جامعة الجلالة
GALALA UNIVERSITY

CSE110 Principles of Programming

Lecture 2: Control Statements Part 2

Professor Shaker El-Sappagh

Shaker.elsappagh@gu.edu.eg

Fall 2023

Outline

1. Loops by while statement
2. Loops by for statement
3. Loops by do-while statement
4. Nested loops
5. The `break` and `continue` Statements

The `while` Loop

- Java provides three different looping structures.
- The `while` loop has the form:

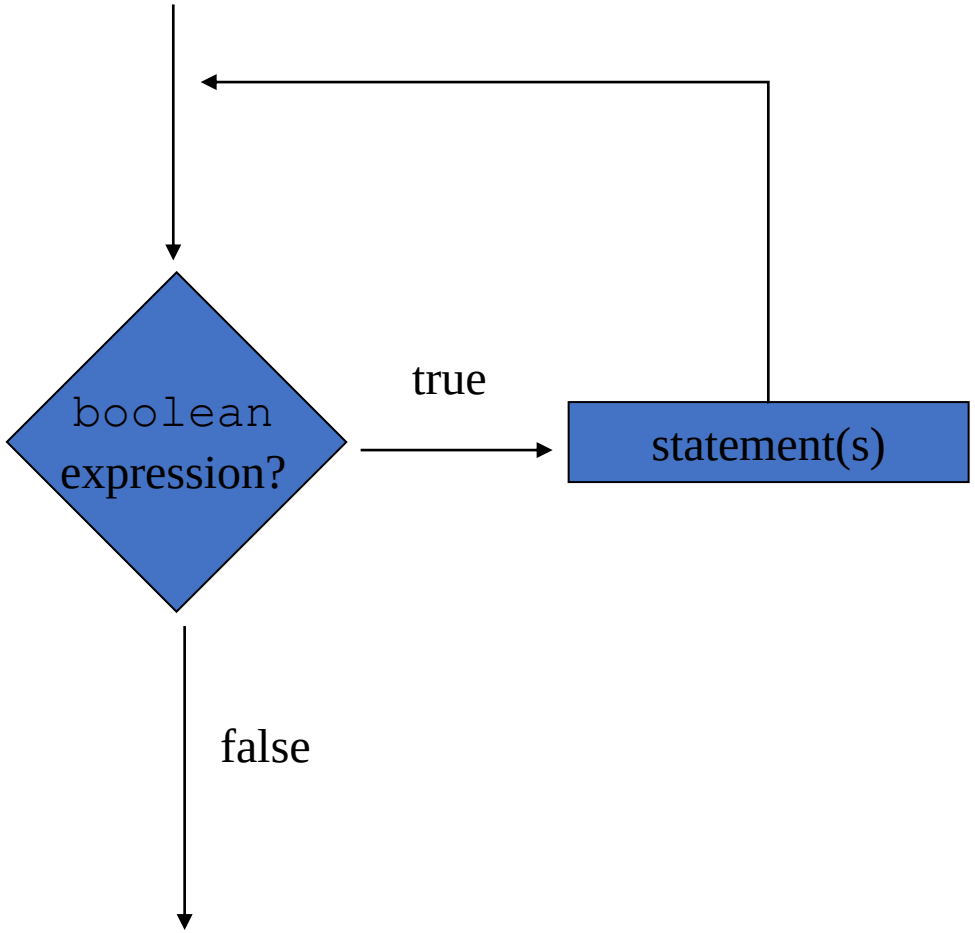
```
while(condition)
{
    statements;
}
```

- While the condition is true, the statements will execute repeatedly.
- The `while` loop is a ***pretest*** loop, which means that it will test the value of the condition prior to executing the loop.

The `while` Loop

- Care must be taken to set the condition to false somewhere in the loop so the loop will end.
- Loops that do not end are called *infinite loops.*
- A `while` loop executes 0 or more times. If the condition is false, the loop will not execute.

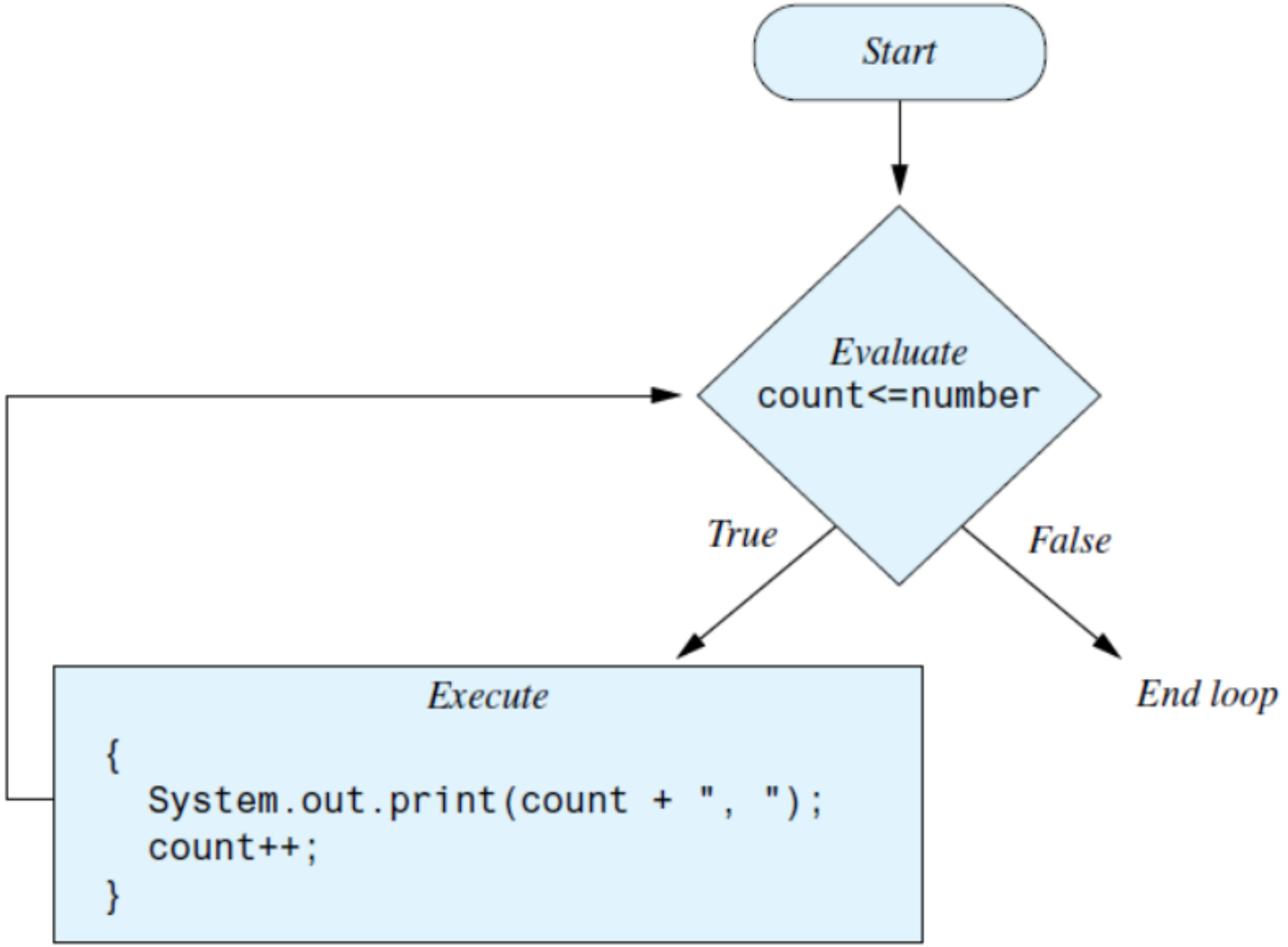
The `while` loop Flowchart




```
initialization;

while (loopContinuationCondition) {
    statements
    increment;
}
```

The `while` loop

```
while (count <= number)
{
   System.out.print(count + ", ");
   count++;
}
```



The `while` Loop

```
public class WhileLoop
{
   public static void main(String[] args)
   {
      int number = 1;

      while (number <= 5)
      {
         System.out.println("Hello");
         number++;
      }
        System.out.println("That's all!");
   }
}
```

Program Output

```
Hello
Hello
Hello
Hello
Hello
That's all!
```

Infinite Loops

- In order for a `while` loop to end, the condition must become false. The following loop will not end:

```
int x = 20;
while(x > 0)
{
    System.out.println("x is greater than 0");
}
```

- The variable `x` never gets decremented so it will always be greater than 0.
- Adding the **`x--`** above fixes the problem.

Infinite Loops

- This version of the loop decrements `x` during each iteration:

```
int x = 20;
while(x > 0)
{
   System.out.println("x is greater than 0");
   x--;
}
```

Block Statements in Loops

- Curly braces are required to enclose block statement while loops. (like block `if` statements)

```
while (condition)
{
    statement;
    statement;
    statement;
}
```

The `while` Loop for Input Validation

- *Input validation* is the process of ensuring that user input is valid.

```
System.out.print("Enter a number in the " +
                 "range of 1 through 100: ");
number = keyboard.nextInt();
// Validate the input.
while (number < 1 || number > 100)
{
  System.out.println("That number is invalid.");
  System.out.print("Enter a number in the " +
                   "range of 1 through 100: ");
  number = keyboard.nextInt();
}
```

The `while` Loop Exercises

1. What output is produced by the following code?

```
int count = 0;
while (count < 5)
{
   System.out.println(count);
   count++;
}
System.out.println("count after loop = " + count);
```

2. Can the body of a `while` loop execute zero times? Can the body of a `do-while` loop execute zero times?

3. What output is produced by the following code?

```
int count = 0;
do
{
   System.out.println(count);
   count++;
}
while (count < 0);
System.out.println("count after loop = " + count);
```

5. What output is produced by the following code?

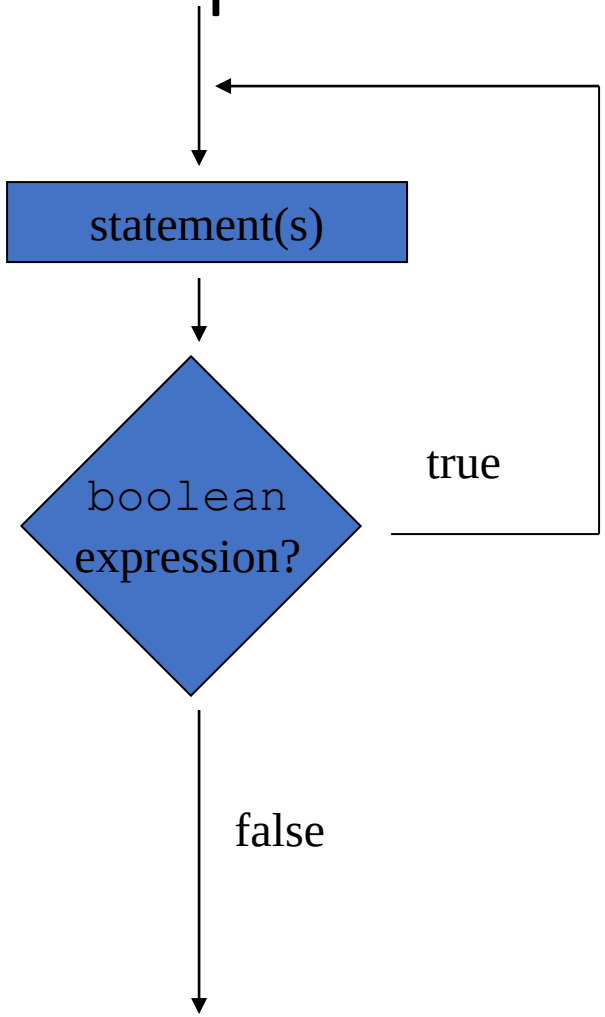
```
int count = 0;
while (count < 5)
{
   System.out.println(count);
   count--;
}
System.out.println("count after loop = " + count);
```

The `do-while` Loop

- The `do-while` loop is a *post-test* loop, which means it will execute the loop prior to testing the condition.
- The `do-while` loop (sometimes called called a `do` loop) takes the form:

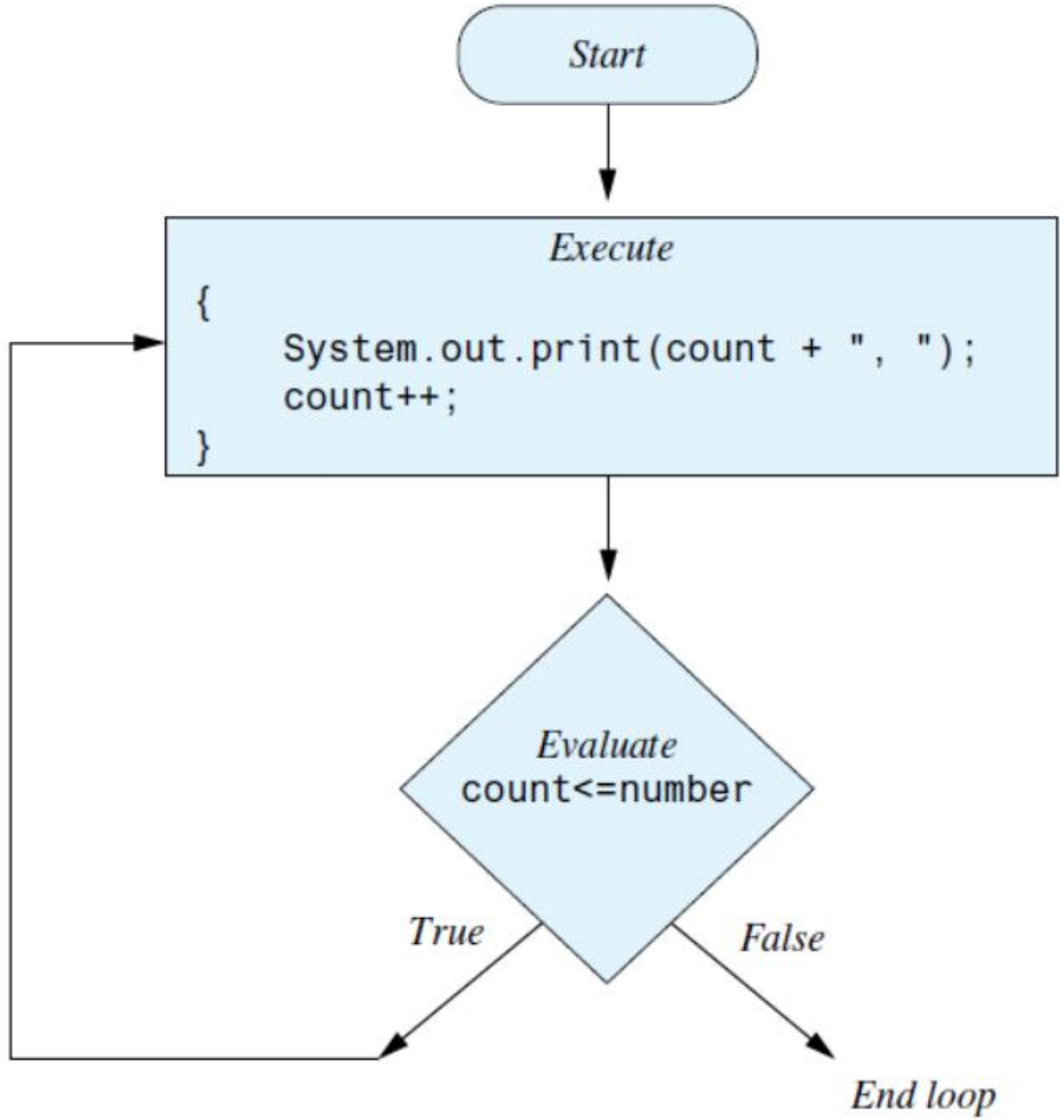
```
do
{
  statement(s);
}while (condition);
```

The `do-while` Loop Flowchart



The `do-while` Loop

```
do
{
   System.out.print(count + ", ");
   count++;
} while (count <= number);
```



The `do-while` Loop: example

```
public class DoWhileTest {
   public static void main(String[] args) {
      int counter = 1;

      do {
         System.out.printf("%d  ", counter);
         ++counter;
      } while (counter <= 10);

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

```
1  2  3  4  5  6  7  8  9  10
```

```
public class TestAverage1
{
   public static void main(String[] args)
   {
      int score1, score2, score3;      // Three test scores
      double average;                  // Average test score
      char repeat;                     // To hold 'y' or 'n'
      String input;                    // To hold input

      System.out.println("This program calculates the " +
                         "average of three test scores.");

      // Create a Scanner object for keyboard input.
      Scanner keyboard = new Scanner(System.in);

      // Get as many sets of test scores as the user wants.
      do
      {
         // Get the first test score in this set.
         System.out.print("Enter score #1: ");
         score1 = keyboard.nextInt();

         // Get the second test score in this set.
         System.out.print("Enter score #2: ");
         score2 = keyboard.nextInt();

         // Get the third test score in this set.
         System.out.print("Enter score #3: ");
         score3 = keyboard.nextInt();

         // Consume the remaining newline.
         keyboard.nextLine();

         // Calculate and print the average test score.
         average = (score1 + score2 + score3) / 3.0;
         System.out.println("The average is " + average);
         System.out.println(); // Prints a blank line

         // Does the user want to average another set?
         System.out.println("Would you like to average " +
                            "another set of test scores?");
         System.out.print("Enter Y for yes or N for no: ");
         input = keyboard.nextLine();    // Read a line.
         repeat = input.charAt(0);       // Get the first char.

      } while (repeat == 'Y' || repeat == 'y');
   }
}
```

```
public class TestAverage1
{
   public static void main(String[] args)
   {
      int score1, score2, s
      double average;
      char repeat;
      String input;

      System.out.println("T
                         "a


      // Create a Scanner c
      Scanner keyboard = ne

      // Get as many sets c
      do
      {
         // Get the first t
         System.out.print("
         score1 = keyboard.


         // Get the second test score in this set.
         System.out.print("Enter score #2: ");
         score2 = keyboard.nextInt();
```

Program Output with Example Input Shown in Bold

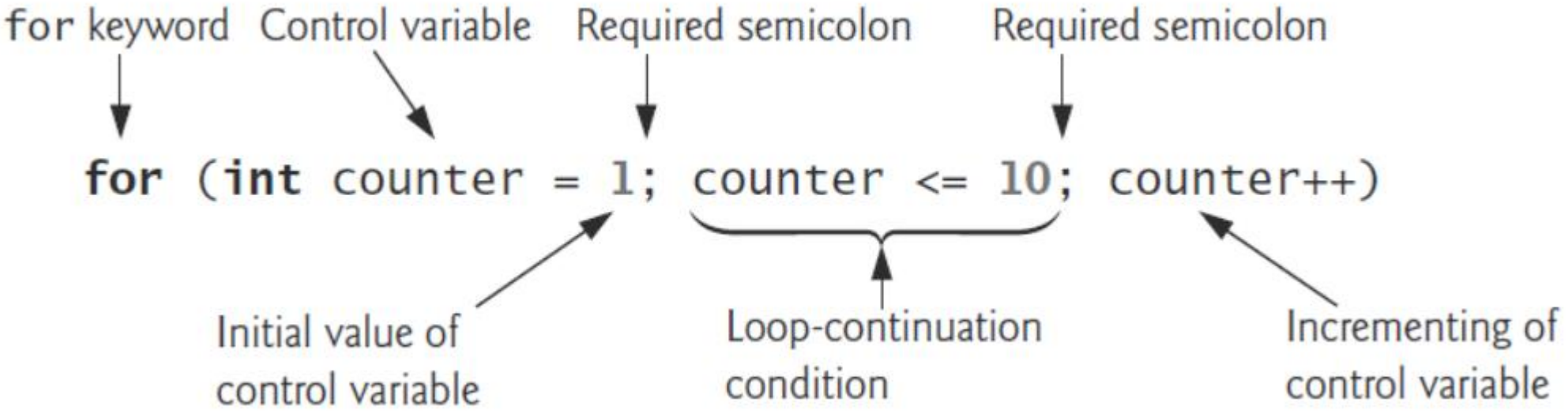
```
This program calculates the average of three test scores.
Enter score #1: 89 [Enter]
Enter score #2: 90 [Enter]
Enter score #3: 97 [Enter]
The average is 92.0

Would you like to average another set of test scores?
Enter Y for yes or N for no: y [Enter]
Enter score #1: 78 [Enter]
Enter score #2: 65 [Enter]
Enter score #3: 88 [Enter]
The average is 77.0

Would you like to average another set of test scores?
Enter Y for yes or N for no: n [Enter]
```

```
         // Get the third test score in this set.
         System.out.print("Enter score #3: ");
         score3 = keyboard.nextInt();

                                                    line.


                                                    verage test score.
                                                     score3) / 3.0;
                                                    age is " + average);
                                                    nts a blank line


                                                    rage another set?
                                                    u like to average " +
                                                    set of test scores?");
                                                    r yes or N for no: ");
                                                       // Read a line.
                                                       // Get the first char.

         } while (repeat == 'Y' || repeat == 'y');
      }
   }
```

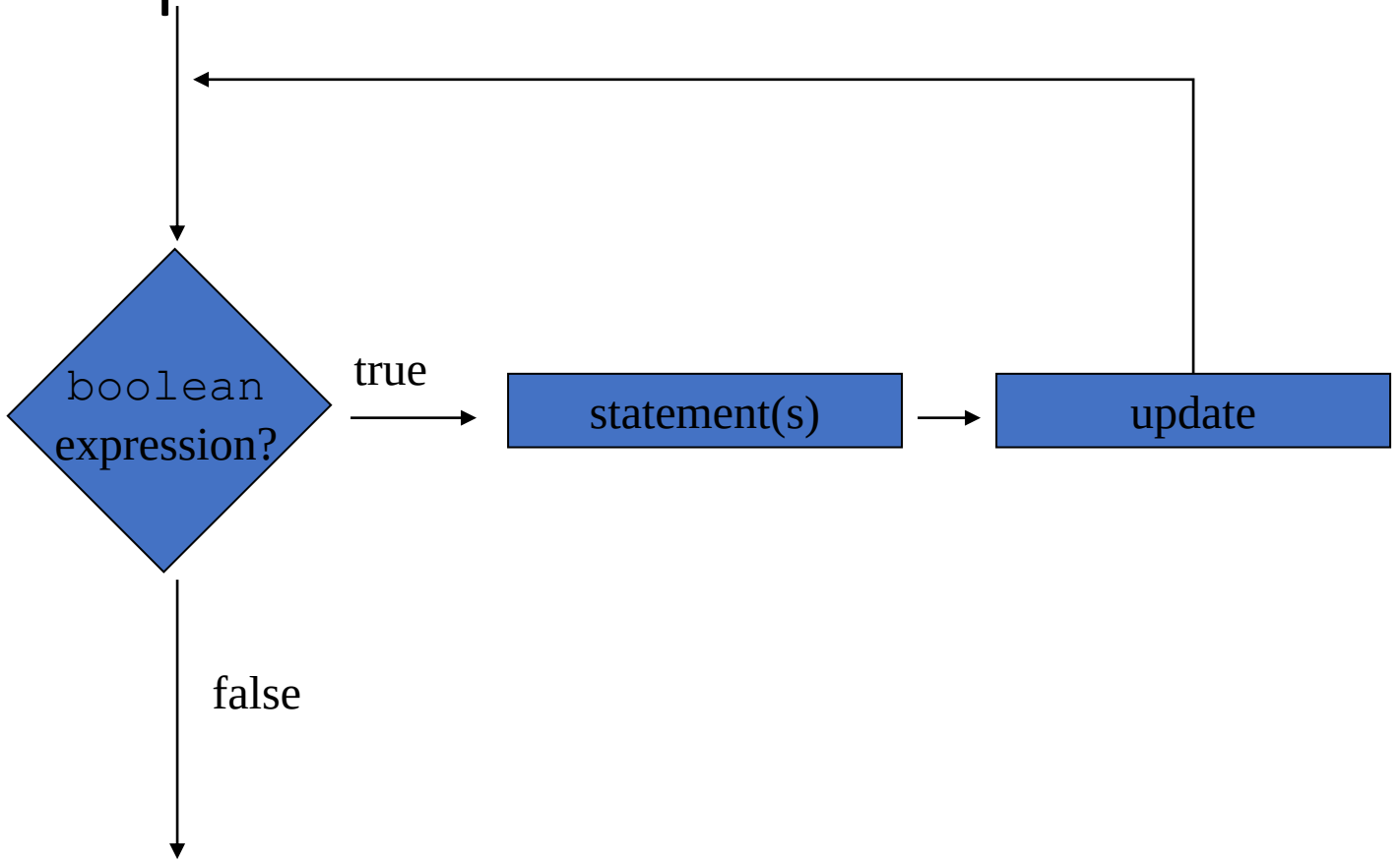
The `for` Loop

- The `for` loop is a pre-test loop.
- The `for` loop allows the programmer to initialize a control variable, test a condition, and modify the control variable all in one line of code.
- The `for` loop takes the form:

```
for(initialization; test; update)
{
    statement(s);
}
```



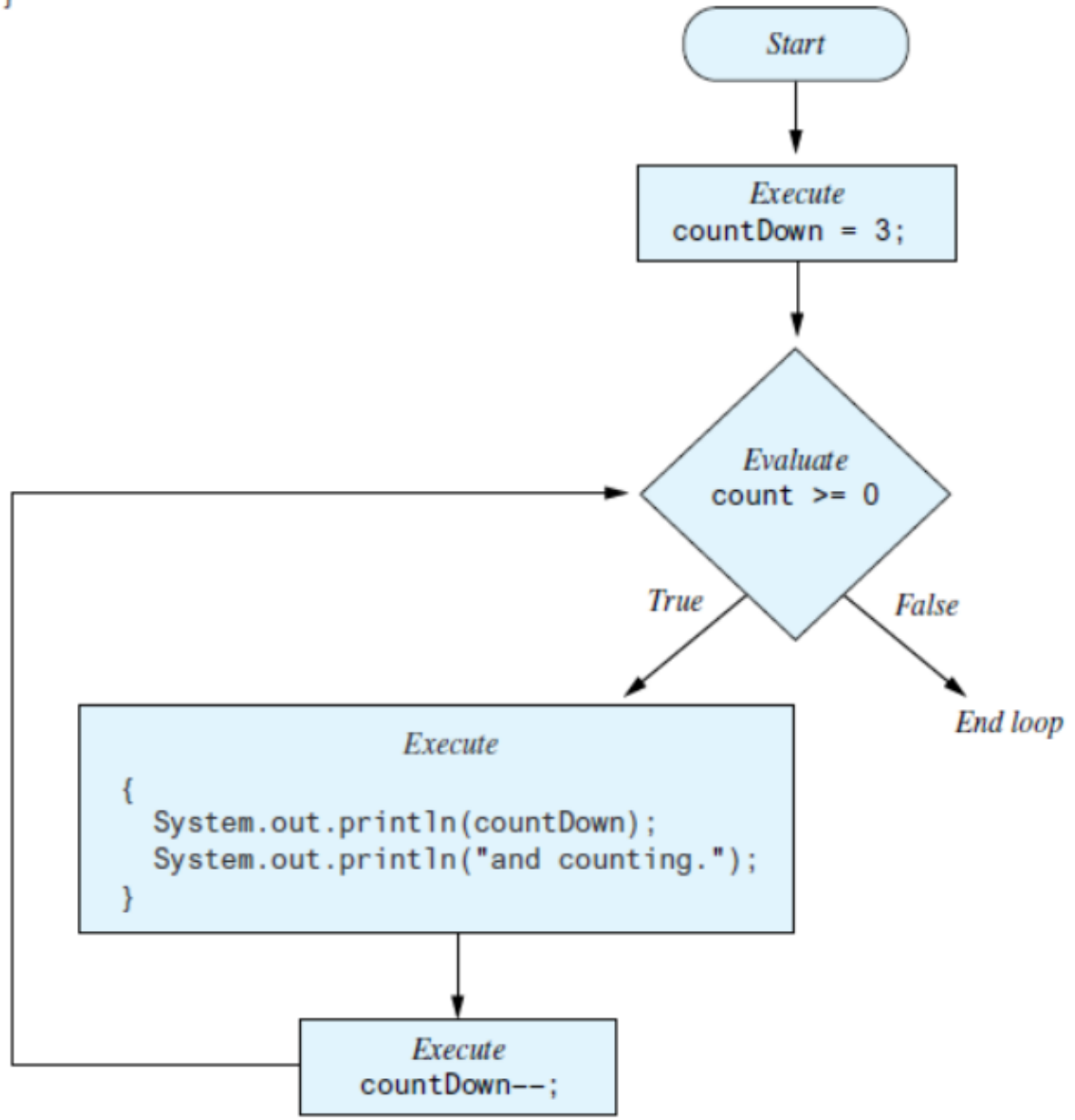
The `for` Loop Flowchart




```
for (initialization; loopContinuationCondition; increment) {
    statements
}
```

The `for` Loop

```
for (countDown = 3; countDown >= 0; countDown--)
{
    System.out.println(countDown);
    System.out.println("and counting.");
}
```



The `for` Loop: examples

a) Vary the control variable from 1 to 100 in increments of 1.

```
for (int i = 1; i <= 100; i++)
```

b) Vary the control variable from 100 to 1 in *decrements* of 1.

```
for (int i = 100; i >= 1; i--)
```

c) Vary the control variable from 7 to 77 in increments of 7.

```
for (int i = 7; i <= 77; i += 7)
```

d) Vary the control variable from 20 to 2 in *decrements* of 2.

```
for (int i = 20; i >= 2; i -= 2)
```

e) Vary the control variable over the values 2, 5, 8, 11, 14, 17, 20.

```
for (int i = 2; i <= 20; i += 3)
```

f) Vary the control variable over the values 99, 88, 77, 66, 55, 44, 33, 22, 11, 0.

```
for (int i = 99; i >= 0; i -= 11)
```

The `for` Loop

```
public class Squares
{
   public static void main(String[] args)
   {
      int number; // Loop control variable

      System.out.println("Number Number Squared");
      System.out.println("-----------------------");

      for (number = 1; number <= 10; number++)
      {
         System.out.println(number + "\t\t" +
                            number * number);
      }
   }
}
```

Program Output

```
Number      Number Squared
-----------------------------
1                   1
2                   4
3                   9
4                   16
5                   25
6                   36
7                   49
8                   64
9                   81
10                  100
```

The Sections of The `for` Loop

- The *initialization section* of the `for` loop allows the loop to initialize its own control variable.
- The *test section* of the `for` statement acts in the same manner as the condition section of a `while` loop.
- The *update section* of the `for` loop is the last thing to execute at the end of each loop.

```
public class UserSquares
{
   public static void main(String[] args)
   {
      int number;      // Loop control variable
      int maxValue;    // Maximum value to display

      System.out.println("I will display a table of " +
                         "numbers and their squares.");

      // Create a Scanner object for keyboard input.
      Scanner keyboard = new Scanner(System.in);

      // Get the maximum value to display.
      System.out.print("How high should I go? ");
      maxValue = keyboard.nextInt();

      // Display the table.
      System.out.println("Number Number Squared");
      System.out.println("-----------------------");
      for (number = 1; number <= maxValue; number++)
      {
         System.out.println(number + "\t\t" +
                            number * number);
      }
   }
}
```

Program Output with Example Input Shown in Bold

```
I will display a table of numbers and their squares.
How high should I go? 7 [Enter]
Number      Number Squared
-----------------------------------------------
1                      1
2                      4
3                      9
4                      16
5                      25
6                      36
7                      49
```

The Sections of The `for` Loop

What will the following program segments display?

a)
```
for (int count = 0; count < 6; count++)
   System.out.println(count + count);
```

b)
```
for (int value = -5; value < 5; value++)
   System.out.println(value);
```

c)
```
int x;
for (x = 5; x <= 14; x += 3)
   System.out.println(x);
System.out.println(x);
```

d)
```
int a, x = 0, y = 0;
while (x < 10)
{
   a = x * 2;
   y += a;
   x++;
}
System.out.println("The sum is " + y);
```

The `for` Loop Initialization

- The initialization section of a `for` loop is optional; however, it is usually provided.
- Typically, `for` loops initialize a counter variable that will be tested by the test section of the loop and updated by the update section.
- The initialization section can initialize multiple variables.
- Variables declared in this section have scope only for the `for` loop.

The Update Expression

- The update expression is usually used to increment or decrement the counter variable(s) declared in the initialization section of the for loop.
- The update section of the loop executes last in the loop.
- The update section may update multiple variables.
- Each variable updated is executed as if it were on a line by itself.

Modifying The Control Variable

- You should avoid updating the control variable of a `for` loop within the body of the loop.
- The update section should be used to update the control variable.
- Updating the control variable in the `for` loop body leads to hard to maintain code and difficult debugging.

Multiple Initializations and Updates

- The `for` loop may initialize and update multiple variables.

```
for(int i = 5, int j = 0; i < 10 || j < 20; i++, j+=2)
{
    statement(s);
}
```

- Note that the only parts of a `for` loop that are mandatory are the semicolons.

```
for(;;)
{
    statement(s);
} // infinite loop
```

- If left out, the test section defaults to true.

Multiple Initializations and Updates

```
int x, y;
for (x = 1, y = 1; x <= 5; x++, y++)
{
   System.out.println(x + " plus " + y +
                      " equals " + (x + y));
}
```

```
1 plus 1 equals 2
2 plus 2 equals 4
3 plus 3 equals 6
4 plus 4 equals 8
5 plus 5 equals 10
```

For loop exercises

7. What output is produced by the following code?

```
for (int n = 1; n <= 4; n++)
     System.out.println(n);
```

8. What output is produced by the following code?

```
int n;
for (n = 1; n > 4; n++)
     System.out.println(n);
```

9. What output is produced by the following code?

```
for (int n = 4; n > 0; n--)
     System.out.println(n);
```

10. What output is produced by the following code?

```
for (int n = 4; n > 0; n--);
     System.out.println(n);
```

(This is not the same as the previous question. Look carefully.)

11. What output is produced by the following code?

```
for (double test = 0; test < 3; test = test + 0.5)
    System.out.println(test);
```

12. Write a for statement that displays the even numbers 2, 4, 6, 8, and 10. Each number should appear on a separate line. Declare all the variables you use.

13. What output is produced by the following code?

```
for (int count = 0; count <= 3; count++)
    for (int count2 = 0; count2 < count; count2++)
        System.out.println(count2);
```

What do these loops print?

a. `for (int i = 1; i < 10; i++) { System.out.print(i + " "); }`

b. `for (int i = 1; i < 10; i += 2) { System.out.print(i + " "); }`

c. `for (int i = 10; i > 1; i--) { System.out.print(i + " "); }`

d. `for (int i = 0; i < 10; i++) { System.out.print(i + " "); }`

e. `for (int i = 1; i < 10; i = i * 2) { System.out.print(i + " "); }`

f. `for (int i = 1; i < 10; i++) { if (i % 2 == 0) { System.out.print(i + " "); } }`

Sentinel Values

- Sometimes the end point of input data is not known.
- A *sentinel value* can be used to notify the program to stop acquiring input.
- If it is a user input, the user could be prompted to input data that is not normally in the input data range (i.e. –1 where normal input would be positive.)
- Programs that get file input typically use the end-of-file marker to stop acquiring input data.

Sentinel Values

```
public static void main(String[] args)
{
   int points;                  // Game points
   int totalPoints = 0;         // Accumulator initialized to 0

   // Create a Scanner object for keyboard input.
   Scanner keyboard = new Scanner(System.in);

   // Display general instructions.
   System.out.println("Enter the number of points your team");
   System.out.println("has earned for each game this season.");
   System.out.println("Enter -1 when finished.");
   System.out.println();

   // Get the first number of points.
   System.out.print("Enter game points or -1 to end: ");
   points = keyboard.nextInt();
```

Sentinel Values

```
        // Accumulate the points until -1 is entered.
        while (points != -1)
        {
           // Add points to totalPoints.
           totalPoints += points;

           // Get the next number of points.
           System.out.print("Enter game points or -1 to end: ");
           points = keyboard.nextInt();
        }

        // Display the total number of points.
        System.out.println("The total points are " +
                           totalPoints);
     }
}
```

Sentinel Values

Program Output with Example Input Shown in Bold

```
Enter the number of points your team
has earned for each game this season.
Enter -1 when finished.

Enter game points or -1 to end: 7 [Enter]
Enter game points or -1 to end: 9 [Enter]
Enter game points or -1 to end: 4 [Enter]
Enter game points or -1 to end: 6 [Enter]
Enter game points or -1 to end: 8 [Enter]
Enter game points or -1 to end: -1 [Enter]
The total points are 34
```

The value −1 was chosen for the sentinel because it is not possible for a team to score negative points.

Nested Loops

- Like `if` statements, loops can be nested.
- If a loop is nested, the inner loop will execute all of its iterations for each time the outer loop executes once.

```
for(int i = 0; i < 10; i++)
    for(int j = 0; j < 10; j++)
        loop statements;
```

- The “loop statements” in this example will execute 100 times.

Nested Loops

```
public class Clock
{
   public static void main(String[] args)
   {
      // Simulate the clock.
      for (int hours = 1; hours <= 12; hours++)
      {
          for (int minutes = 0; minutes <= 59; minutes++)
          {
             for (int seconds = 0; seconds <= 59; seconds++)
              {
                 System.out.printf("%02d:%02d:%02d\n", hours, minutes, seconds);
              }
           }
        }
     }
  }
```

Program Output

```
01:00:00
01:00:01
01:00:02
01:00:03

(The loop continues to count . . . )

12:59:57
12:59:58
12:59:59
```

Nested Loops

```
public static void main(String [] args)
{
   int numStudents,     // Number of students
       numTests,        // Number of tests per student
       score,           // Test score
       total;           // Accumulator for test scores
   double average;      // Average test score

   // Create a Scanner object for keyboard input.
   Scanner keyboard = new Scanner(System.in);

   // Get the number of students.
   System.out.print("How many students do you have? ");
   numStudents = keyboard.nextInt();

   // Get the number of test scores per student.
   System.out.print("How many test scores per student? ")
   numTests = keyboard.nextInt();

   for (int student = 1; student <= numStudents; student++)
   {
      total = 0; // Set the accumulator to zero.

      // Get the test scores for a student.
      System.out.println("Student number " + student);
      System.out.println("--------------------");
      for (int test = 1; test <= numTests; test++)
      {
         System.out.print("Enter score " + test + ": ");
         score = keyboard.nextInt();
         total += score; // Add score to total.
      }

      // Calculate and display the average.
      average = total / numTests;
      System.out.printf("The average for student %d is %.1f.\n\n",
                        student, average);
   }
}
}
```

Nested Loops

Program Output with Example Input Shown in Bold

```
How many students do you have? 3 [Enter]
How many test scores per student? 3 [Enter]
Student number 1
----------------------
Enter score 1: 100 [Enter]
Enter score 2: 95 [Enter]
Enter score 3: 90 [Enter]
The average for student number 1 is 95.0.

Student number 2
----------------------
Enter score 1: 80 [Enter]
Enter score 2: 81 [Enter]
Enter score 3: 82 [Enter]
The average for student number 2 is 81.0.

Student number 3
----------------------
Enter score 1: 75 [Enter]
Enter score 2: 85 [Enter]
Enter score 3: 80 [Enter]
The average for student number 3 is 80.0.
```

Nested Loops

```
public class ExamAverager
{
    public static void main(String[] args)
    {
        System.out.println("This program computes the average of");
        System.out.println("a list of (nonnegative) exam scores.");
        double sum;
        int numberOfStudents;
        double next;
        String answer;
        Scanner keyboard = new Scanner(System.in);

        do
        {
            System.out.println();
            System.out.println("Enter all the scores to be averaged.");
            System.out.println("Enter a negative number after");
            System.out.println("you have entered all the scores.");
            sum = 0;
            numberOfStudents = 0;
            next = keyboard.nextDouble();
            while (next >= 0)
            {
               sum = sum + next;
               numberOfStudents++;
               next = keyboard.nextDouble();
            }
            if (numberOfStudents > 0)
                System.out.println("The average is " +
                                   (sum / numberOfStudents));
            else
                System.out.println("No scores to average.");

            System.out.println("Want to average another exam?");
            System.out.println("Enter yes or no.");
            answer = keyboard.next();
        } while (answer.equalsIgnoreCase("yes"));
    }
}
```

Sample Screen Output

```
This program computes the average of
a list of (nonnegative) exam scores.

Enter all the scores to be averaged.
Enter a negative number after
you have entered all the scores.
100
90
100
90
-1
The average is 95.0
Want to average another exam?
Enter yes or no.
yes

Enter all the scores to be averaged.
Enter a negative number after
you have entered all the scores. 90
70
80
-1
The average is 80.0
Want to average another exam?
Enter yes or no.
no
```

Nested Loops

Printing this: ******

```
final int COLS = 6;
for (int col = 0; col < COLS; col++)
{
   System.out.print("*");
}
```

Nested Loops

Printing this:

```
**********
**********
**********
**********
**********
```

```
public static void main(String[] args)
{
   int rows, cols;

   // Create a Scanner object for keyboard input.
   Scanner keyboard = new Scanner(System.in);

   // Get the number of rows and columns.
   System.out.print("How many rows? ");
   rows = keyboard.nextInt();
   System.out.print("How many columns? ");
   cols = keyboard.nextInt();

   for (int r = 0; r < rows; r++)
   {
      for (int c = 0; c < cols; c++)
          {
             System.out.print("*");
          }
          System.out.println();
       }
    }
 }
```

Nested Loops

Printing this:

```
*
**
***
****
*****
******
*******
********
```

```
public class TrianglePattern
{
   public static void main(String[] args)
   {
      final int BASE_SIZE = 8;

      for (int r = 0; r < BASE_SIZE; r++)
      {
         for (int c = 0; c < (r + 1); c++)
         {
            System.out.print("*");
         }
         System.out.println();
      }
   }
}
```

Nested Loops

Printing this:

```
#
 #
  #
   #
    #
     #
```

```
public class StairStepPattern
{
   public static void main(String[] args)
   {
      final int NUM_STEPS = 6;

      for (int r = 0; r < NUM_STEPS; r++)
      {
         for (int c = 0; c < r; c++)
         {
            System.out.print(" ");
         }
         System.out.println("#");
      }
   }
}
```

The `break` Statement

- The `break` statement can be used to abnormally terminate a loop.
- The use of the `break` statement in loops bypasses the normal mechanisms and makes the code hard to read and maintain.
- It is considered bad form to use the `break` statement in this manner.

```
while (itemNumber <= MAX_ITEMS)
{
    . . .
    if (itemCost <= leftToSpend)
    {
        . . .
        if (leftToSpend > 0)
            itemNumber++;
        else
        {
            System.out.println("You are out of money.");
            break;
        }
    }
    else
        . . .
}

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

The `break` Statement

```
3  public class BreakTest {
4     public static void main(String[] args) {
5        int count; // control variable also used after loop terminates
6
7        for (count = 1; count <= 10; count++) { // loop 10 times
8           if (count == 5) {
9              break; // terminates loop if count is 5
10          }
11
12          System.out.printf("%d ", count);
13       }
14
15       System.out.printf("%nBroke out of loop at count = %d%n", count);
16    }
17 }
```

```
1 2 3 4
Broke out of loop at count = 5
```

The `continue` Statement

- The `continue` statement will cause the currently executing iteration of a loop to terminate and the next iteration will begin.
- The `continue` statement will cause the evaluation of the condition in `while` and `for` loops.
- Like the `break` statement, the `continue` statement should be avoided because it makes the code hard to read and debug.

The `continue` Statement

```
3  public class ContinueTest {
4     public static void main(String[] args) {
5        for (int count = 1; count <= 10; count++) { // loop 10 times
6           if (count == 5) {
7              continue; // skip remaining code in loop body if count is 5
8           }
9
10          System.out.printf("%d ", count);
11       }
12
13       System.out.printf("%nUsed continue to skip printing 5%n");
14    }
15 }
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
```

Deciding Which Loops to Use

- The `while` loop:
  - Pretest loop
  - Use it where you do not want the statements to execute if the condition is false in the beginning.
- The `do-while` loop:
  - Post-test loop
  - Use it where you want the statements to execute at least one time.
- The `for` loop:
  - Pretest loop
  - Use it where there is some type of counting variable that can be evaluated.

Loop bugs

The two most common kinds of loop errors are

- Unintended infinite loops
- Off-by-one errors

Loop bugs

- Unintended infinite loops

Certain data can cause an infinite loop

```
count = 0;
while (balance < 0)
{
      balance = balance - penalty;
      balance = balance + deposit;
      count++;
}
System.out.println("You will have a nonnegative " +
                   "balance in " + count + " months.");
```

Code should try to guard against user error

```
if (deposit <= penalty)
    System.out.println("Deposit is too small.");
else {
   count = 0;
   while (balance < 0)
   {
       balance = balance - penalty;
       balance = balance + deposit;
       count++;
   }
   System.out.println("You will have a nonnegative " +
                      "balance in " + count + " months.");
}
```

Off-by-one errors are caused by an incorrect Boolean expression

Loop bugs

- Unintended infinite loops (**Tracing Variables**)

You can trace the variables by adding `println` statements, as follows:

```
count = 0;
System.out.println("count == " + count);                    //*
System.out.println("balance == " + balance);                //*
System.out.println("penalty == " + penalty);                //*
System.out.println("deposit == " + deposit);                //*
while (balance < 0)
{
    balance = balance + penalty;
    System.out.println("balance + penalty == " + balance);  //*
    balance = balance - deposit;
    System.out.println("balance - deposit == " + balance);  //*
    count++;
    System.out.println("count == " + count);                //*
}
System.out.println("Nonnegative balance in " + count +
                   " months.");
```

Loop bugs

- Unintended infinite loops (**Assertion Checks**)

Assert *Boolean_Expression*;

assert n >= limit;

```
assert n == 1;
while (n < limit)
{
    n = 2 * n;
}
assert n >= limit;
//n is the smallest power of 2 >= limit.
```

Exercises

Rewrite the following `for` loop into a `while` loop.

```
int s = 0;
for (int i = 1; i <= 10; i++)
{
   s = s + i;
}
```

Rewrite the following `do` loop into a `while` loop.

```
int n = in.nextInt();
double x = 0;
double s;
do
{
   s = 1.0 / (1 + n * n);
   n++;
   x = x + s;
}
while (s > 0.01);
```

Provide trace tables of the following loops.

a.
```
int s = 1;
int n = 1;
while (s < 10) { s = s + n; }
n++;
```

b.
```
int s = 1;
for (int n = 1; n < 5; n++)
{
   s = s + n;
}
```

c.
```
int s = 1;
int n = 1;
do
{
   s = s + n;
   n++;
}
while (s < 10 * n);
```

What do the following loops print? Work out the answer by tracing the code, not by using the computer.

a.
```
int s = 1;
for (int n = 1; n <= 5; n++)
{
   s = s + n;
   System.out.print(s + " ");
}
```

b.
```
int s = 1;
for (int n = 1; s <= 10; System.out.print(s + " "))
{
   n = n + 2;
   s = s + n;
}
```

c.
```
int s = 1;
int n;
for (n = 1; n <= 5; n++)
{
   s = s + n;
   n++;
}
System.out.print(s + " " + n);
```

What do the following program segments print? Find the answers by tracing the code, not by using the computer.

a.
```
int n = 1;
for (int i = 2; i < 5; i++) { n = n + i; }
System.out.print(n);
```

b.
```
int i;
double n = 1 / 2;
for (i = 2; i <= 5; i++) { n = n + 1.0 / i; }
System.out.print(i);
```

c.
```
double x = 1;
double y = 1;
int i = 0;
do
{
   y = y / 2;
   x = x + y;
   i++;
}
while (x < 1.8);
System.out.print(i);
```

d.
```
double x = 1;
double y = 1;
int i = 0;
while (y >= 1.5)
{
   x = x / 2;
   y = x + y;
   i++;
}
System.out.print(i);
```

Generating Random numbers with the Random Class

- Random numbers are commonly used in games. For example, computer games that let the player roll dice use random numbers to represent the values of the dice. Programs that show cards being drawn from a shuffled deck use random numbers to represent the face values of the cards.
- Random numbers are useful in simulation programs. In some simulations, the computer must randomly decide how a person, animal, insect, or other living being will behave. Formulas can be constructed in which a random number is used to determine various actions and events that take place in the program.
- Random numbers are useful in statistical programs that must randomly select data for analysis.
- Random numbers are commonly used in computer security to encrypt sensitive data.

Generating Random numbers with the Random Class

The class is part of the `java.util` package, so any program that uses it will need an `import` statement such as:

```
import java.util.Random;
```

You create an object from the `Random` class with a statement such as this:

```
Random randomNumbers = new Random();
```

Once you have created a `Random` object, you can call its `nextInt` method to get a random integer number. The following code shows an example:

```
// Declare an int variable.
int number;

// Create a Random object.
Random randomNumbers = new Random();

// Get a random integer and assign it to number.
number = randomNumbers.nextInt();
number = randomNumbers.nextInt(100);
number = randomNumbers.nextInt(10) + 1;
```

Generating Random numbers with the Random Class

Method	Description
`nextDouble()`	Returns the next random number as a `double`. The number will be within the range of `0.0` through `1.0`.
`nextFloat()`	Returns the next random number as a `float`. The number will be within the range of `0.0` through `1.0`.
`nextInt()`	Returns the next random number as an `int`. The number will be within the range of an `int`, which is –2,147,483,648 to +2,147,483,648.
`nextInt(int n)`	This method accepts an integer argument, *n*. It returns a random number as an `int`. The number will be within the range of 0 through *n*.
`nextLong()`	Returns the next random number as a `long`. The number will be within the range of a `long`, which is –9,223,372,036,854,775,808 to +9,223,372,036,854,775,808.

Example: `MathTutor`

```
import java.util.Scanner;   // Needed for the Scanner class
import java.util.Random;    // Needed for the Random class

 public static void main(String[] args)
 {
    int number1;         // A number
    int number2;         // Another number
    int sum;             // The sum of the numbers
    int userAnswer;      // The user's answer

    // Create a Scanner object for keyboard input.
    Scanner keyboard = new Scanner(System.in);

    // Create a Random class object.
    Random randomNumbers = new Random();

    // Get two random numbers.
    number1 = randomNumbers.nextInt(100);
    number2 = randomNumbers.nextInt(100);

    // Display an addition problem.
    System.out.println("What is the answer to the " +
                       "following problem?");
    System.out.print(number1 + " + " +
                     number2 + " = ? ");

    // Calculate the answer.
    sum = number1 + number2;

    // Get the user's answer.
    userAnswer = keyboard.nextInt();

    // Display the user's results.
    if (userAnswer == sum)
       System.out.println("Correct!");
    else
    {
       System.out.println("Sorry, wrong answer. " +
                          "The correct answer is " +
                          sum);
    }
 }
}
```

Generating Random numbers with the Random Class

```
import java.util.Scanner;    // Needed for the Scanner class
import java.util.Random;     // Needed for the Random class

 public static
 {
    int number1
    int number2
    int sum;
    int userAns

    // Create a
    Scanner key

    // Create a
    Random rand

    // Get two random numbers.
    number1 = randomNumbers.nextInt(100);
    number2 = randomNumbers.nextInt(100);

    // Display an addition problem.
    System.out.println("What is the answer to the " +
                       "following problem?");
```

Program Output with Example Input Shown in Bold

```
What is the answer to the following problem?
52 + 19 = ? 71 [Enter]
Correct!
```

Program Output with Example Input Shown in Bold

```
What is the answer to the following problem?
27 + 73 = ? 101 [Enter]
Sorry, wrong answer. The correct answer is 100
```

```
                                    System.out.print(number1 + " + " +
                                                     "? ");
                                 }
                              }
                              else
                              {
                                 System.out.println("Sorry, wrong answer. " +
                                                    "The correct answer is " +
                                                    sum);
                              }
                           }
```

Example 2:

RollDice

```
import java.util.Scanner;
import java.util.Random;

/**
   This program simulates the rolling of dice.
*/

public class RollDice
{
   public static void main(String[] args)
   {
      String again = "y";   // To control the loop
      int die1;             // To hold the value of die #1
      int die2;             // to hold the value of die #2

      // Create a Scanner object to read keyboard input.
      Scanner keyboard = new Scanner(System.in);

      // Create a Random object to generate random numbers.
      Random rand = new Random();
```

Example 2:

```
        while (again.equalsIgnoreCase("y"))
        {
            System.out.println("Rolling the dice ...");
            die1 = rand.nextInt(6) + 1;
            die2 = rand.nextInt(6) + 1;
            System.out.println("Their values are:");
            System.out.println(die1 + " " + die2);

            System.out.print("Roll them again (y = yes)? ");
            again = keyboard.nextLine();
        }
    }
}
```

Example 2:

Program Output with Example Input Shown in Bold

```
Rolling the dice ...
Their values are:
4 3
Roll them again (y = yes)? y [Enter]
Rolling the dice ...
Their values are:
2 6
Roll them again (y = yes)? y [Enter]
Rolling the dice ...
Their values are:
1 5
Roll them again (y = yes)? n [Enter]
```

Example 3:

CoinToss

```
public class CoinToss
{
   public static void main(String[] args)
   {
      // Create a Random object to generate random numbers.
      Random rand = new Random();

      // Simulate the coin tosses.
      for (int count = 0; count < 10; count++)
      {
         if (rand.nextInt(2) == 0)
            System.out.println("Tails");
         else
            System.out.println("Heads");
      }
   }
}
```

Program Output

```
Tails
Tails
Heads
Tails
Heads
Heads
Heads
Tails
Heads
Tails
```

Method 2: using the Math class

```
public class RandomDemo
{
   public static void main(String[] args)
   {
      for (int i = 1; i <= 10; i++)
      {
         double r = Math.random();
         System.out.println(r);

      }
   }
}
```

```
0.6513550469421886
0.920193662882893
0.6904776061289993
0.8862828776788884
0.7730177555323139
0.3020238718668635
0.0028504531690907164
0.9099983981705169
0.1151636530517488
0.1592258808929058
```

Method 2: using the Math class (Dice)

```
public class Dice
{
   public static void main(String[] args)
   {
      for (int i = 1; i <= 10; i++)
      {
         // Generate two random numbers between 1 and 6

         int d1 = (int) (Math.random() * 6) + 1;
         int d2 = (int) (Math.random() * 6) + 1;
         System.out.println(d1 + " " + d2);
      }
      System.out.println();
   }
}
```

Program Run

```
5 1
2 1
1 2
5 1
1 2
6 4
4 4
6 1
6 3
5 2
```

Thank you