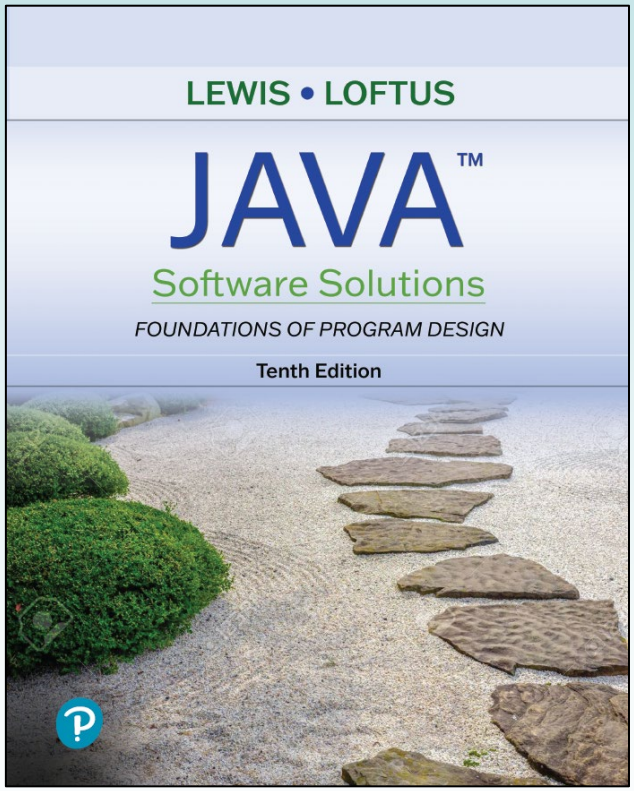
Chapter 5 A
Conditionals and Loops



Java Software Solutions
Foundations of Program Design
10th Edition

Edits: 03/17/2023

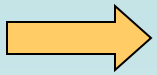
8/12/2026

John Lewis
William Loftus

Conditionals and Loops

- Statements that allow us to:
  - make decisions
  - repeat processing steps in a loop
- Chapter 5 focuses on:
  - boolean expressions
  - the if and if-else statements
  - comparing data
  - while loops
  - iterators
  - the `ArrayList` class
  - more GUI controls

Outline



Boolean Expressions

The `if` Statement

Comparing Data

The `while` Statement

- **Iterators**

The `ArrayList` Class

- **Determining Event Sources**
- **Managing Fonts**
- **Check Boxes and Radio Buttons**

Flow of Control

- The order of statement execution is called the *flow of control*
  - Like a finger that points at statements one-by-one as they are executed
- Unless specified otherwise, the flow of control through a method is sequential: one statement after another

Flow of Control

```
public class Demo
{
    // object is accessed using reference
   variable phrase
    public static void main(String[] args)
    {
        String phrase = "Change is inevitable";
        System.out.println("Original string:" +
                        phrase );

        // Change what phrase points to
        phrase = "Things stay the same";
        System.out.println("New string:" + phrase
    );
      }
}
```

Flow of Control

- Some programming statements make decisions or perform repetitions, altering the flow of control
- These decisions are based on *boolean expressions* that evaluate to true or false

§ 5.1 Boolean Expressions

- Java's *relational operators* return boolean (true/false) results:

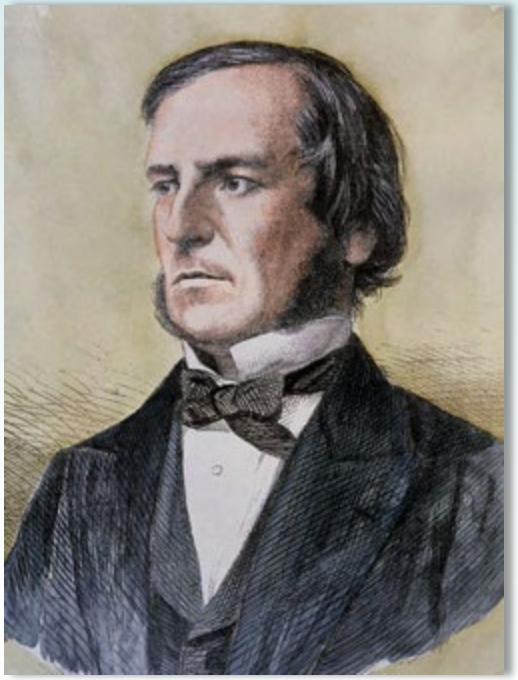
`==`	equal to
`!=`	not equal to
`<`	less than
`>`	greater than
`<=`	less than or equal to
`>=`	greater than or equal to

- Note the difference between the equality operator (==) and the assignment operator (=)
- These work with numeric data (not Strings.)


Effort > Fun.

Boolean Expressions

- **George Boole** (1815 – 1864) was an English mathematician, educator, philosopher and logician.
- He is best known as the author of *The Laws of Thought* (1854) which described Boolean algebra.



Boolean Expressions

- An **`if`** statement with its boolean condition:

```
if ( sum > MAX )
   delta = sum - MAX;
```

- First, the condition is evaluated: the value of `sum` is either greater than the value of `MAX`, or it is not
- If the condition is true, the assignment statement is executed; if it isn't, it is skipped
- See `Age.java`

```
//********************************************************************
//  Age.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an if statement.
//********************************************************************
import java.util.Scanner;

public class Age
{
   //-----------------------------------------------------------------
   //  Reads the user's age and prints comments accordingly.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final int MINOR = 21;

      Scanner scan = new Scanner(System.in);

      System.out.print("Enter your age: ");
      int age = scan.nextInt();
      System.out.println("You entered: " + age);

      if (age < MINOR)
         System.out.println("Youth is a wonderful thing. Enjoy.");

      System.out.println("Age is a state of mind.");
   }
}
```

Sample Run

```
Enter your age: 47
You entered: 47
Age is a state of mind.
```

```
        System.out.println("You entered: " + age);

        if (age < MINOR)
           System.out.println("Youth is a wonderful thing. Enjoy.");

        System.out.println("Age is a state of mind.");
    }
}
```

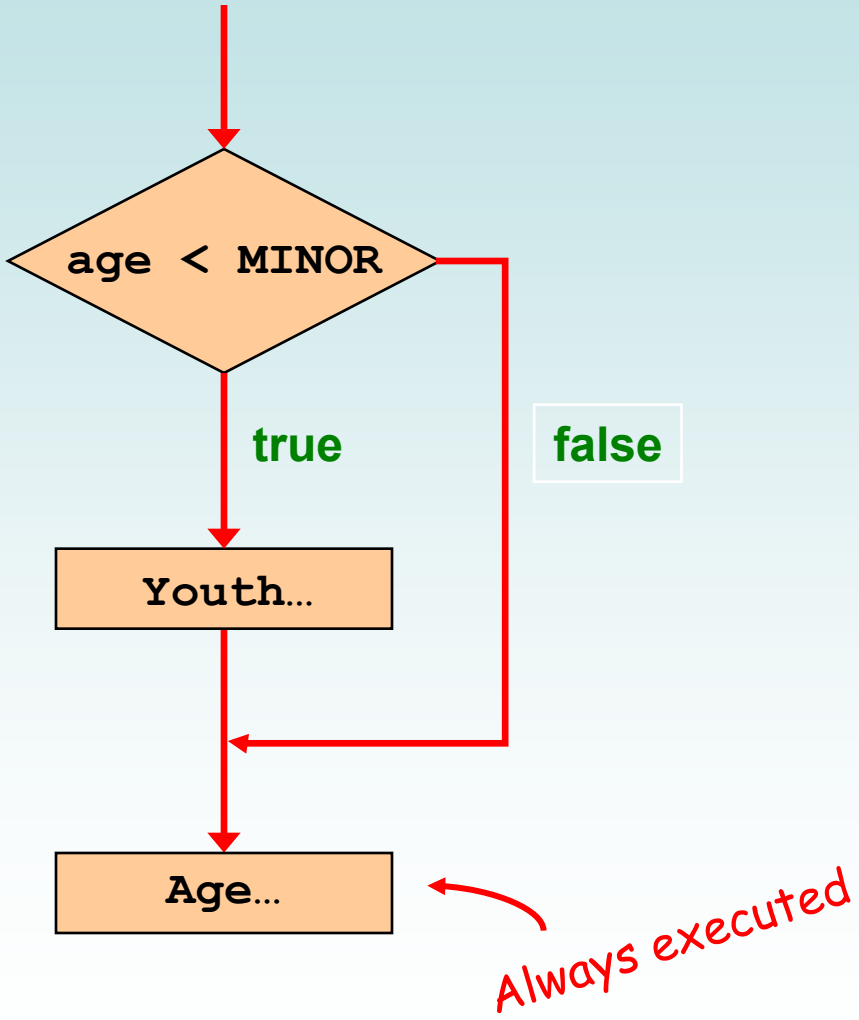
executed if condition is true

Always executed

Another Sample Run

```
Enter your age: 12
You entered: 12
Youth is a wonderful thing. Enjoy.
Age is a state of mind.
```

Logic of an if statement



Boolean Variables

- A variable is a small section of memory that contains a value
  - We have seen variables that hold primitives int, double, and others (see chapter 2)
  - boolean is a primitive type, and a variable can hold a boolean value
  - The values are true or false (only)

```
boolean bval = false;
boolean decision = true;
```

Logical Operators

- Boolean expressions can also use the following *logical operators*:

`!`	Logical NOT
`&&`	Logical AND
`||`	Logical OR

- They all take boolean operands and produce boolean results
- Logical NOT is a unary operator (it operates on one operand)
- Logical AND and logical OR are binary operators (each operates on two operands)

Logical NOT

- The *logical NOT* operation is also called *logical negation* or *logical complement*
- If some boolean condition `a` is true, then `!a` is false; if `a` is false, then `!a` is true
- Logical expressions can be shown using a *truth table*:

`a`	`!a`
true	false
false	true

Logical AND and Logical OR

- The *logical AND* expression

```
a && b
```

  is true if both `a` and `b` are true and false otherwise

- The *logical OR* expression

```
a || b
```

  is true if either `a` or `b` or both are true, and false otherwise

Logical AND and Logical OR

- A truth table shows all possible true-false combinations of the terms
- Since `&&` and `||` each have two operands, there are four possible combinations of conditions `a` and `b`

`a`	`b`	`a && b`	`a || b`
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Outline

- Boolean Expressions
- The `if` Statement
- Comparing Data
- The `while` Statement
  - ☺ Iterators
- The `ArrayList` Class
  - ☺ Determining Event Sources
  - ☺ Managing Fonts
  - ☺ Check Boxes and Radio Buttons

§ 5.2 The `if` Statement

- The *if statement* has the following syntax:

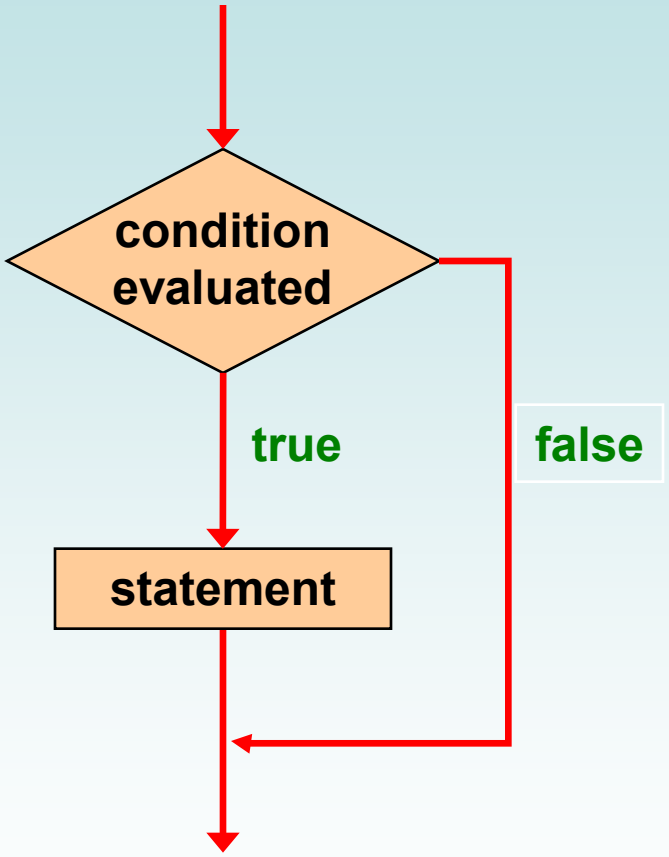
`if` is a Java reserved word

The *`condition`* must be a boolean expression. It must evaluate to either true or false.

```
if ( condition )
   statement;
```

If the *`condition`* is true, the *`statement`* is executed. If it is false, the *`statement`* is skipped.

Logic of an if statement



Indentation

- The statement controlled by the `if` statement is indented to indicate that relationship
- The use of a consistent indentation style makes a program easier to read and understand
- The compiler ignores indentation, which can lead to errors if the indentation is misleading

```
public class Picnic
{
   //-----------------------------------------------------------------
   //  Determines if conditions are right for a  picnic
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean sunny=true, warm = true;

      if ( sunny && warm )
         System.out.println("Time for a picnic!");
   }
}
```

Program prints "Time for a picnic!" , since the condition is true.

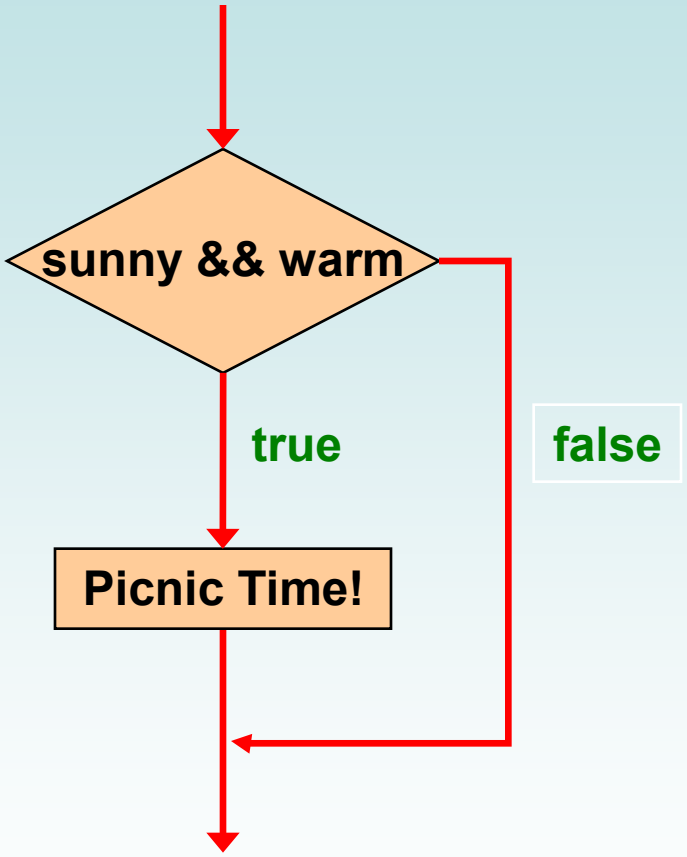
true && true == true

```
public class Picnic
{
   //-----------------------------------------------------------------
   //  Determines if conditions are right for a  picnic
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean sunny=true, warm = false;

      if ( sunny && warm )
         System.out.println("Time for a picnic!");
   }
}
```

Program prints nothing, since the condition is false.

true && false == false

Logic of an if statement



```
public class Picnic
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for a  picnic
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int percentClouds=30, temp = 76;
      boolean sunny, warm;

      sunny = (percentClouds < 50); // computes true and assigns to sunny
      warm  = (temp > 75); // computes true and assigns to warm

      if ( sunny && warm )
         System.out.println("Time for a picnic!");
   }
}
```

Program prints “Time for a picnic”.

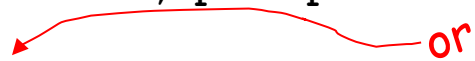
true && true == true

```
public class Picnic
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for a  picnic
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int percentClouds=30, temp = 76;

      if ((percentClouds<50) && (temp>75) )
         System.out.println("Time for a picnic!");
   }
}
```

Program prints “Time for a picnic”.

true && true == true

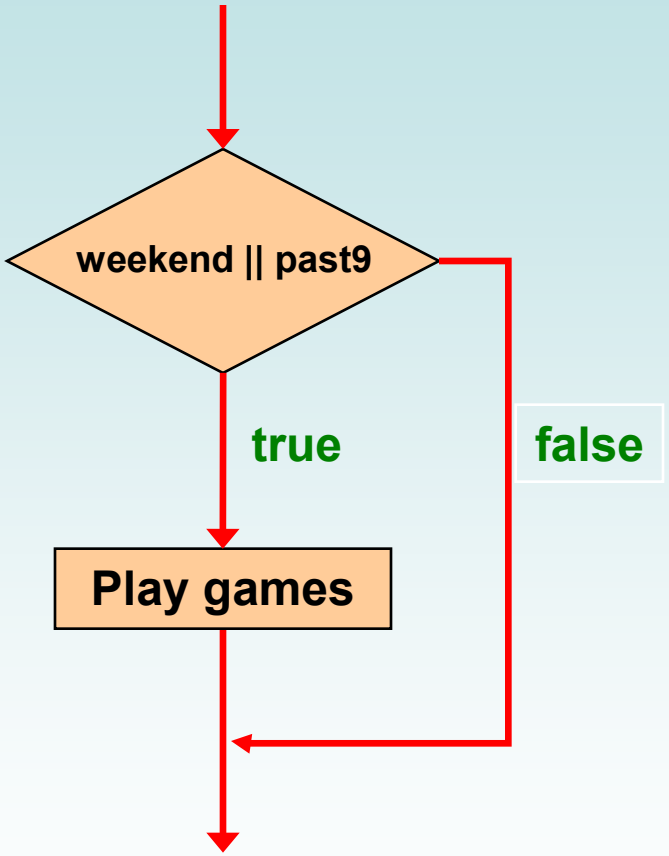
```
public class GameTime
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for playing games.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean weekend=true, past9pm = false;

      if ( weekend || past9pm)
         System.out.println("Play Games!");
   }
}
```



Program prints “Play Games!”, since the condition is true.

true || false == true

Logic of an if statement



```
public class GameTime
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for playing games.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean weekend=false, past9pm = true;

      if ( weekend || past9pm)
         System.out.println("Play Games!");
   }
}
```

Program prints “Play Games!”, since the condition is true.

false || true == true

```
public class GameTime
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for playing games.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean weekend=false, past9pm = false;

      if ( weekend || past9pm)
         System.out.println("Play Games!");
   }
}
```

Program prints nothing, since the condition is false.

false || false == false

```
public class GameTime
{
   //-----------------------------------------------------------------
   // Determines if conditions are right for playing games.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      boolean weekend=true, past9pm = true;

      if ( weekend || past9pm)
         System.out.println("Play Games!");
   }
}
```

Program prints “Play Games, since the condition is true.

true || true == true

Logical Operators

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
   System.out.println("Processing…");
```

- Logical operators have lower precedence than the relational operators

  ! Has High precedence

  math has next higher precedence

  < has higher precedence than &&

  see Appendix D

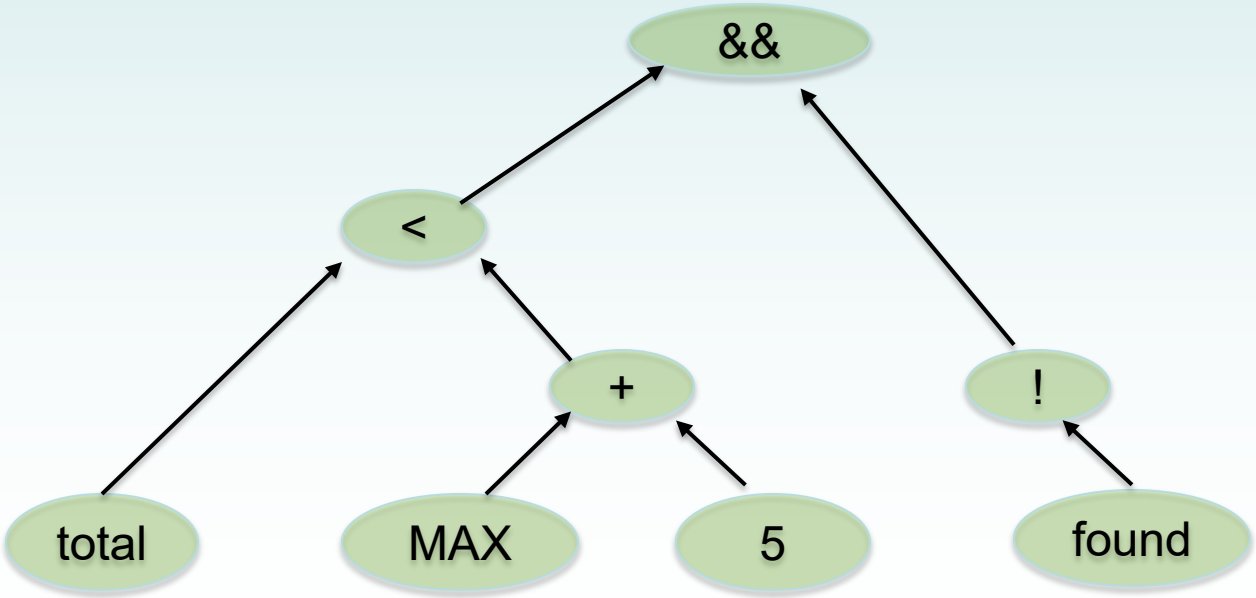
Operator's Precedence in Java

Operators	Precedence
!, +, - (unary Operators)	First (Highest)
* , / , %	Second
+ , -	Third
< , <= , >=, >	Fourth
== , !=	Fifth
&&	Sixth
||	Seventh
= (assignment Operator)	Lowest

Expression Tree

- Expressions that use logical operators can form complex conditions

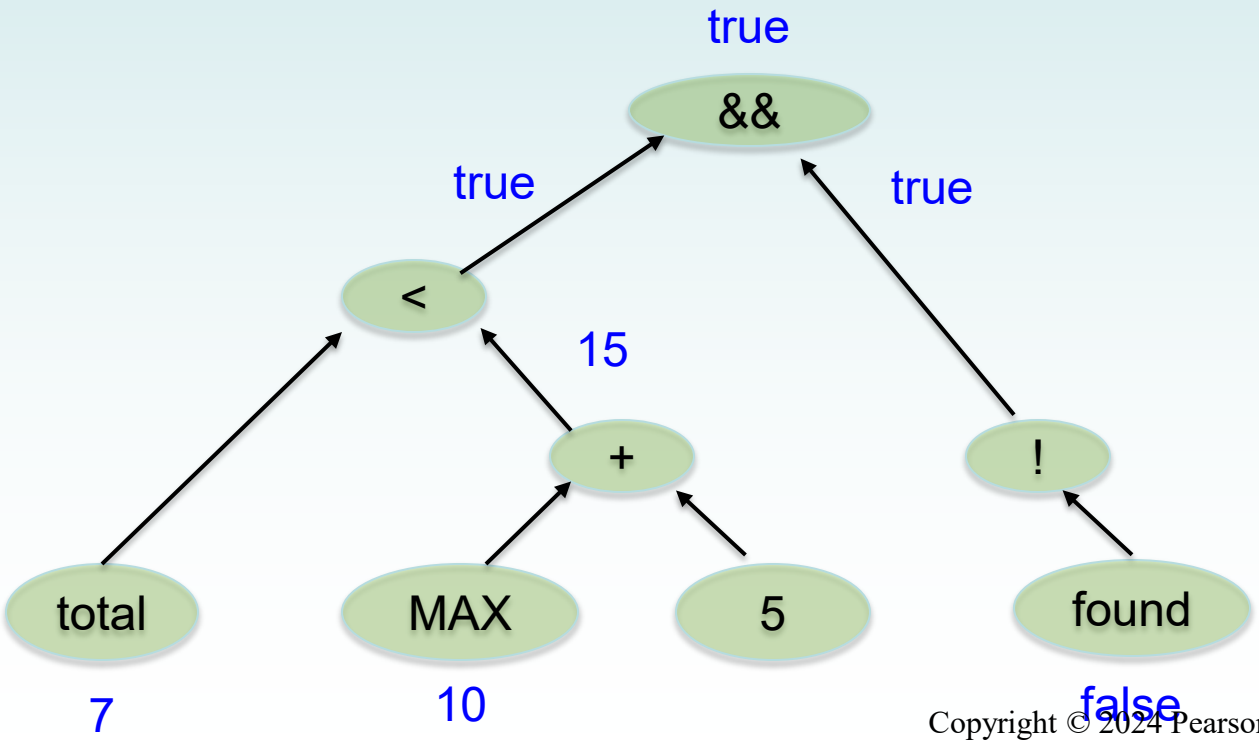
```
if (total < MAX+5 && !found)
   System.out.println("Processing…");
```



Expression Tree

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
   System.out.println("Processing…");
```



Boolean Expressions

evaluate expressions with truth tables

`total < MAX+5`	`found`	`!found`	`total < MAX+5 && !found`
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

Parentheses

- This…..

```
if (total < MAX+5 && !found)
   System.out.println("Processing…");
```

- …..works as if it were written

```
if ( (total < (MAX+5)) && (!found) )
   System.out.println("Processing…");
```

Short-Circuited Operators

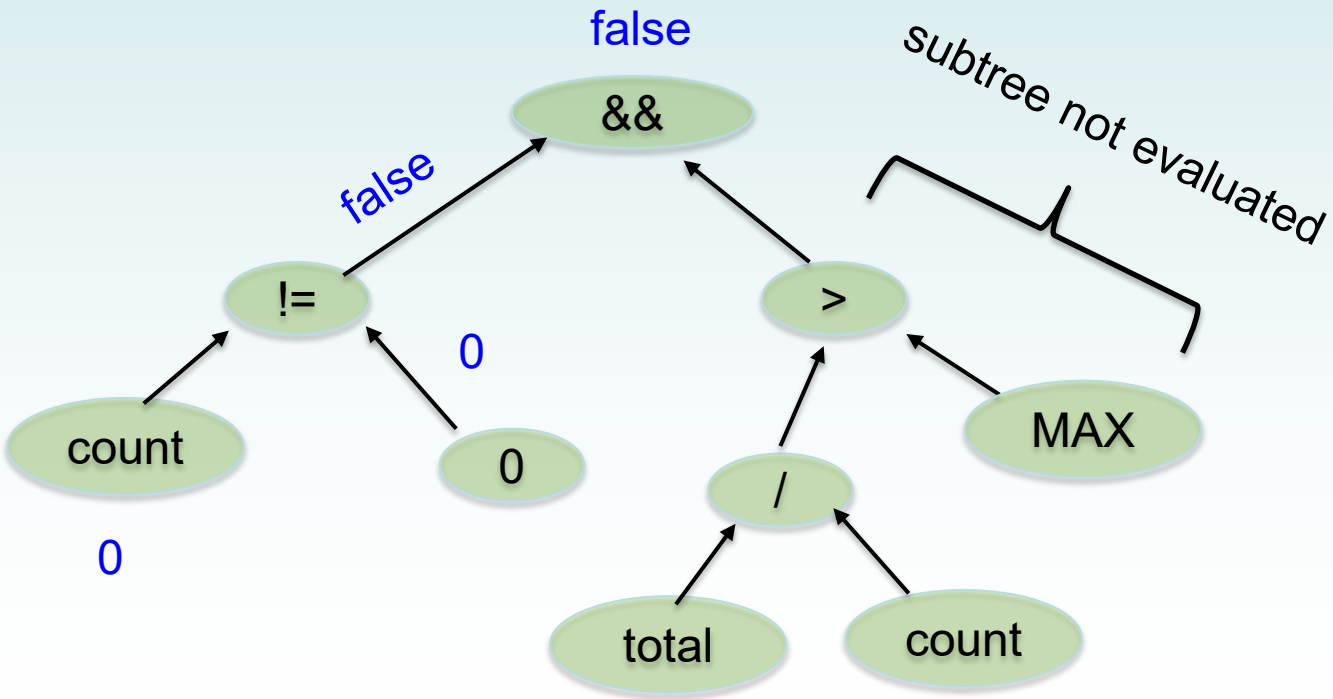
- `&&` and || are "short-circuited"
- If the left operand is sufficient to determine the result, the right operand is not evaluated

```
if (count != 0 && total/count > MAX)
   System.out.println("Testing.");
```

  - Say count is 0. Then the right of && is not done.
- This type of processing should be used carefully
  - Beware of side effects in skipped sub-expressions

Expression Tree

```
if (count != 0 && total/count > MAX)
   System.out.println("Testing");
```



Quick Check

What do the following statements do?

```
int stock = 5, warehouse = 10, total = 13;
boolean inventoryError;

if (total != stock + warehouse)
   inventoryError = true;
```

```
boolean found = false, done = true.
if (found || !done)
   System.out.println("Ok");
```

Quick Check

What do the following statements do?

```
int stock = 5, warehouse = 10, total = 13;
boolean inventoryError;
```

```
      13     !=     5     +     10
if (total != stock + warehouse)
   inventoryError = true;
```

set inventoryError to true

```
boolean found = false, done = true.
if (found || !done)
   System.out.println("Ok");
```

nothing printed

Block Statements

- Several statements can be grouped together into a *block statement* delimited by braces
- A block statement can be used wherever a statement is called for in Java syntax

```
if (total > MAX)
{
   System.out.println("Error!!");
   errorCount++;
}
```

The if-else Statement

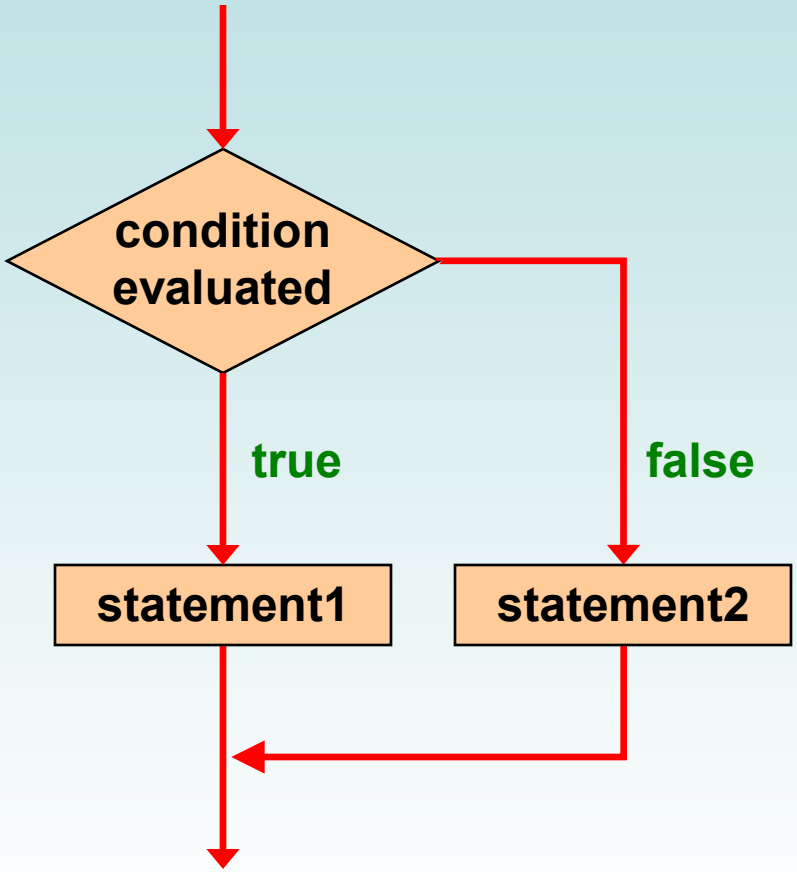
- An *else clause* can be added to an `if` statement to make an *if-else statement*

```
if ( condition )
    statement1;
else
    statement2;
```

Note semicolons

- If the *condition* is true, *statement1* is executed; if the *condition* is false, *statement2* is executed
- One or the other will be executed, but not both
- See `Wages.java`

Logic of an if-else statement



```
//********************************************************************
//  Wages.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an if-else statement.
//********************************************************************
import java.util.Scanner;

public class Wages
{
   //-----------------------------------------------------------------
   //  Reads the number of hours worked and calculates wages.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final double RATE = 8.25;  // regular pay rate
      final int STANDARD = 40;   // standard hours in a work week

      Scanner scan = new Scanner(System.in);

      double pay = 0.0;
```

continue

```
        System.out.print ("Enter the number of hours worked: ");
        int hours = scan.nextInt();

        System.out.println ();

        // Pay overtime at "time and a half"
        if (hours > STANDARD)
           pay = STANDARD * RATE + (hours-STANDARD) * (RATE * 1.5);
        else
           pay = hours * RATE;

        System.out.println("Gross earnings: " + pay );
    }
}
```

```
Enter the number of hours worked: 46

Gross earnings: 404.25
```

```
Enter the number of hours worked: 30

Gross earnings: 247.5
```

```
import java.util.Scanner;

public class Wages
{
   //-----------------------------------------------------------------
   //  Reads the number of hours worked and calculates wages.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final double RATE = 8.25;  // regular pay rate
      final int STANDARD = 40;   // standard hours in a work week

      Scanner scan = new Scanner(System.in);

      double pay = 0.0;
      System.out.print ("Enter the number of hours worked: ");
      int hours = scan.nextInt();

      System.out.println ();

      // Pay overtime at "time and a half"
      if (hours > STANDARD)
         pay = STANDARD * RATE + (hours-STANDARD) * (RATE * 1.5);
      else
         pay = hours * RATE;

      System.out.println("Gross earnings: " + pay );
   }
}
```

Indentation Revisited

- Indentation is for the human reader and is ignored by the compiler

```
if (depth >= UPPER_LIMIT)
   delta = 100;
else
   System.out.println("Reseting Delta");
   delta = 0;
```

- Despite what the indentation implies, `delta` will be set to 0 no matter what

The Coin Class

- A class that represents a coin that can be flipped
- Instance data indicates which face (heads or tails) is showing
- See `CoinFlip.java`
- See `Coin.java`

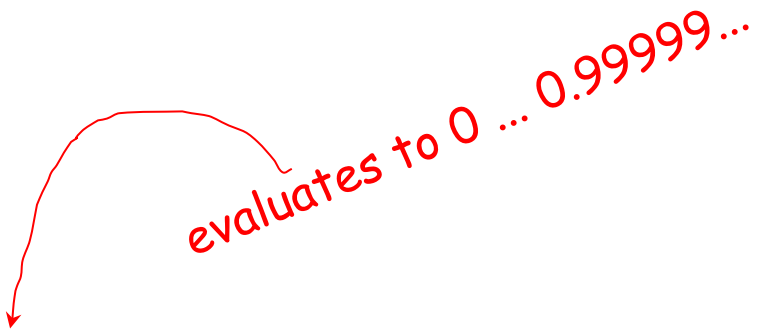
```
//********************************************************************
//  CoinFlip.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an if-else statement.
//********************************************************************

public class CoinFlip
{
   //-----------------------------------------------------------------
   //  Creates a Coin object, flips it, and prints the results.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      Coin myCoin = new Coin();

      myCoin.flip();

      System.out.println(myCoin);

      if ( myCoin.isHeads() )
         System.out.println("You win.");
      else
         System.out.println("Better luck next time.");
   }
}
```

Sample Run

```
Tails
Better luck next time.
```

```
//********************                                  **********************
//  CoinFlip.java
//
//  Demonstrates the                                    .
//********************                                  **********************

public class CoinFlip  //  in CoinFlip.java
{
   //-----------------------------------------------------------------
   //  Creates a Coin object, flips it, and prints the results.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      Coin myCoin = new Coin();

      myCoin.flip();

      System.out.println(myCoin);

      if (myCoin.isHeads())
         System.out.println("You win.");
      else
         System.out.println("Better luck next time.");
   }
}
```

```
public class Coin   //  in Coin.java
{
   private final int HEADS = 0;
   private final int TAILS = 1;

   private int face;

   //-----------------------------------------------------------------
   //  Sets up the coin by flipping it initially.
   //-----------------------------------------------------------------
   public Coin()
   {
      flip();
   }

   public void flip()
   {
      face = (int) (Math.random() * 2);
   }

   //-----------------------------------------------------------------
   //  Returns true if the current face of the coin is heads.
   //-----------------------------------------------------------------
   public boolean isHeads()
   {
      return (face == HEADS);
   }
```



```
    //-----------------------------------------------------------------
    //  Returns the current face of the coin as a string.
    //-----------------------------------------------------------------
    public String toString()
    {
        String faceName;

        if (face == HEADS)
            faceName = "Heads";
        else
            faceName = "Tails";

        return faceName;
    }
}
```

```
//********************************************************************
//  CoinFlip.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an if-else statement.
//********************************************************************

public class CoinFlip  //  in CoinFlip.java
{
   //-----------------------------------------------------------------
   //  Creates a Coin object, flips it, and prints the results.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      Coin myCoin = new Coin();

      myCoin.flip();

      System.out.println(myCoin);

      if (myCoin.isHeads())
         System.out.println("You win.");
      else
         System.out.println("Better luck next time.");
   }
}
```

```
public class Coin   //  in Coin.java
{
   private final int HEADS = 0;
   private final int TAILS = 1;

   private int face;

   public Coin()
   {
      flip();
   }

   public void flip()
   {
      face = (int) (Math.random() * 2);
   }

   public boolean isHeads()
   {
      return (face == HEADS);
   }

   public String toString()
   {
      String faceName;

      if (face == HEADS)
         faceName = "Heads";
      else
         faceName = "Tails";

      return faceName;
   }
}
```

Block Statements (again)

- Several statements can be grouped together into a *block statement* delimited by braces
- A block statement can be used wherever a statement is called for in the Java syntax rules

```
if (total > MAX)
{
   System.out.println("Error!!");
   errorCount++;
}
```

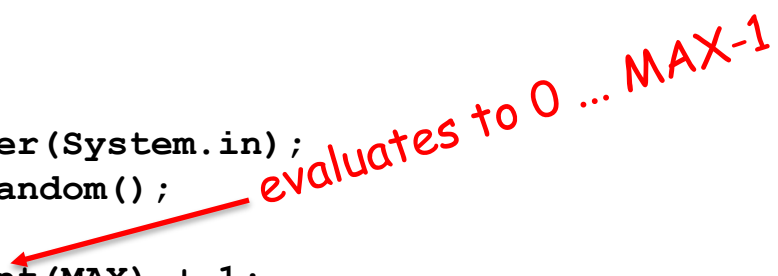
Block Statements

- The `if` clause, or the `else` clause, or both, could have block statements

```
if (total > MAX)
{
   System.out.println("Error!!");
   errorCount++;
}
else
{
   System.out.println("Total: " + total);
   current = total*2;
}
```

- See `Guessing.java`

```
//********************************************************************
//  Guessing.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a block statement in an if-else.
//********************************************************************

import java.util.*;

public class Guessing
{
   //-----------------------------------------------------------------
   //  Plays a simple guessing game with the user.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final int MAX = 10;
      int answer, guess;

      Scanner scan = new Scanner(System.in);
      Random generator = new Random();

      answer = generator.nextInt(MAX) + 1;
```



continue

```
      System.out.print ("I'm thinking of a number between 1 and "
                        + MAX + ". Guess what it is: ");

      guess = scan.nextInt();

      if (guess == answer)
         System.out.println("You got it! Good guessing!");
      else
      {
         System.out.println("That is not correct, sorry.");
         System.out.println("The number was " + answer);
      }
   }
}
```

Sample Run

```
I'm thinking of a number between 1 and 10. Guess what it is: 6
That is not correct, sorry.
The number was 9
```

```
import java.util.*;

public class Guessing
{
   public static void main(String[] args)
   {
      final int MAX = 10;
      int answer, guess;

      Scanner scan = new Scanner(System.in);
      Random generator = new Random();

      answer = generator.nextInt(MAX) + 1;
      System.out.print ("I'm thinking of a number between 1 and "
                        + MAX + ". Guess what it is: ");

      guess = scan.nextInt();

      if (guess == answer)
         System.out.println("You got it! Good guessing!");
      else
      {
         System.out.println("That is not correct, sorry.");
         System.out.println("The number was " + answer);
      }
   }
}
```

Nested `if` Statements

- The statement that follows an `if` or `else` clause could be another `if` statement
- These are called *nested if statements*
  - Two-way decisions (if statements) are used to build many-way decisions
  - Like the game "20 questions"
- There can be `if` statements nested within `if` statements nested within `if` statements….

Nested if Statements

- An `else` clause matches the last previous unmatched `if` (no matter what the indentation implies)
  - Scan a program top to bottom
  - When you find an `else` scan upward to find an unmatched `if`
    - If you don't find one, syntax is incorrect
  - Match the `else` with the `if` and continue scanning

```
import java.util.Scanner;
public class MinOfThree
{
   public static void main(String[] args)
   {
      int num1, num2, num3, min = 0;

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter three integers: ");
      num1 = scan.nextInt();
      num2 = scan.nextInt();
      num3 = scan.nextInt();

      if (num1 < num2)
      {
         if (num1 < num3)
            min = num1;
         else
            min = num3;
      }
      else
         if (num2 < num3)
            min = num2;
         else
            min = num3;

      System.out.println("Minimum value: " + min);
   }
}
```

Sample Run

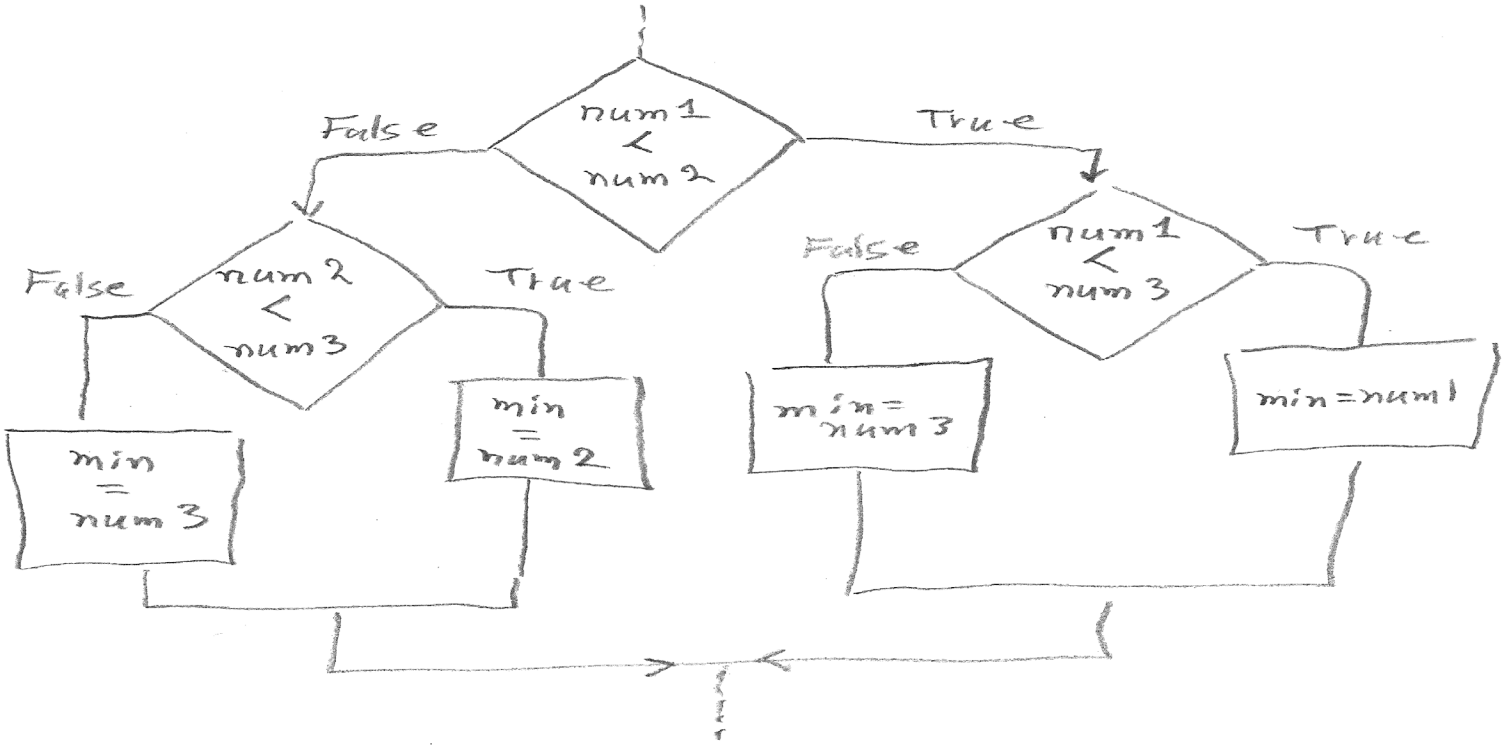
```
Enter three integers:
12  69  24
Minimum value: 12
```

Sample Run

```
Enter three integers:
12  69  9
Minimum value: 9
```

Sample Run

```
Enter three integers:
84  69  90
Minimum value: 69
```



```
import java.util.Scanner;
public class Grade
{
   public static void main(String[] args)
   {
      int score;

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter score: ");
      score = scan.nextInt();

      if (score>=95)
        System.out.println("Grade: A");
      else
         if (score>=90)
            System.out.println("Grade: B");
         else
            if (score>=80)
              System.out.println("Grade: C");
            else
              System.out.println("More Study Needed");
   }
}
```

Sample Run

```
Enter score:
98
Grade: A
```

Sample Run

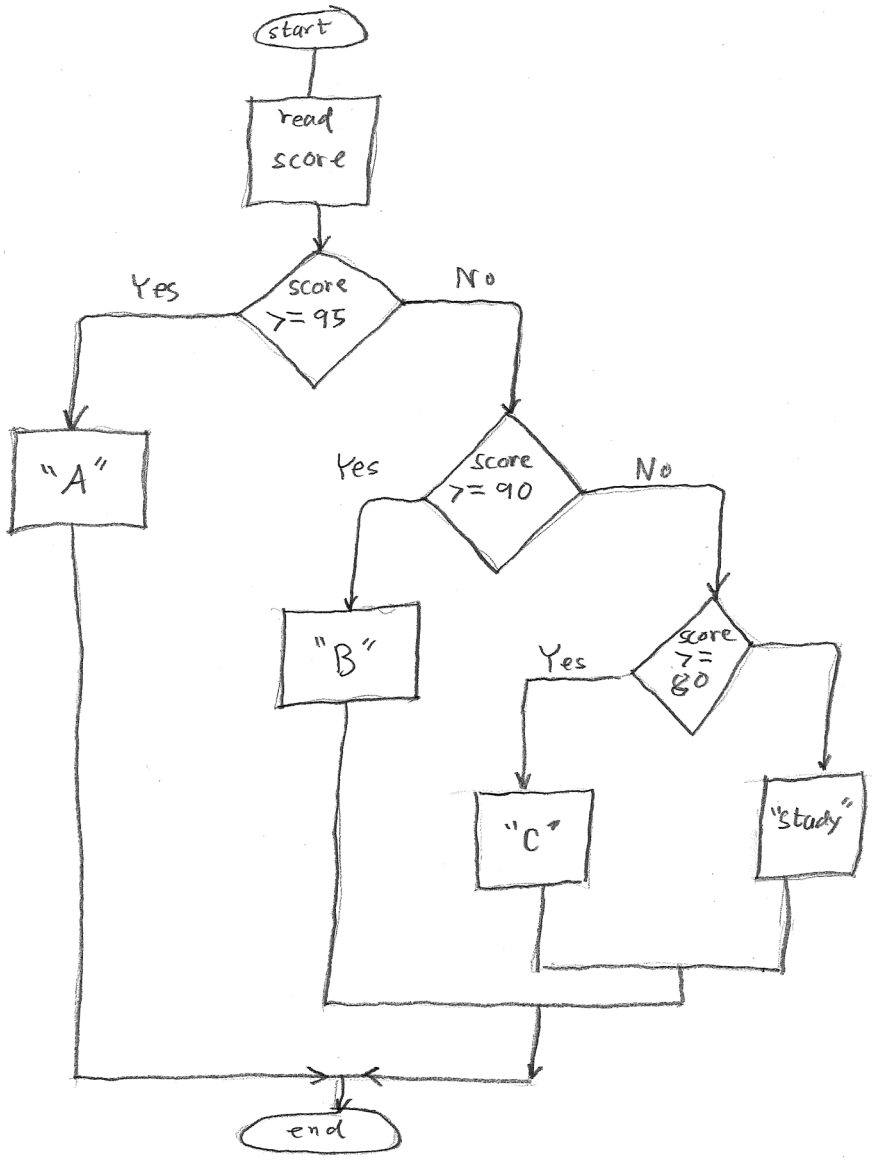
```
Enter score:
93
Grade: B
```

Sample Run

```
Enter score:
84
Grade: C
```

Sample Run

```
Enter score:
75
More Study Needed
```



Same Syntax, Different indenting style: else if

```
import java.util.Scanner;
public class Grade
{
   public static void main(String[] args)
   {
      int score;

      Scanner scan = new Scanner(System.in);

      System.out.println("Enter score: ");
      score = scan.nextInt();

      if (score>=95)
        System.out.println("Grade: A");
      else if (score>=90)
        System.out.println("Grade: B");
      else if (score>=80)
        System.out.println("Grade: C");
      else
        System.out.println("More Study Needed");
   }
}
```

Sample Run

```
Enter score:
84
Grade: C
```

Outline

- Boolean Expressions
- The `if` Statement
- Comparing Data
- The `while` Statement
  - ☺ Iterators
- The `ArrayList` Class
  - ☺ Determining Event Sources
  - ☺ Managing Fonts
  - ☺ Check Boxes and Radio Buttons

Summary

- Chapter 5 focused on:
  - boolean expressions
  - if and if-else statements