

Building Java Programs

Chapter 4: Conditional Execution

Lecture 4-1: `if` and `if/else` Statements

if/else Statements

reading: 4.2

self-check: #4-5, 7, 10, 11

exercises: #7

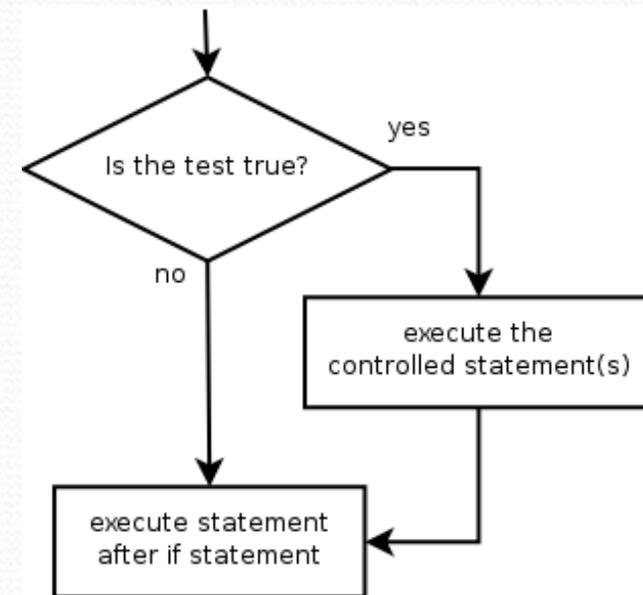
The `if` statement

Executes a block of statements only if a test is true

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

- **Example:**

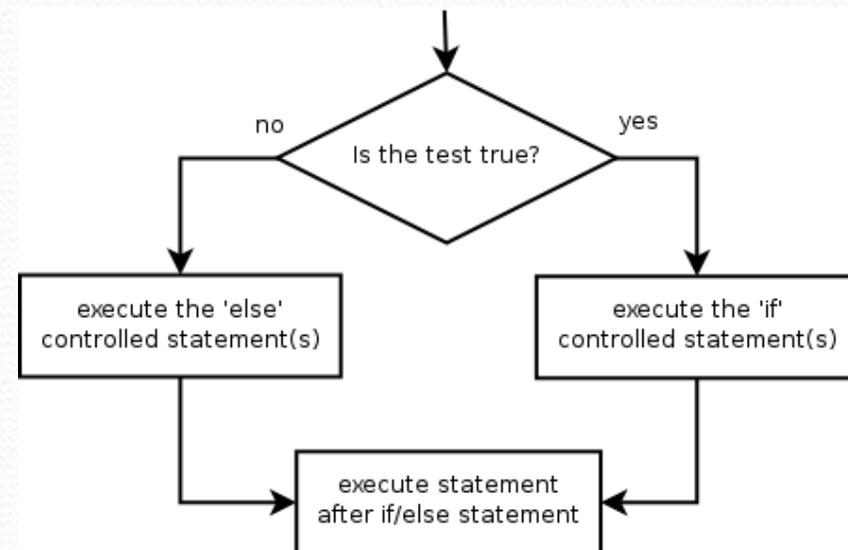
```
double gpa = console.nextDouble();  
if (gpa >= 2.0) {  
    System.out.println("Application accepted.");  
}
```



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

- A **<test>** in an `if` is the same as in a `for` loop.

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

- These tests are `boolean` expressions, seen in Ch. 5.
- These expressions 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><</code>	less than	<code>10 < 5</code>	false
<code>></code>	greater than	<code>10 > 5</code>	true
<code><=</code>	less than or equal to	<code>126 <= 100</code>	false
<code>>=</code>	greater than or equal to	<code>5.0 >= 5.0</code>	true

Logical operators: `&&`, `||`, `!`

- Conditions can be combined using *logical operators*:

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

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

p	q	p && q	p q
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

p	!p
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 (assume that x is 15)
```

```
true <= 10
```

```
error!
```

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

```
2 <= x && x <= 10 (assume that x is 15)
```

```
true && false
```

```
false
```

Logical questions

- What is the result of each of the following expressions?

```
int x = 42;
```

```
int y = 17;
```

```
int z = 25;
```

- `y < x && y <= z`
 - `x % 2 == y % 2 || x % 2 == z % 2`
 - `x <= y + z && x >= y + z`
 - `!(x < y && x < z)`
 - `(x + y) % 2 == 0 || !((z - y) % 2 == 0)`
- **Answers:** true, false, true, true, false

Loops with if/else

- if/else statements can be used with loops or methods:

```
// read 10 numbers and print how many were (non)negative
```

```
Scanner console = new Scanner(System.in);
```

```
System.out.print("Type 10 numbers: ");
```

```
int nonNegative = 0;
```

```
int negative = 0;
```

```
for (int i = 1; i <= 10; i++) {  
    int next = console.nextInt();  
    if (next >= 0) {  
        nonNegative++;  
    } else {  
        negative++;  
    }  
}
```

```
System.out.println(nonNegative + " non-negative");
```

```
System.out.println(negative + " negative");
```

Nested if/else

reading: 4.2

self-check: #6, 8

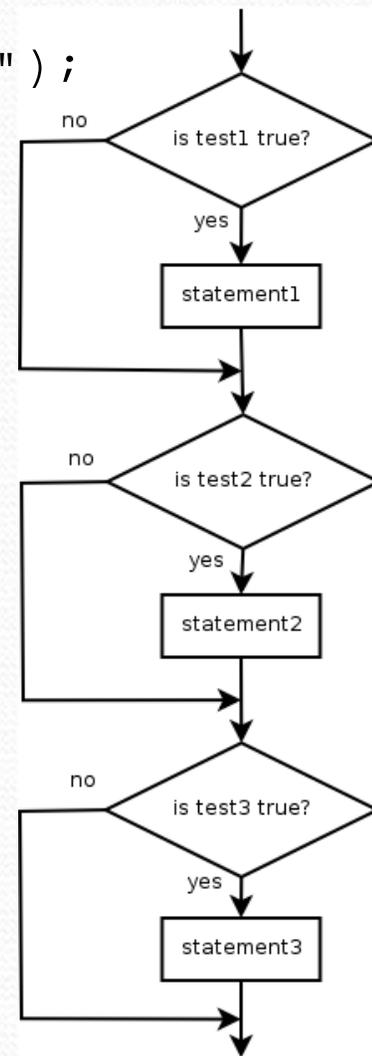
exercises: #10

Sequential if bug

- 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!");
}
else {
    System.out.println("You got an F!");
}
...

```



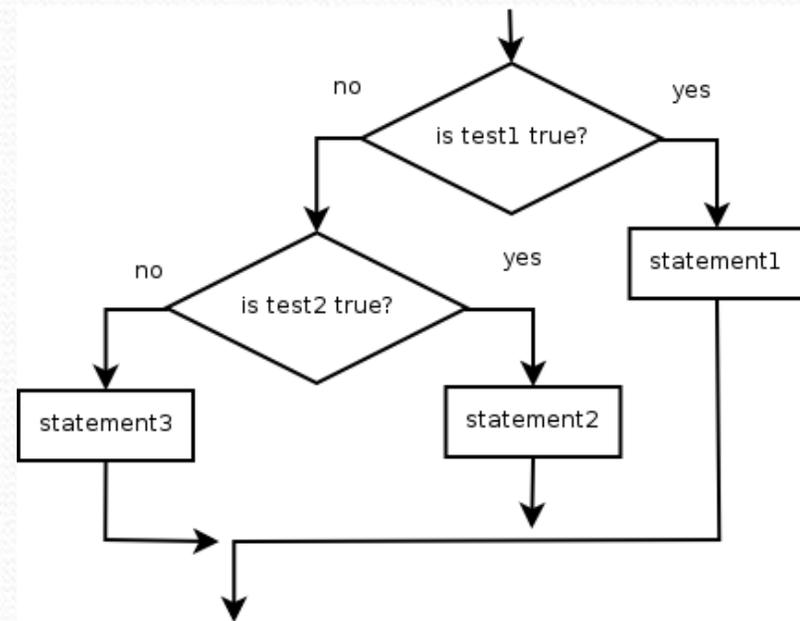
Nested if/else

Chooses between outcomes using many tests

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

- Example:

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



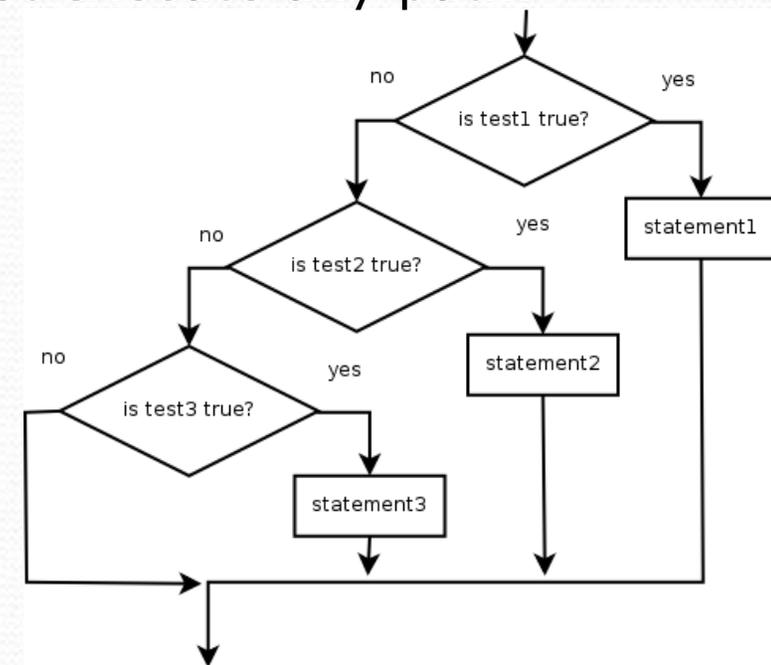
Nested if/else/if

- If it ends with `else`, one code path must be taken.
- If it ends with `if`, the program 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("You win the gold medal!");  
} else if (place == 2) {  
    System.out.println("You win a silver medal!");  
} else if (place == 3) {  
    System.out.println("You earned a bronze medal.");  
}
```

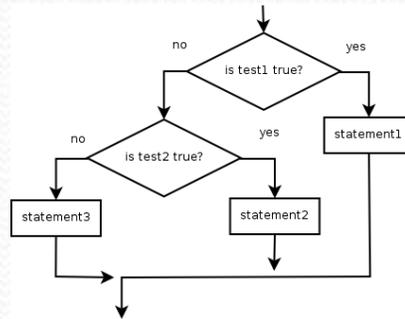


Structures

- Exactly 1 path: (mutually exclusive)

```

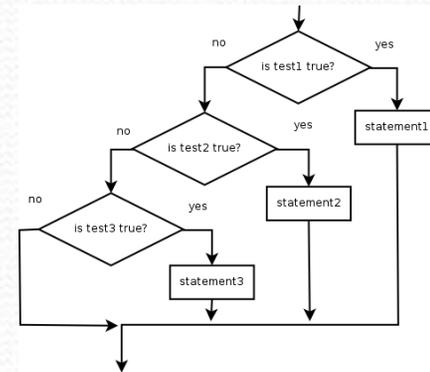
if ( <test> ) {
    <statement(s)>;
} else if ( <test> ) {
    <statement(s)>;
} else {
    <statement(s)>;
}
    
```



- 0 or 1 path:

```

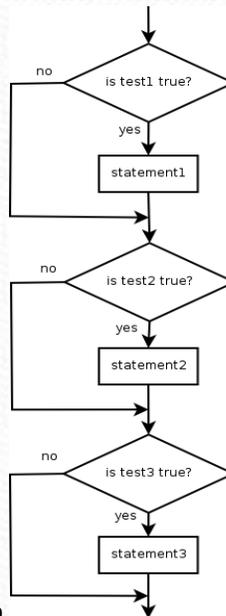
if ( <test> ) {
    <statement(s)>;
} else if ( <test> ) {
    <statement(s)>;
} else if ( <test> ) {
    <statement(s)>;
}
    
```



- 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**
 - Reading the user's GPA and printing whether the student is on the dean's list (3.8 to 4.0) or honor roll (3.5 to 3.8).
 - **(3) nested if / else if**
 - Printing whether a number is even or odd.
 - **(N/A) simple if / else**
 - Printing whether a user is lower-class, middle-class, or upper-class based on their income.
 - **(2) nested if / else if / else**
 - Reading a number from the user and printing whether it is divisible by 2, 3, and/or 5.
 - **(1) sequential if / if / if**
 - Printing 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 problem

- Modify our previous BMI program so it prints information about each person's weight according to the table at right.

Produce the following output:

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

```
Enter next person's information:
height (in inches)? 62.5
weight (in pounds)? 130.5
normal
```

```
Enter next person's information:
height (in inches)? 58.5
weight (in pounds)? 90
underweight
```

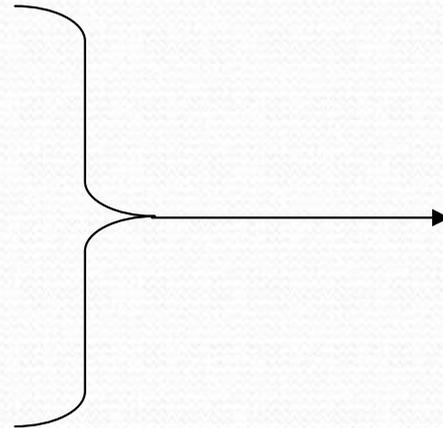
```
Person #1 body mass index = 23.485824
Person #2 body mass index = 18.487836949375414
Difference = 4.997987050624587
```

BMI	Status
below 18.5	underweight
18.5 - 24.9	normal
25 - 29.9	overweight
30 and up	obese

Factoring if/else code

- **factoring:** extracting common/redundant code
 - Factoring `if/else` code reduces the size of the `if` and `else` statements and can sometimes eliminate the need for `if/else` altogether.
- **Example:**

```
if (a == 1) {  
    x = 3;  
} else if (a == 2) {  
    x = 6;  
    y++;  
} else { // a == 3  
    x = 9;  
}
```



```
x = 3 * a;  
if (a == 2) {  
    y++;  
}
```

Code in need of factoring

```
if (money < 500) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Caution! Bet carefully.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
} else if (money < 1000) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Consider betting moderately.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
} else {
    System.out.println("You have, $" + money + " left.");
    System.out.print("You may bet liberally.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
}
```

Code after factoring

```
System.out.println("You have, $" + money + " left.");  
if (money < 500) {  
    System.out.print("Caution!  Bet carefully.");  
} else if (money < 1000) {  
    System.out.print("Consider betting moderately.");  
} else {  
    System.out.print("You may bet liberally.");  
}  
  
System.out.print("How much do you want to bet? ");  
bet = console.nextInt();
```

- If the start of each branch is the same, move it *before* the `if/else`.
- If the end of each branch is the same, move it *after* the `if/else`.
- If similar but code exists in each branch, look for patterns.

Methods with `if/else` and `return`

reading: 4.5

self-check: #9, 24-27

exercises: #11-14

if/else with return

- Methods can return different values using if/else:

// Returns the largest of the three given integers.

```
public static int max3(int a, int b, int c) {  
    if (a >= b && a >= c) {  
        return a;  
    } else if (b >= c && b >= a) {  
        return b;  
    } else {  
        return c;  
    }  
}
```

- Whichever path the code enters, it will return the appropriate value.
- Returning a value causes a method to immediately exit.
- All code paths must reach a `return` statement.

All paths must return

```
public static int max3(int a, int b, int c) {  
    if (a >= b && a >= c) {  
        return a;  
    } else if (b >= c && b >= a) {  
        return b;  
    }  
    // Error: not all paths return a value  
}
```

- The following also does not compile:

```
public static int max3(int a, int b, int c) {  
    if (a >= b && a >= c) {  
        return a;  
    } else if (b >= c && b >= a) {  
        return b;  
    } else if (c >= a && c >= b) {  
        return c;  
    }  
}
```

- The compiler thinks `if/else/if` code might skip all paths.

if/else, return question

- Write a method `countFactors` that returns the number of factors of an integer.
 - `countFactors(60)` returns 12 because 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60 are factors of 60.
- Write a program that prompts the user for a maximum integer and prints all prime numbers up to that max.

```
Maximum number? 52
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

```
15 primes (28.84%)
```

if/else, return answer 1

```
// Prompts for a maximum number and prints each prime up to that maximum.
import java.util.*;

public class Primes {
    public static void main(String[] args) {
        // read max from user
        Scanner console = new Scanner(System.in);
        System.out.print("Maximum number? ");
        int max = console.nextInt();
        printPrimes(max);
    }

    // Prints all prime numbers up to the given maximum.
    public static void printPrimes(int max) {
        int primes = 0;
        for (int i = 2; i <= max; i++) {
            if (countFactors(i) == 2) { // i is prime
                System.out.print(i + " ");
                primes++;
            }
        }
        System.out.println();

        double percent = 100.0 * primes / max;
        System.out.printf("%d primes (%.2f%%)\n", primes, percent);
    }
}
```

if/else, return answer 2

...

```
// Returns how many factors the given number has.  
public static int countFactors(int number) {  
    int count = 0;  
    for (int i = 1; i <= number; i++) {  
        if (number % i == 0) {  
            count++; // i is a factor of number  
        }  
    }  
    return count;  
}  
}
```