slide
14.         }
15.         else {
16.           System.out.println("Please enter a number.");
17.           in.next();
18.         }
```

EXERCISE – FIX THE GUESSING GAME

```
13.         guess = in.nextInt();
14.         if(guess < 1 || guess > 100)
15.           System.out.println("Invalid guess.");
16.         else if(guess == r)
17.           System.out.println("You won.");
18.         else if(guess < r) {
19.           System.out.println("Too low.");
20.           ++guesses;
21.         }
22.         else {
23.           System.out.println("Too high.");
24.           ++guesses;
25.         }
```

CAUTION

- Don't use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing 1 + 0.9 + 0.8 + ... + 0.1:

```
1. double item = 1; double sum = 0;
2. while (item != 0) { // No guarantee item will be 0
3.    sum += item;
4.    item -= 0.1;
5. }
6. System.out.println(sum);
```

EXERCISE – WITH A PARTNER

- Random walk. You begin standing at the center of a disk of radius r. At each time-step you pick a random direction in with respect to the x-axis and take a step of 1 meter. How many steps did it take you to fall off?
  - Start at $(x, y) = (0, 0)$; *YES DECIMAL PLACES ARE ALLOWED*
  - Randomly generate theta $\theta \in [0, 2\pi)$
  - Then your new position $(x, y) = (x + \cos(\theta), y + \sin(\theta))$
- Start by planning you algorithm. Then implement it.
- This question has applications to simulating cellular and molecular systems.

ASIDE - FORMATTING OUTPUT

- There is too much to cover in one slide, so here is a link to help
- Basics
  - Use `System.out.printf()` or `System.out.format()`
  - Their first argument is a string. Each time a % appears in the string, it is a directive to substitute it for a variable value. Attach each value after the string (comma separated)
    `System.out.printf("Hello %s", "World");`
  - Use \n in the string to add a new line
- %
  - %s – String
  - %b – Boolean
  - %d – Integer
  - %f – Float/double
  - Etc.
- Examples
  - `System.out.printf("My int: %d", a);`
  - `System.out.printf("My float: %f", d);`

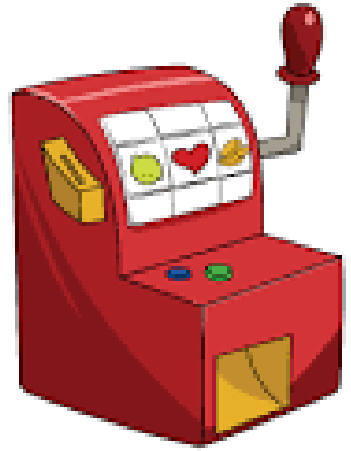
ASIDE - FORMATTING OUTPUT

- The power of printf!
- Can control field width – how many characters are used to output item
  - Can right justify text
- Example %5d – always uses 5 characters to output an integer. Beginning would be white space, not zeroes
- Can also do the same on other types. Floats can determine number of decimal places: %5.7f means 5 characters before the decimal and 7 after
- The possibilities become infinite

```
1. public class PlayWithFormat {
2.     public static void main(String args[]) {
3.         System.out.printf("%5.7f\n", 3.14159);
4.     }
5. }
```

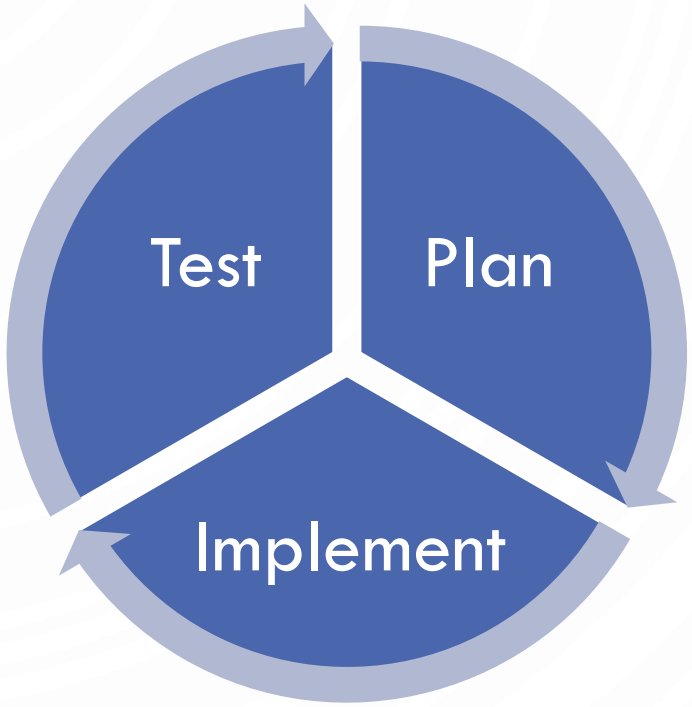
EXERCISE – TOGETHER

- Starters: You work for JLDiablo Consultants Inc., which specializes in making software for Casino games (Cha-ching! $$$$). A new casino in Reno needs a slot game called Binary Slots 101010.
  - How it works:
    - A player enters a bet of their choice
    - Three Boolean values are randomly generated
    - If they are all true, then the player earns twice their money back!



EXERCISE – WHERE TO BEGIN

- When developing programs
  - Always think first!
  - Sketch out solution, i.e., plan
  - Implement solution
  - Test solution
  - Repeat!
- Called iterative development



EXERCISE – START THE PROGRAM

```
1.public class BinarySlots101010 {
2.     public static void main(String[] args) {
3.          System.out.println(
4.            "Welcome to Binary Slots 101010!\n\n\n");
5.     }
6.}
```

EXERCISE – GET BET

```
1. import java.util.Scanner;
2. public class BinarySlots101010 {
3.     public static void main(String[] args) {
4.         System.out.println("Welcome to Binary Slots 101010!\n\n\n");
5.
6.         System.out.print("Please enter your bet: ");
7.         Scanner in = new Scanner(System.in);
8.         double bet = in.nextDouble();
9.         System.out.printf("Your bet is $.2f\n\n", bet);
10.    }
11.}
```

EXERCISE – GET BET ROBUSTLY

```
1. import java.util.Scanner;
2. public class BinarySlots101010 {
3.     public static void main(String[] args) {
4.         System.out.println("Welcome to Binary Slots 101010!\n\n\n");
5.
6.         System.out.print("Please enter your bet: ");
7.         Scanner in = new Scanner(System.in);
8.         while(!in.hasNextDouble()) {
9.             System.out.println("Please enter a valid bet: ");
10.            in.next(); //Remember to eat up (read) bad input…
11.        }
12.        double bet = in.nextDouble();
13.        System.out.printf("Your bet is $.2f\n\n", bet);
14.        //… continued on next slide
15.    }
16.}
```

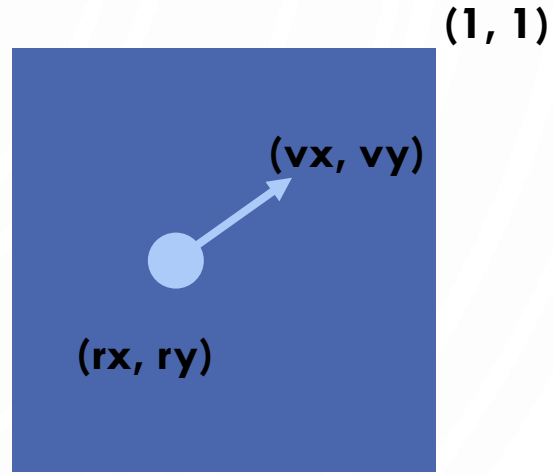
EXERCISE – GAME LOGIC

```
14.         System.out.println("Spinning….match all to win!\n");
15.         boolean a = Math.random() < 0.5,
16.           b = Math.random() < 0.5,
17.           c = Math.random() < 0.5;
18.         System.out.println("Binary slots: " +
              a + " " + b + " " + c + "\n");
19.
20.         if(a && b && c)
21.           System.out.printf("You win 2x your bet! You won $%.2f\n",
                2*bet);
22.         else
23.           System.out.println("Sorry you lose…");
```

VISUAL DISPLAY FOR SLOT MACHINE

ANIMATION

- Animation loop. Repeat the following:
  - Clear the screen.
  - Move the object.
  - Draw the object.
  - Display and pause for a short while.



- Ex. Bouncing ball.
  - Ball has position (rx, ry) and constant velocity (vx, vy).
  - Detect collision with wall and reverse velocity.

BOUNCING BALL

```
1.  public class BouncingBall {
2.     public static void main(String[] args) {
3.        double rx = .480, ry = .860;                                  //x, y position
4.        double vx = .015, vy = .023;                                  //x, y velocity
5.        double radius = .05;                                  //radius of ball
6.        StdDraw.setXscale(-1.0, 1.0);
7.        StdDraw.setYscale(-1.0, 1.0);       //Set coordinate system
8.
9.        while(true) { //Simulation loop
10.          if (Math.abs(rx + vx) + radius > 1.0) vx = -vx; //Update ball velocity if at the boundary
11.          if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;
12.
13.          rx = rx + vx; ry = ry + vy;         //Update position (add velocity)
14.
15.          StdDraw.setPenColor(StdDraw.GRAY);                        //Clear screen
16.          StdDraw.filledSquare(0.0, 0.0, 1.0);
17.          StdDraw.setPenColor(StdDraw.BLACK);                       //Render ball
18.          StdDraw.filledCircle(rx, ry, radius);
19.          StdDraw.show();  StdDraw.pause(20); //Pause for 20ms so that we can see it nicely
20.       }
21.    }
22. }
```

EXERCISE – INITIALIZATION STEP

```
1. //Initialization. Add this before anything
2. StdDraw.setCanvasSize(600, 600);      // Resize the window
3. StdDraw.setXscale(-100, 100); //Redefine coordinate system
4. StdDraw.setYscale(-100, 100);
5. StdDraw.enableDoubleBuffering();      //Allows for smooth animation
```

EXERCISE – GAME LOOP

```
1. //Play game until quit 'p' is pressed. We will inject game logic into this
2. while(true) {
3.     //Wait for key press
4.     while(!StdDraw.hasNextKeyTyped()) {
5.         StdDraw.show();
6.         StdDraw.pause(33);
7.     }
8.     char key = StdDraw.nextKeyTyped();
9.     if(key == 'q')
10.        break;
11.}
12.System.exit(0);
```

EXERCISE – BEFORE THE KEY PRESSED. SHOW A MESSAGE FOR USER TO START THE GAME

```
1. //Draw slot machine.
2. StdDraw.textRight(90, -70, "Binary Slots 101!");
3. StdDraw.textRight(90, -80, "Match 3 to win");
4. StdDraw.textRight(90, -90, "Press a key to play. Q to quit.");
5. StdDraw.circle(-55, -5, 10);
6. StdDraw.circle(-5, -5, 10);
7. StdDraw.circle(45, -5, 10);
```

EXERCISE – NOW WE CAN DO OUR MAIN GAME LOOP

```
1.  //Simulate spin. Randomly switch color of circles
2.  boolean a = false, b = false, c = false; int i = 1;
3.  while(i <= 30) {
4.    StdDraw.clear();                                          //Clear
5.
6.    StdDraw.textRight(90, -70, "Binary Slots 101!");          //Redraw text
7.    StdDraw.textRight(90, -80, "Match 3 to win");
8.
9.    a = Math.random() < 0.5; b = Math.random() < 0.5;  c = Math.random() < 0.5;  //Simulate game
10.
11.   if(a) StdDraw.setPenColor(StdDraw.BLUE);                  //Draw all the circles
12.   else  StdDraw.setPenColor(StdDraw.RED);
13.   StdDraw.filledCircle(-55, -5, 10);
14.   if(b) StdDraw.setPenColor(StdDraw.BLUE);
15.   else  StdDraw.setPenColor(StdDraw.RED);
16.   StdDraw.filledCircle(-5, -5, 10);
17.   if(c) StdDraw.setPenColor(StdDraw.BLUE);
18.   else  StdDraw.setPenColor(StdDraw.RED);
19.   StdDraw.filledCircle(45, -5, 10);
20.   StdDraw.setPenColor(StdDraw.BLACK);
21.   StdDraw.circle(-55, -5, 10); StdDraw.circle(-5, -5, 10); StdDraw.circle(45, -5, 10);
22.
23.   StdDraw.show();                                           //Render
24.   StdDraw.pause(i++*10);                                    //Pause
25. }
```

EXERCISE – FINISH OFF WITH A FINAL MESSAGE

```
1.//Draw final slot machine
2.StdDraw.pause(1000);
3.if(a && b && c)
4.   StdDraw.text(-20, 50, "You win!");
5.else
6.   StdDraw.text(-20, 50, "You lose :(");
7.StdDraw.show();
```

THE DO-WHILE LOOP

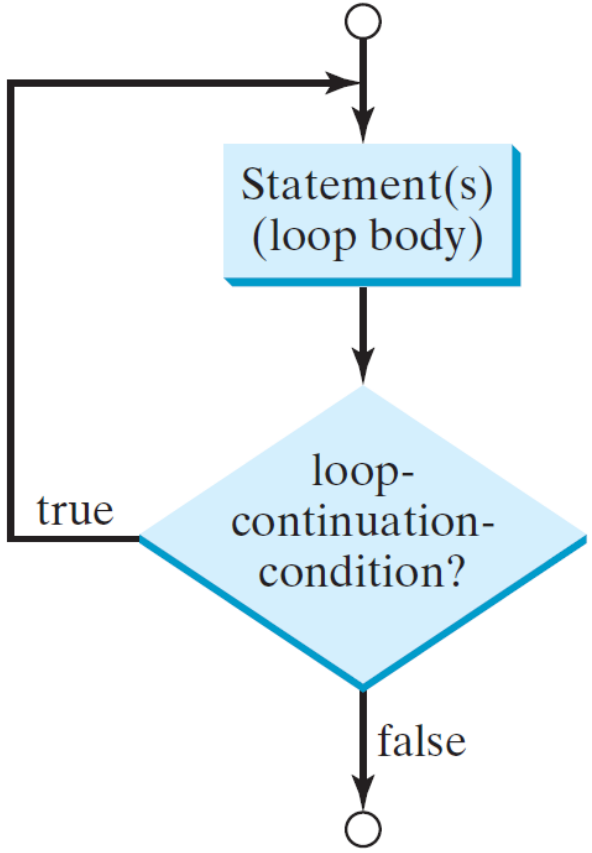
DO-WHILE LOOP

- In Java

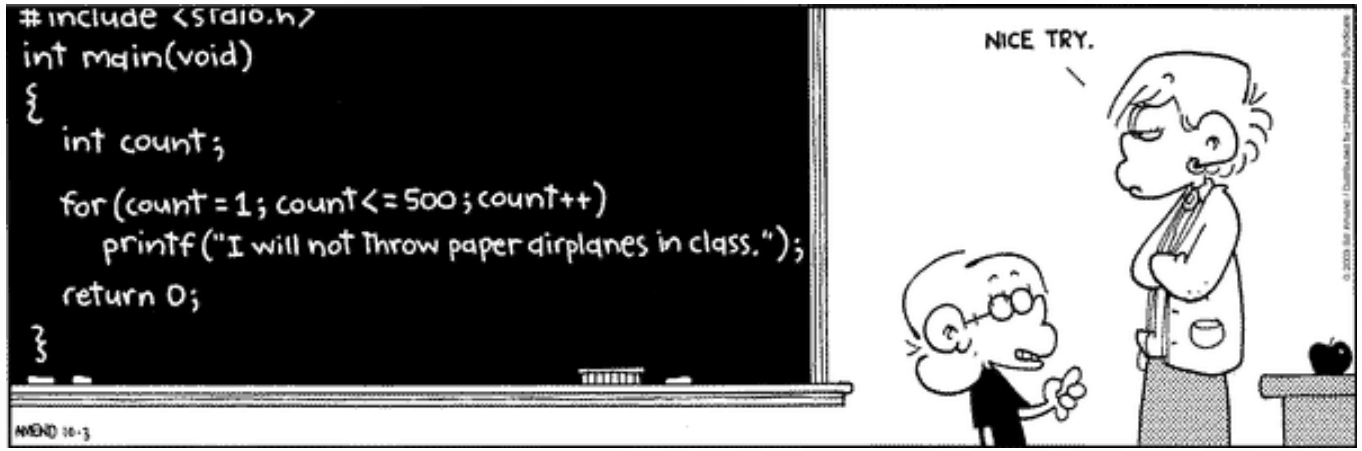
```
1. do {
2.    // Loop body;
3.    Statement(s);
4. } while (loop-continuation-condition);
```

- In pseudocode

```
1. repeat
2.    loop-body
3. until loop-continuation-condition
```

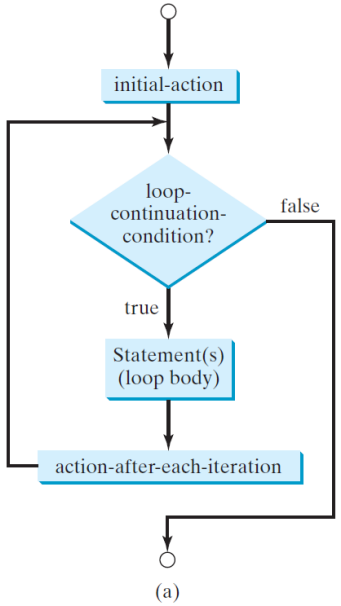


THE FOR LOOP


#include <stdio.h>
int main(void)
{
int count;
for (count = 1; count <= 500; count++)
printf ("I will not throw paper airplanes in class.");
return 0;
}
NICE TRY.

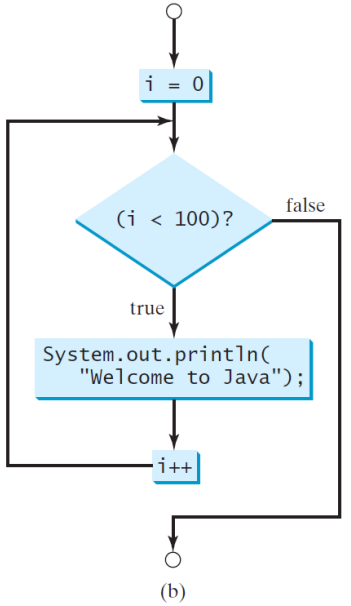
FOR LOOPS

```
1. for (initial-action;
       loop-continuation-condition;
       action-after-each-iteration) {
2.     // loop body;
3.     Statement(s);
4. }
```



(a)

```
1. for(int i = 0; i < 100; i++) {
2.   System.out.println("Welcome to Java!");
3. }
```



(b)

FOR LOOPS IN PSEUDOCODE

- With variable

```
1. for i ← 1...n do
2.   loop body
```

- Over all elements of set

```
1. for e ∈ E do
2.   loop body
```

TRACING FOR LOOPS

```
1. for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3. }
```

Initialize Count

Memory	
Count	0

TRACING FOR LOOPS

```
1.for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3.}
```

i < 2 is true

Memory	
Count	0

TRACING FOR LOOPS

```
1. for(int i = 0; i < 2; i++) {
2.     System.out.println("Welcome to Java!");
3. }
```

Output

Memory

Count	0

TRACING FOR LOOPS

```
1.for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3.}
```

Memory	
Count	1

TRACING FOR LOOPS

```
1.for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3.}
```

i < 2 is true

Memory	
Count	1

TRACING FOR LOOPS

```
1. for(int i = 0; i < 2; i++) {
2.     System.out.println("Welcome to Java!");
3. }
```

Memory	
Count	1

TRACING FOR LOOPS

```
1.for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3.}
```

Memory	
Count	2

TRACING FOR LOOPS

```
1. for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3. }
```

i < 2 is false

Memory	
Count	2

TRACING FOR LOOPS

```
1.for(int i = 0; i < 2; i++) {
2.   System.out.println("Welcome to Java!");
3.}
```

Memory

~~Count 1~~

PRACTICE

- Table 1: Write a for loop to output all numbers between integers a and b
- Table 2: Write a for loop to output all powers of two up to a number N
- Table 3: Write a for loop to output the multiples of an integer a up to N
- Table 4: Write a for loop to output all the even numbers from 100 to 999 in reverse order.

NOTE

- The *initial-action* in a for loop can be a list of zero or more comma-separated expressions. The *action-after-each-iteration* in a for loop can be a list of zero or more comma-separated statements. Therefore, the following two for loops are correct. The first is never used in practice, however.

```
1.for(int i = 1; i < 100; System.out.println(i++));
2.
3.for(int i = 0, j = 0; (i + j < 10); i++, j++) {
4.   // Do something
5.}
```

NOTE

- If the *loop-continuation-condition* in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {
  // Do something
}
```

(a)

Equivalent

```
while (true) {
  // Do something
}
```

(b)

CAUTION

- Adding a semicolon at the end of the for (or while) clause before the loop body is a common mistake, as shown below:

```
1. for(int i=0; i<10; i++);
2. {
3.   System.out.println("i is " + i);
4. }
```

WHICH LOOP TO USE?

- The three forms of loop statements, while, do-while, and for, are expressively equivalent; that is, you can write a loop in any of these three forms. For example, a while loop in (a) in the following figure can always be converted into the following for loop in (b):

```
while (loop-continuation-condition) {
  // Loop body
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )
  // Loop body
}
```

(b)

- A for loop in (a) in the following figure can generally be converted into the following while loop in (b) except in certain special cases (see Review Question 3.19 for one of them):

```
for (initial-action;
     loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
}
```

(a)

Equivalent

```
initial-action;
while (loop-continuation-condition) {
  // Loop body;
  action-after-each-iteration;
}
```

(b)

COMPARISON OF LOOPS

- for loop – used when you know how many times to execute or each iteration has a natural increment
- while loop – used to execute 0 or more times. Pre-condition check.
- do-while loop – used to execute 1 or more time. Post-condition check.

NESTING

NESTING

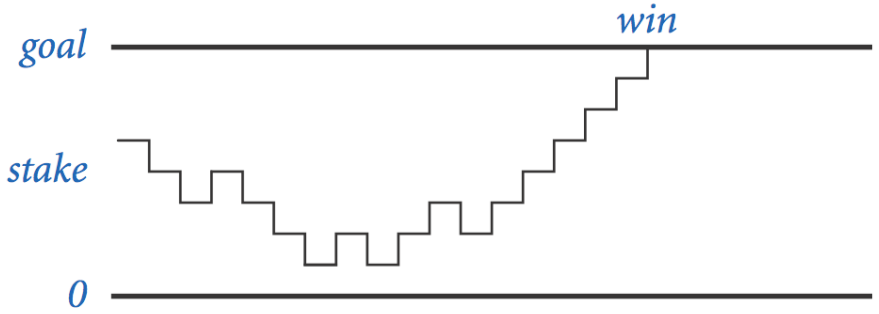
- In control flow, nesting is where you place a control structure inside of another
- Example: 2 for loops to print a multiplication table

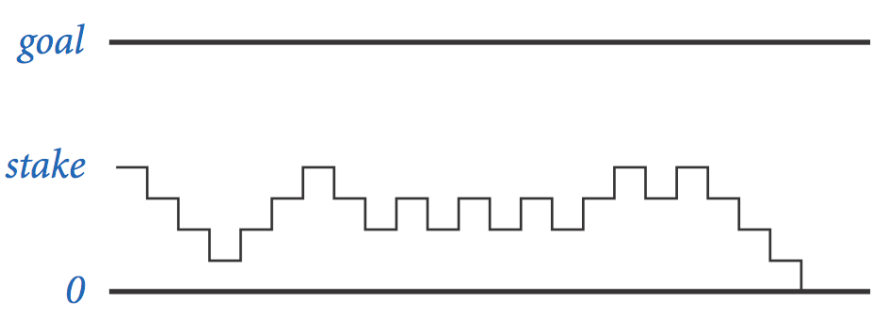
```
1. for(int i = 0; i < 10; ++i) {
2.     for(int j = 0; j < 10; ++j)
3.         System.out.printf("%d*%d = %2d\t", i, j, i*j);
4.     System.out.println();
5. }
```

MONTE CARLO SIMULATION

GAMBLER'S RUIN

- Gambler's ruin. Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal.
  - What are the chances of winning?
  - How many bets will it take?
- One approach. Monte Carlo simulation.
  - Flip digital coins and see what happens.
  - Repeat and compute statistics.





GAMBLER'S RUIN

```
1.  public class Gambler {
2.      public static void main(String[] args) {
3.          int stake = Integer.parseInt(args[0]), goal = Integer.parseInt(args[1]); T = Integer.parseInt(args[2]);
4.          int wins = 0;
5.          // repeat experiment T times
6.          for (int t = 0; t < T; t++) {
7.              // do one gambler's ruin experiment
8.              int cash = stake;
9.              while (cash > 0 && cash < goal) {
10.                 // flip coin and update
11.                 if (Math.random() < 0.5) cash++;
12.                 else                     cash--;
13.             }
14.             if (cash == goal) wins++;
15.         }
16.         System.out.println(wins + " wins of " + T);
17.     }
18. }
```

```
% java Gambler 5 25 1000
191 wins of 1000

% java Gambler 5 25 1000
203 wins of 1000

% java Gambler 500 2500 1000
197 wins of 1000
```

OTHER CONTROL FLOW STATEMENTS

OTHER HELPFUL STATEMENTS FOR LOOPS

- **break** – immediately exit the loop. Do not continue executing any more of the loop:

```
while(true) {
    if(q-key-is-pressed())
        //quit the game
        break;
    Game-loop();
}
```

- **continue** – immediately skip to the end of the body of the loop, i.e., start next iteration (checking the condition):

```
for(int i = 0; i < 10; ++i) {
    if(isPrime(i))
    //OCD against prime numbers
        continue;
    HandleNotPrimes();
}
```

SCOPE

SCOPE OF LOCAL VARIABLES

- **Scope** – the "lifetime" of a variable
  - The part of the program where the variable can be referenced.
- The scope of a variable starts from its *declaration* and continues to the end of the *block* that contains the variable.
-
```
for(int i = 0; i < n; ++i) {
   //Can refer to i here
}
//Cannot refer to i here
```

SCOPE OF LOCAL VARIABLES

- The scope of a variable declared in the initial action part of a for loop is the entire loop. But a variable declared inside a for loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.

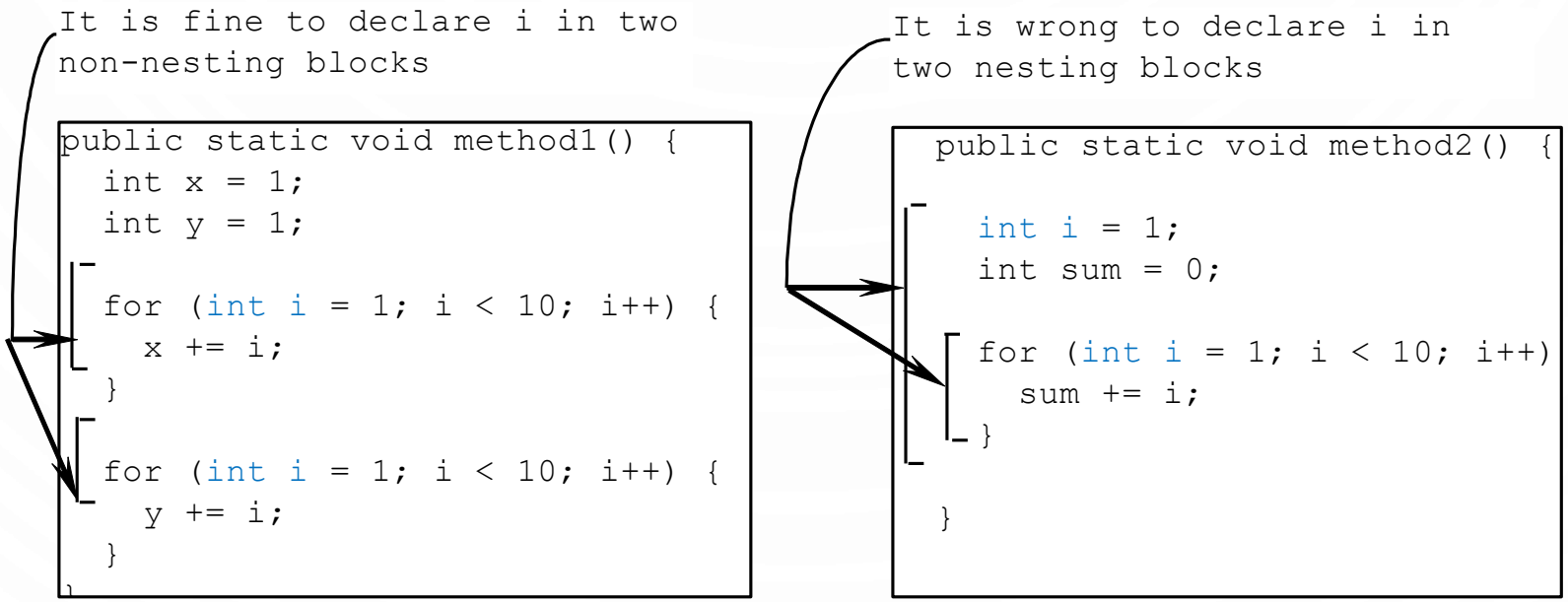
```
public static void main(
    String[] args) {
  // stuff
  for (int i = 1; i < 10; i++) {
    // more stuff
    int j;
    // even more stuff
  }
}
```

Scope of `i`

Scope of `j`

SCOPE OF LOCAL VARIABLES

- You can declare a variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks (…in Java).



CONTROL FLOW SUMMARY

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph the control flow.

Control Flow	Description	Examples
Straight-line programs	All statements are executed in the order given	
Conditionals	Certain statements are executed depending on the values of certain variables	**if; if-else; switch**
Loops	Certain statements are executed repeatedly until certain conditions are met	**while; for; do-while**