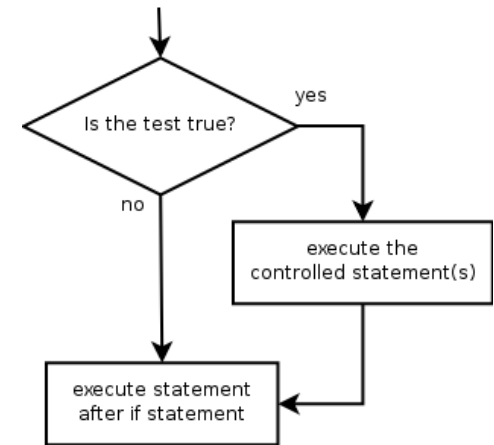


# Conditional Execution

# The `if` statement

*Executes a block of statements only if a test is true*

```
if (test) {  
    statement;  
    ...  
    statement;  
}
```



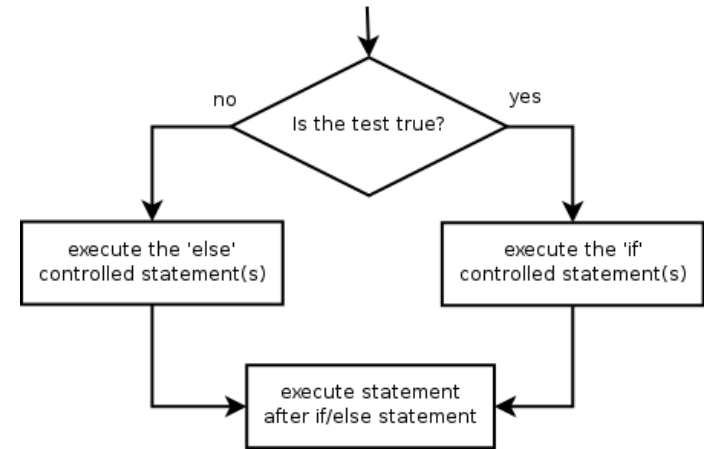
- Example:

```
double gpa = console.nextDouble();  
if (gpa >= 2.0) {  
    System.out.println("Application accepted.");  
}
```

# The if/else statement

*Executes one block if a test is true, another if false*

```
if (test) {  
    statement(s);  
} else {  
    statement(s);  
}
```



- **Example:**

```
double gpa = console.nextDouble();  
if (gpa >= 2.0) {  
    System.out.println("Welcome to Mars University!");  
} else {  
    System.out.println("Application denied.");  
}
```

# Relational expressions

- `if` statements and `for` loops both use logical tests.

```
for (int i = 1; i <= 10; i++) { ...  
if (i <= 10) { ...
```

– These are `boolean` expressions, seen in Ch. 5.

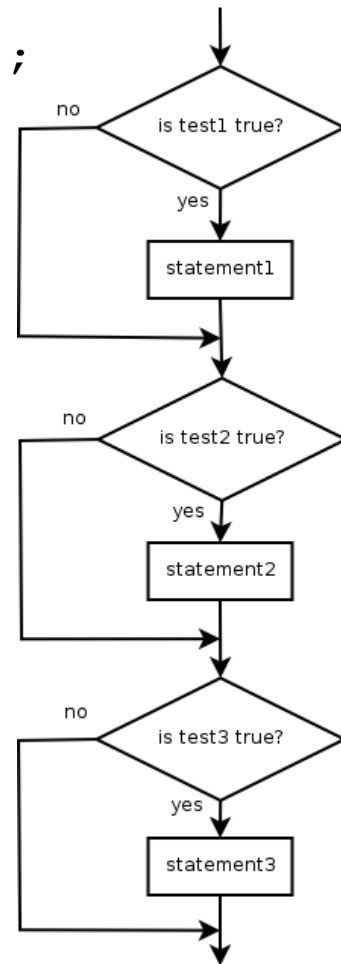
- Tests use *relational operators*:

Operator	Meaning	Example	Value
<code>==</code>	equals	<code>1 + 1 == 2</code>	true
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	true
<code>&lt;</code>	less than	<code>10 &lt; 5</code>	false
<code>&gt;</code>	greater than	<code>10 &gt; 5</code>	true
<code>&lt;=</code>	less than or equal to	<code>126 &lt;= 100</code>	false
<code>&gt;=</code>	greater than or equal to	<code>5.0 &gt;= 5.0</code>	true

# Misuse of if

- What's wrong with the following code?

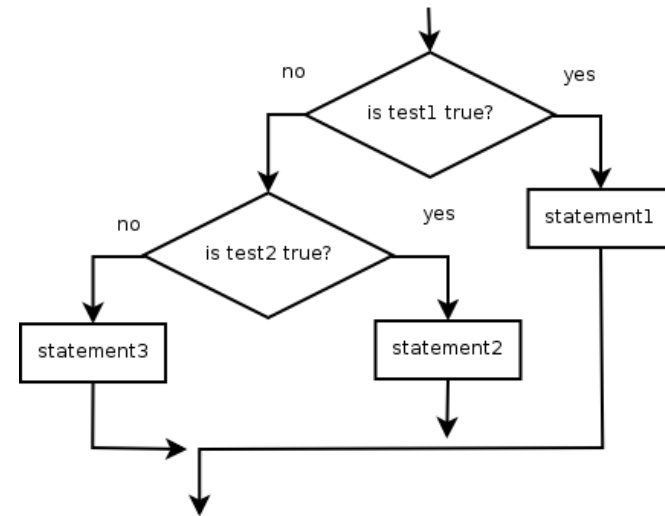
```
Scanner console = new Scanner(System.in);
System.out.print("What percentage did you earn? ");
int percent = console.nextInt();
if (percent >= 90) {
    System.out.println("You got an A!");
}
if (percent >= 80) {
    System.out.println("You got a B!");
}
if (percent >= 70) {
    System.out.println("You got a C!");
}
if (percent >= 60) {
    System.out.println("You got a D!");
}
if (percent < 60) {
    System.out.println("You got an F!");
}
...
```



# Nested if/else

*Chooses between outcomes using many tests*

```
if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
} else {  
    statement(s);  
}
```



- Example:

```
if (x > 0) {  
    System.out.println("Positive");  
} else if (x < 0) {  
    System.out.println("Negative");  
} else {  
    System.out.println("Zero");  
}
```

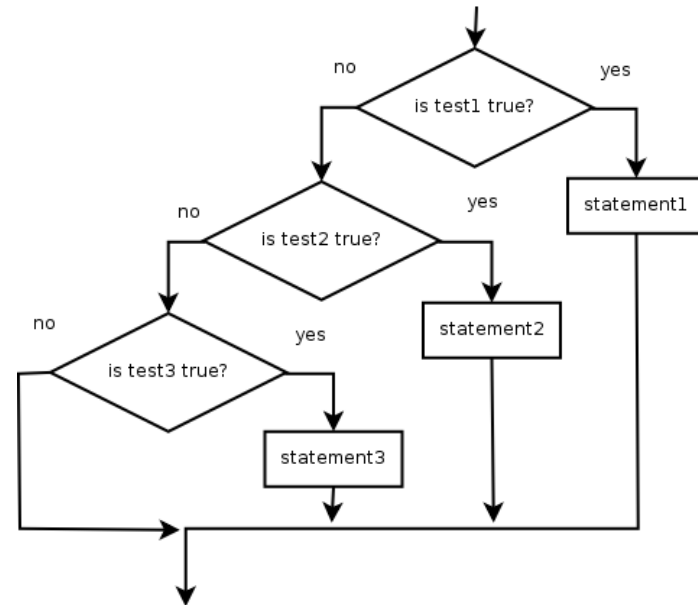
# Nested if/else/if

- If it ends with `else`, exactly one path must be taken.
- If it ends with `if`, the code might not execute any path.

```
if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
}
```

- Example:

```
if (place == 1) {  
    System.out.println("Gold medal!");  
} else if (place == 2) {  
    System.out.println("Silver medal!");  
} else if (place == 3) {  
    System.out.println("Bronze medal.");  
}
```



# Nested if structures

- exactly 1 path (*mutually exclusive*)

```
if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
} else {  
    statement(s);  
}
```

- 0 or 1 path (*mutually exclusive*)

```
if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
} else if (test) {  
    statement(s);  
}
```

- 0, 1, or many paths (*independent tests; not exclusive*)

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



# Which nested if/else?

- **(1) if/if/if    (2) nested if/else    (3) nested if/else/if**
  - Whether a user is lower, middle, or upper-class based on income.
    - **(2)**    nested `if / else if / else`
  - Whether you made the dean's list ( $\text{GPA} \geq 3.8$ ) or honor roll (3.5-3.8).
    - **(3)**    nested `if / else if`
  - Whether a number is divisible by 2, 3, and/or 5.
    - **(1)**    sequential `if / if / if`
  - Computing a grade of A, B, C, D, or F based on a percentage.
    - **(2)**    nested `if / else if / else if / else if / else`

# Nested if/else question

Formula for body mass index (BMI):

$$BMI = \frac{weight}{height^2} \times 703$$

BMI	Weight class
below 18.5	underweight
18.5 - 24.9	normal
25.0 - 29.9	overweight
30.0 and up	obese

- Write a program that produces output like the following:

```
This program reads data for two people and
computes their body mass index (BMI).
```

```
Enter next person's information:
```

```
height (in inches)? 70.0
```

```
weight (in pounds)? 194.25
```

```
Enter next person's information:
```

```
height (in inches)? 62.5
```

```
weight (in pounds)? 130.5
```

```
Person 1 BMI = 27.868928571428572
```

```
overweight
```

```
Person 2 BMI = 23.485824
```

```
normal
```

```
Difference = 4.3831045714285715
```

# Nested if/else answer

```
// This program computes two people's body mass index (BMI) and  
// compares them. The code uses Scanner for input, and parameters/returns.
```

```
import java.util.*; // so that I can use Scanner
```

```
public class BMI {  
    public static void main(String[] args) {  
        introduction();  
        Scanner console = new Scanner(System.in);  
  
        double bmi1 = person(console);  
        double bmi2 = person(console);  
  
        // report overall results  
        report(1, bmi1);  
        report(2, bmi2);  
        System.out.println("Difference = " + Math.abs(bmi1 - bmi2));  
    }  
  
    // prints a welcome message explaining the program  
    public static void introduction() {  
        System.out.println("This program reads data for two people and");  
        System.out.println("computes their body mass index (BMI).");  
        System.out.println();  
    }  
    ...  
}
```

# Nested if/else, cont'd.

```
// reads information for one person, computes their BMI, and returns it
public static double person(Scanner console) {
    System.out.println("Enter next person's information:");
    System.out.print("height (in inches)? ");
    double height = console.nextDouble();

    System.out.print("weight (in pounds)? ");
    double weight = console.nextDouble();
    System.out.println();

    double bodyMass = bmi(height, weight);
    return bodyMass;
}

// Computes/returns a person's BMI based on their height and weight.
public static double bmi(double height, double weight) {
    return (weight * 703 / height / height);
}

// Outputs information about a person's BMI and weight status.
public static void report(int number, double bmi) {
    System.out.println("Person " + number + " BMI = " + bmi);
    if (bmi < 18.5) {
        System.out.println("underweight");
    } else if (bmi < 25) {
        System.out.println("normal");
    } else if (bmi < 30) {
        System.out.println("overweight");
    } else {
        System.out.println("obese");
    }
}
}
```

# Logical operators

- Tests can be combined using *logical operators*:

Operator	Description	Example	Result
&&	and	(2 == 3) && (-1 < 5)	false
	or	(2 == 3)    (-1 < 5)	true
!	not	!(2 == 3)	true

- "Truth tables" for each, used with logical values  $p$  and  $q$ :

<b>p</b>	<b>q</b>	<b>p &amp;&amp; q</b>	<b>p    q</b>
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

<b>p</b>	<b>!p</b>
true	false
false	true

# Evaluating logic expressions

- Relational operators have lower precedence than math.

```
5 * 7 >= 3 + 5 * (7 - 1)
```

```
5 * 7 >= 3 + 5 * 6
```

```
35 >= 3 + 30
```

```
35 >= 33
```

```
true
```

- Relational operators cannot be "chained" as in algebra.

```
2 <= x <= 10
```

```
true <= 10
```

```
error!
```

(assume that x is 15)

- Instead, combine multiple tests with && or ||

```
2 <= x && x <= 10
```

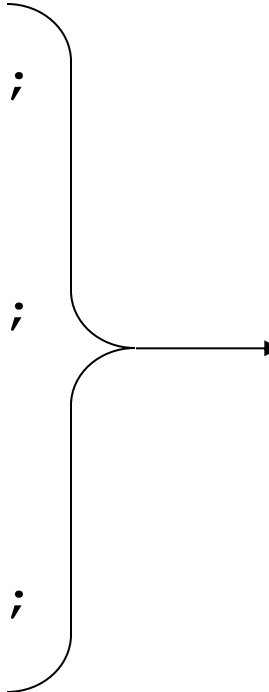
```
true && false
```

```
false
```

# Factoring if/else code

- **factoring:** Extracting common/redundant code.
  - Can reduce or eliminate redundancy from `if/else` code.
- Example:

```
if (a == 1) {  
    System.out.println(a);  
    x = 3;  
    b = b + x;  
} else if (a == 2) {  
    System.out.println(a);  
    x = 6;  
    y = y + 10;  
    b = b + x;  
} else { // a == 3  
    System.out.println(a);  
    x = 9;  
    b = b + x;  
}
```



```
System.out.println(a);  
x = 3 * a;  
if (a == 2) {  
    y = y + 10;  
}  
b = b + x;
```

# if/else with return

*// Returns the larger of the two given integers.*

```
public static int max(int a, int b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

- Methods can return different values using `if/else`
  - Whichever path the code enters, it will return that value.
  - Returning a value causes a method to immediately exit.
  - All paths through the code must reach a `return` statement.



# All paths must return

```
public static int max(int a, int b) {  
    if (a > b) {  
        return a;  
    }  
    // Error: not all paths return a value  
}
```

- The following also does not compile:

```
public static int max(int a, int b) {  
    if (a > b) {  
        return a;  
    } else if (b >= a) {  
        return b;  
    }  
}
```

- The compiler thinks `if/else/if` code might skip all paths, even though mathematically it must choose one or the other.

# if/else, return answer

```
public static int quadrant(double x, double y) {  
    if (x > 0 && y > 0) {  
        return 1;  
    } else if (x < 0 && y > 0) {  
        return 2;  
    } else if (x < 0 && y < 0) {  
        return 3;  
    } else if (x > 0 && y < 0) {  
        return 4;  
    } else {           // at least one coordinate equals 0  
        return 0;  
    }  
}
```

# **while loops**

# Categories of loops

- **definite loop:** Executes a known number of times.
  - The `for` loops we have seen are definite loops.
    - Print "hello" 10 times.
    - Find all the prime numbers up to an integer  $n$ .
    - Print each odd number between 5 and 127.
- **indefinite loop:** One where the number of times its body repeats is not known in advance.
  - Prompt the user until they type a non-negative number.
  - Print random numbers until a prime number is printed.
  - Repeat until the user has types "q" to quit.

# The while loop

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

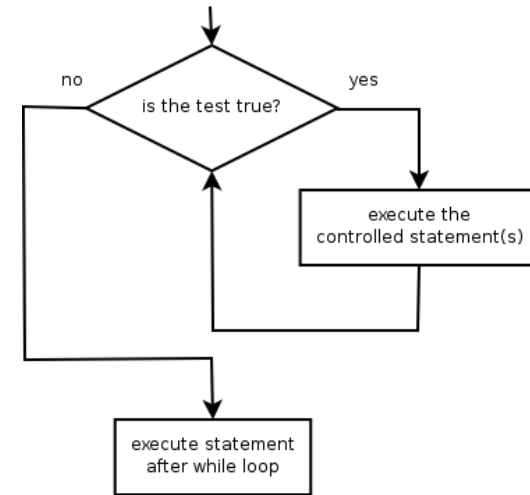
```
while (test) {  
    statement(s);  
}
```

- Example:

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

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

```
// initialization  
// test  
  
// update
```



# Example while loop

```
// finds the first factor of 91, other than 1
int n = 91;
int factor = 2;
while (n % factor != 0) {
    factor++;
}
System.out.println("First factor is " + factor);
// output: First factor is 7
```

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

# Sentinel values

- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.
- Example: Write a program that prompts the user for numbers until the user types 0, then outputs their sum.
  - (In this case, 0 is the sentinel value.)

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

# Flawed sentinel solution

- What's wrong with this solution?

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

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

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



# Changing the sentinel value

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

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

# Changing the sentinel value

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

```
Scanner console = new Scanner(System.in);  
int sum = 0;  
int number = 1;  // "dummy value", anything but -1  
  
while (number != -1) {  
    System.out.print("Enter a number (-1 to quit): ");  
    number = console.nextInt();  
    sum = sum + number;  
}  
  
System.out.println("The total is " + sum);
```

- Now the solution produces the wrong output. Why?

The total was 79