Building Java Programs

A Back to Basics Approach

CS 210

CHAPTER 4

CONDITIONAL EXECUTION

Please download the PPT, and use Slide Show for a better viewing experience

Winnie Li

Topics will be covered

CS 210

- The `if/else` statements
- Logical Operators and Factoring
- Cumulative Algorithms
- Text Processing
- System.out.printf

`if/else` Statements

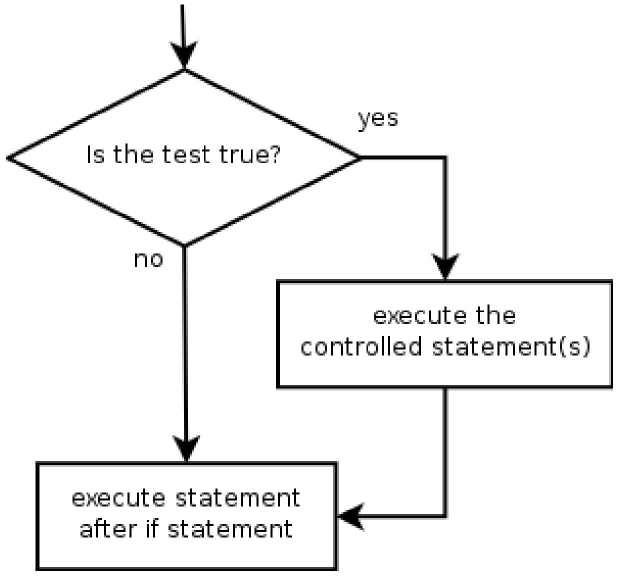
CS 210

IF STATEMENT
IF/ELSE STATEMENT
NESTED IF/ELSE STATEMENT
IF/ELSE/IF STATEMENT

The `if` statement

Executes a block of statements only if a test is true

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


Is the test true?
yes
no
execute the controlled statement(s)
execute statement after if statement

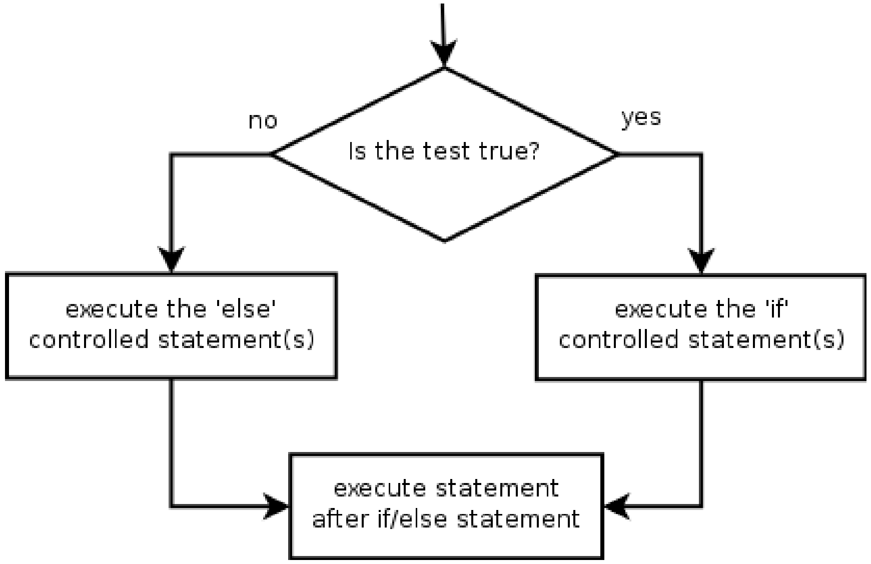
- Example:

```
double gpa = console.nextDouble();
if (gpa >= 3.5) {
    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);
}
```



Is the test true?
no
yes
execute the 'else' controlled statement(s)
execute the 'if' controlled statement(s)
execute statement after if/else statement

- Example:

```
double gpa = console.nextDouble();
if (gpa >= 3.5) {
    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, will be taught in Ch. 5.

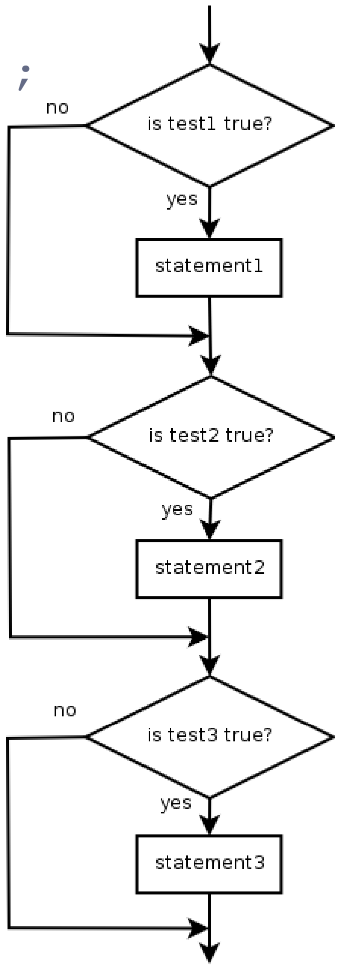
- Tests use *relational operators*:

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=`	does not equal	`3.2 != 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`

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!");
}
...
```

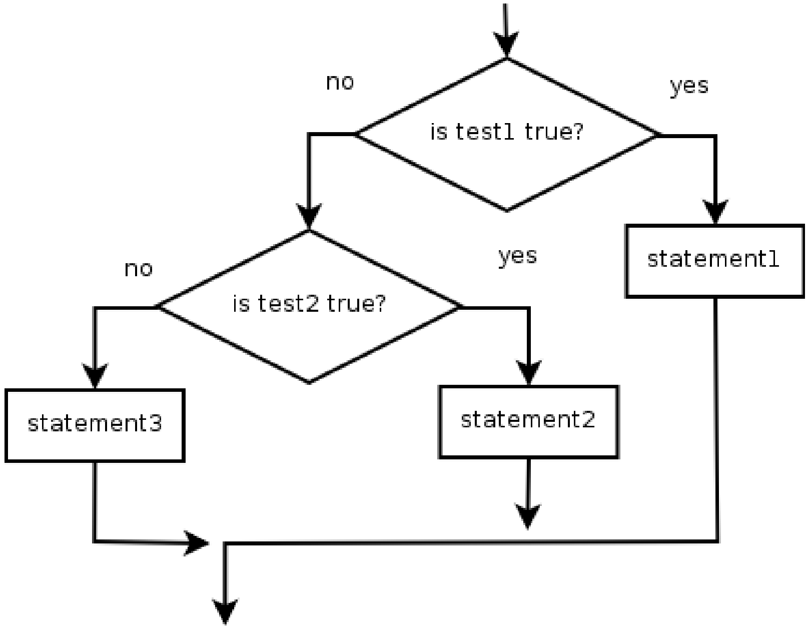


Nested `if/else`

CS 210

Chooses between outcomes using many tests

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



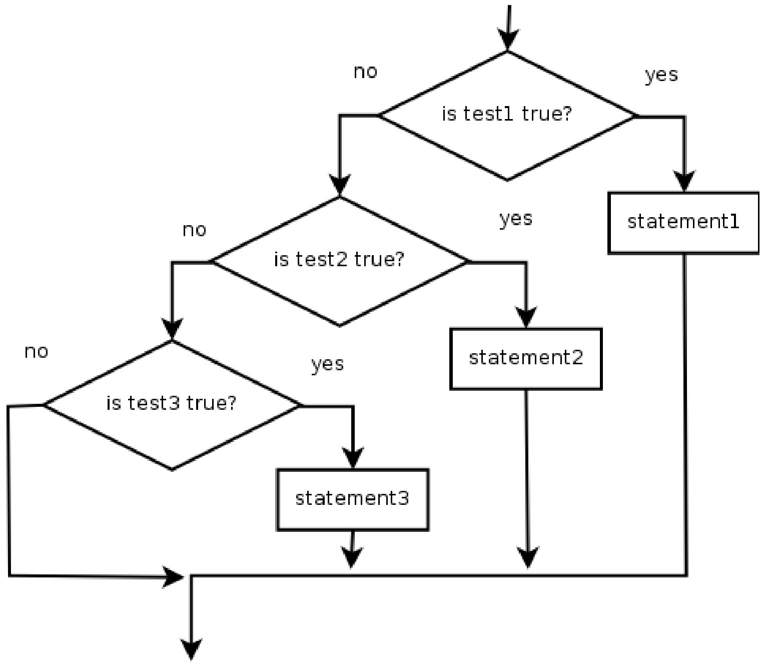
- Example:

```
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);
}
```

What is the output? 1

```
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!");
}
...
```

Input:

95
86
74
62
58
-12

What is the output? 2

CS 210

```
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!");
}
else if (percent >= 80) {
    System.out.println("You got a B!");
}
else if (percent >= 70) {
    System.out.println("You got a C!");
}
else if (percent >= 60) {
    System.out.println("You got a D!");
}
else {
    System.out.println("You got an F!");
}
...
```

Input:

95
86
74
62
58
-12

What is the output? 3

CS 210

```
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!");
}
else if (percent >= 80) {
    System.out.println("You got a B!");
}
else if (percent >= 70) {
    System.out.println("You got a C!");
}
else if (percent >= 60) {
    System.out.println("You got a D!");
}
else if (percent > 0) {
    System.out.println("You got an F!");
}
...
```

Input:

95
86
74
62
58
-12

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 (GPA ≥ 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`

Loops with if/else

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

```
int evenSum = 0;
int oddSum = 0;
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        evenSum = evenSum + i;
    } else {
        oddSum = oddSum + i;
    }
}
System.out.println("Even sum: " + evenSum);
System.out.println("Odd sum: " + oddSum);
```

The `if/else` hammer

- Just because you learned a new construct does not mean that every new problem has to be solved using that construct!

```
int z;
if (x > y) {
    z = x;
} else {
    z = y;
}
```

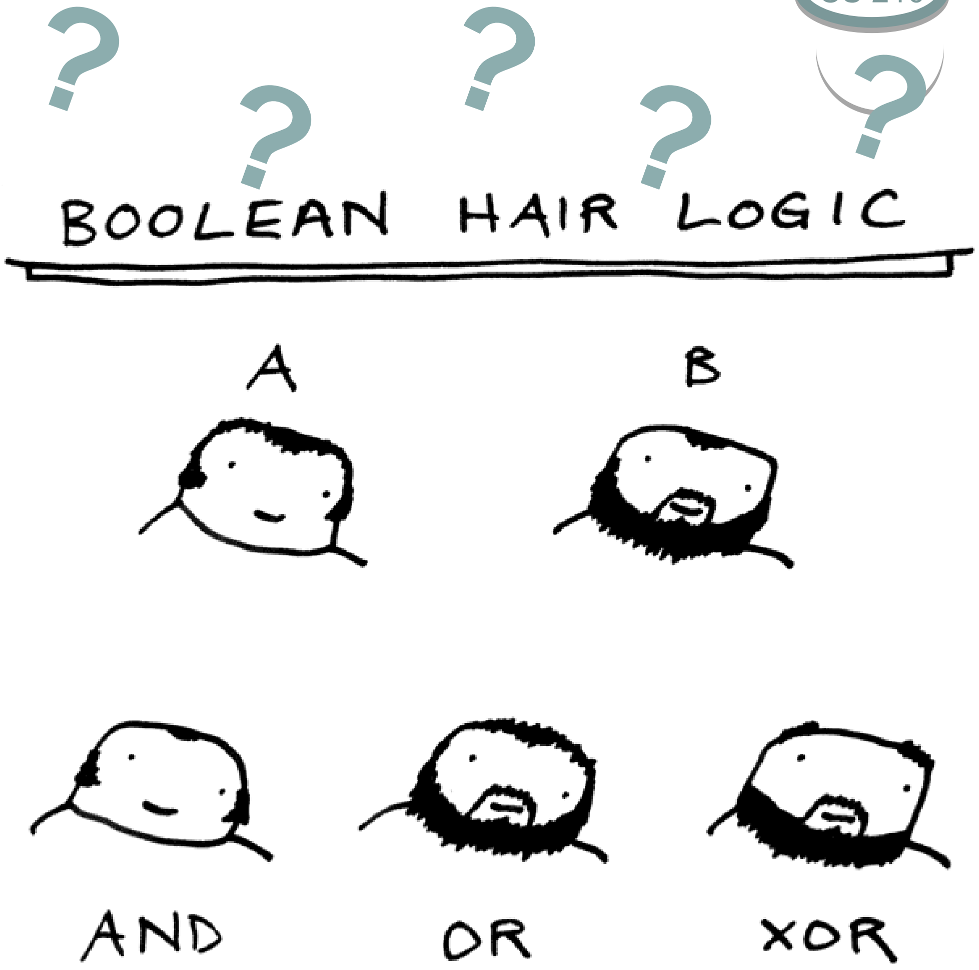
```
int z = Math.max(x, y);
```

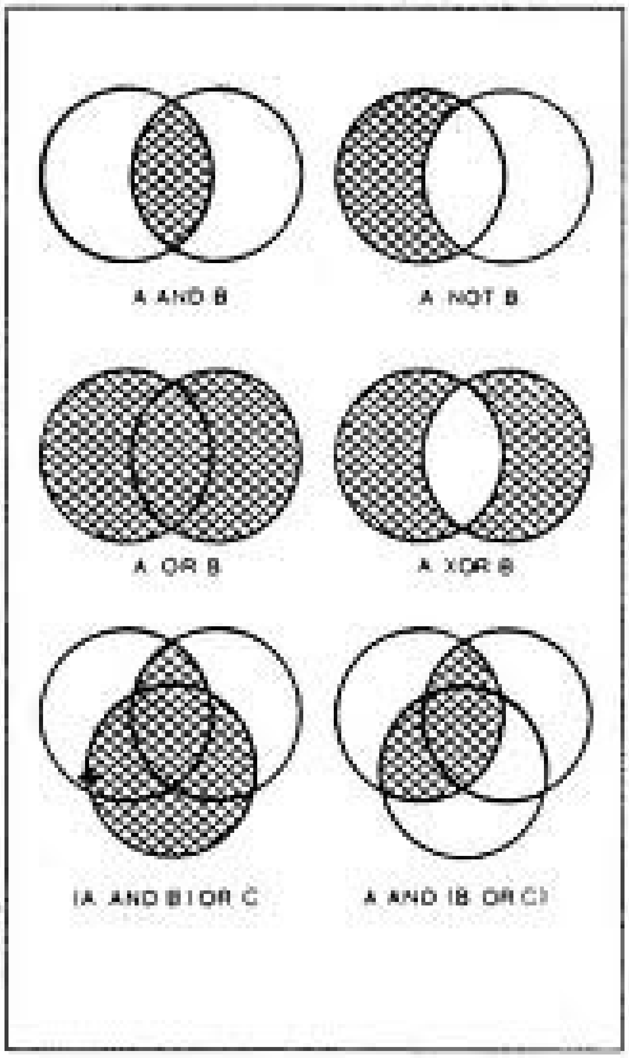
Logical Operators and Factoring

CS 210

LOGICAL OPERATORS
LOGICAL EXPRESSIONS
FACTORING
IF/ELSE AND RETURN

AND? OR? NOT?


BOOLEAN HAIR LOGIC
A
B
AND
OR
XOR


A AND B
A NOT B
A OR B
A XOR B
(A AND B) OR C
A AND (B OR C)

Logical operators

- Tests can be combined using *logical operators*:

Operato r	Descriptio n	Example	Resul t
`&&`	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*:

p	q	p && q	p || q
`true`	`true`	`true`	`true`
`true`	`fals e`	`false`	`true`
`fals e`	`true`	`false`	`true`
`fals`	`fals`	`false`	`false`

p	!p
`true`	`false`
`fals e`	`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
```

Logical Exercises

- #1 - #5: What is the result of each of the following expressions? True or False?

```
int x = 42;
int y = 17;
int z = 25;
```

```
1.  y < x && y <= z
2.  x % 2 == y % 2 || x % 2 == z % 2
3.  x <= y + z && x >= y + z
4.  !(x < y && x < z)
5.  (x + y) % 2 == 0 || !((z - y) % 2 == 0)
```

- Fun Question: Write a program that prompts for information about a person and uses it to decide whether to date them.

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;
```

The "dangling if" problem

- What can be improved about the following code?

```
if (x < 0) {
    System.out.println("x is negative");
} else if (x >= 0) {
    System.out.println("x is non-negative");
}
```

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
}
```

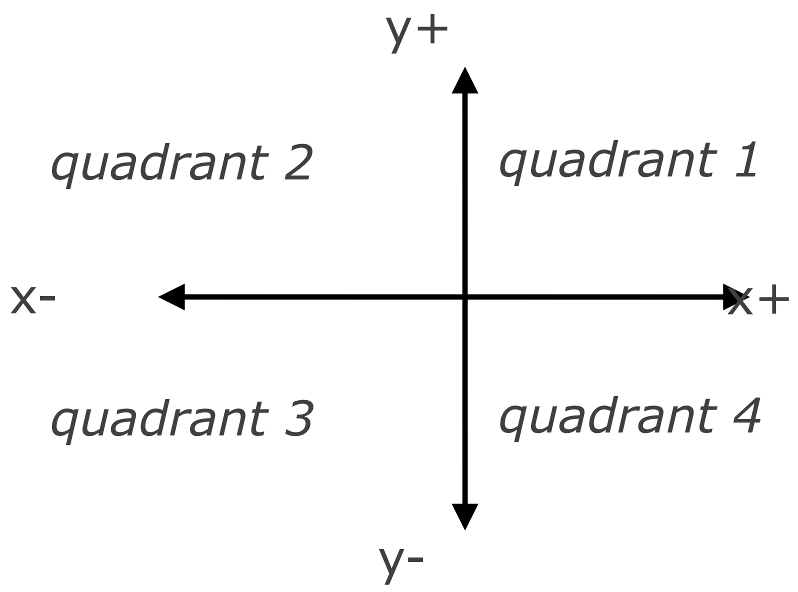
- The following also does not compile:

```
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.
  - How can we fix it?

`if/else, return` question

- Write a method `quadrant` that accepts a pair of real numbers *x* and *y* and returns the quadrant for that point:



  - Example: `quadrant(-4.2, 17.3)` returns `2`

  If the point falls directly on either axis, return `0`.

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;
    }
}
```

Cumulative Algorithms

CS 210

CUMULATIVE SUM
CUMULATIVE PRODUCT
RECEIPT EXAMPLE

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
// This may require a lot of typing
int sum = 1 + 2 + 3 + 4 + ... ;
System.out.println("The sum is " + sum);
```

- What if we want the sum from 1 - 1,000,000? Or the sum up to any maximum?
  - How can we generalize the above code?

Cumulative sum loop

```
int sum = 0;
for (int i = 1; i <= 1000; i++) {
    sum = sum + i;
}
System.out.println("The sum is " + sum);
```

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The `sum` in the above code is an attempt at a cumulative sum.
  - Cumulative sum variables must be declared *outside* the loops that update them, so that they will still exist after the loop.

Cumulative product

- This cumulative idea can be used with other operators:

```
int product = 1;
for (int i = 1; i <= 20; i++) {
    product = product * 2;
}
System.out.println("2 ^ 20 = " + product);
```

  - Any other way to achieve this?
  - How would we make the base and exponent adjustable?

Scanner and cumul. sum

- We can do a cumulative sum of user input:

```
Scanner console = new Scanner(System.in);
int sum = 0;
for (int i = 1; i <= 100; i++) {
    System.out.print("Type a number: ");
    sum = sum + console.nextInt();
}
System.out.println("The sum is " + sum);
```

Cumulative sum question

- Modify the `Receipt` program from Ch. 2.
  - Prompt for how many people, and each person's dinner cost.
  - Use static methods to structure the solution.
- Example log of execution:

```
How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 30.0
Person #4: How much did your dinner cost? 10.00

Subtotal: $75.0
Tax: $6.0
Tip: $11.25
Total: $92.25
```

Cumulative sum answer

```
// This program enhances our Receipt program using a cumulative sum.
import java.util.*;

public class Receipt2 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        double subtotal = meals(console);
        results(subtotal);
    }

    // Prompts for number of people and returns total meal subtotal.
    public static double meals(Scanner console) {
        System.out.print("How many people ate? ");
        int people = console.nextInt();
        double subtotal = 0.0;           // cumulative sum

        for (int i = 1; i <= people; i++) {
            System.out.print("Person #" + i +
                             ": How much did your dinner cost? ");
            double personCost = console.nextDouble();
            subtotal = subtotal + personCost;  // add to sum
        }
        return subtotal;
    }
    ...
```

Cumulative answer, cont'd.

```
    ...

    // Calculates total owed, assuming 8% tax and 15% tip
    public static void results(double subtotal) {
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        System.out.println("Subtotal: $" + subtotal);
        System.out.println("Tax: $" + tax);
        System.out.println("Tip: $" + tip);
        System.out.println("Total: $" + total);
    }
}
```

Strings

CS 210

INDEXES
STRING METHODS
COMPARE STRINGS

Recall: Strings

- **string**: An object storing a sequence of text characters.
  - Unlike most other objects, a `String` is not created with `new`.

    ```
    String <name> = "<text>";
    String <name> = <expression with String value>;
    ```

  - Examples:

    ```
    String name = “Winnie Li";
    int x = 3;
    int y = 5;
    String point = "(" + x + ", " + y + ")";
    ```

Indexes

- Characters of a string are numbered with 0-based *indexes*:

```
String name = "M. Mouse";
```

index	0	1	2	3	4	5	6	7
character	`M`	`.`		`M`	`o`	`u`	`s`	`e`

  - First character's index : 0
  - Last character's index : 1 less than the string's length

  - The individual characters are values of type `char` (seen later)

`String` methods

CS 210

Method name	Description
`indexOf(`***<string>***`)`	index where the start of the given string appears in this string (-1 if not found)
`length()`	number of characters in this string
`substring(`***<index1>***`, `***<index2>***`)` or `substring(`***<index1>***`)`	the characters in this string from *index1* (inclusive) to *index2* (exclusive); if *index2* is omitted, grabs until end of string
`toLowerCase()`	a new string with all lowercase letters
`toUpperCase()`	a new string with all uppercase letters

- These methods are called using the dot notation:

```
String popStarz = "Prince vs. Michael";
```

String method examples

```
// index      012345678901
String s1 = "Stuart Reges";
String s2 = "Marty Stepp";

System.out.println(s1.length());          // 12
System.out.println(s1.indexOf("e"));      // 8
System.out.println(s1.substring(7, 10)); // "Reg"

String s3 = s2.substring(1, 7);
System.out.println(s3.toLowerCase()); // "arty s"
```

- Given the following string:

```
// index         0123456789012345678901
String book = "Building Java Programs";
```

  - How would you extract the word `"Java"` ?

```
System.out.println(book.substring(9, 13));
```

Modifying strings

- Methods like `substring` and `toLowerCase` build and return a new string, rather than modifying the current string.

```
String s = "Mumford & Sons";
s.toUpperCase();
System.out.println(s);    // Mumford & Sons
```

- To modify a variable's value, you must reassign it:

```
String s = "Mumford & Sons";
s = s.toUpperCase();
System.out.println(s);    // MUMFORD & SONS
```

Strings as user input

- `Scanner`'s `next` method reads a word of input as a `String`.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
name = name.toUpperCase();
System.out.println(name + " has " + name.length() +
    " letters and starts with " + name.substring(0, 1));
```

Output:

```
What is your name? Winnie
Winnie has 6 letters and starts with W
```

- The `nextLine` method reads a line of input as a `String`.

```
System.out.print("What is your address? ");
String address = console.nextLine();
```

Comparing strings

- Relational operators such as < and == fail on objects.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Barney") {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

  - This code will compile, but it will not print the song.
  - == compares objects by *references* (seen later), so it often gives `false` even when two `String`s have the same letters.

The `equals` method

- Objects are compared using a method named `equals`.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Barney")) {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

  - Technically this is a method that returns a value of type `boolean`, the type used in logical tests.

String test methods

CS 210

Method	Description
`equals(`**`str`**`)`	whether two strings contain the same characters
`equalsIgnoreCase(`**`str`**`)`	whether two strings contain the same characters, ignoring upper vs. lower case
`startsWith(`**`str`**`)`	whether one contains other's characters at start
`endsWith(`**`str`**`)`	whether one contains other's characters at end
`contains(`**`str`**`)`	whether the given string is found within this one

```
String name = console.next();
if (name.startsWith("Prof")) {
    System.out.println("When are your office hours?");
} else if (name.equalsIgnoreCase("STUART")) {
    System.out.println("Let's talk about meta!");
}
```

Text Processing

CS 210

TYPE CHAR
COMPARE CHAR VALUE
CHARACTER METHODS

Type `char`

CS 210

- **`char`** : A primitive type representing single characters.
  - A `String` is stored internally as an array of `char`

```
String s = "Ali G.";
```

index	0	1	2	3	4	5
value	`'A'`	`'l'`	`'i'`	`' '`	`'G'`	`'.'`

  - It is legal to have variables, parameters, returns of type `char`
    - surrounded with apostrophes: `'a'` or `'4'` or `'\n'` or `'\''`

```
char letter = 'L';
System.out.println(letter);                   // L
System.out.println("Prof. " + letter);        // Prof. L
```

The charAt method

- The chars in a String can be accessed using the charAt method.
  - accepts an int index parameter and returns the char at that index

```
String food = "cookie";
char firstLetter = food.charAt(0);    // 'c'
System.out.println(firstLetter + " is for " + food);
```

- You can use a for loop to print or examine each character.

```
String major = "CSE";
for (int i = 0; i < major.length(); i++) {     // output:
    char c = major.charAt(i);                  // C
    System.out.println(c);                     // S
}                                              // E
```

Comparing `char` values

- You can compare `char`s with `==`, `!=`, and other operators:

```
String word = console.next();
char last = word.charAt(word.length() - 1);
if (last == 's') {
    System.out.println(word + " is plural.");
}

// prints the alphabet
for (char c = 'a'; c <= 'z'; c++) {
    System.out.print(c);
}
```

`char` vs. `int`

- Each `char` is mapped to an integer value internally
  - Called an **ASCII value**

    `'A'` is 65 `'B'` is 66 `' '` is 32
    `'a'` is 97 `'b'` is 98 `'*'` is 42

  - Mixing `char` and `int` causes automatic conversion to `int`.

    `'a' + 10` is 107, `'A' + 'A'` is 130

  - To convert an `int` into the equivalent `char`, type-cast it.

    `(char) ('a' + 2)` is `'c'`

char vs. String

- "h" is a String, but 'h' is a char *(they are different)*
- A String is an object; it contains methods.

```
String s = "h";
s = s.toUpperCase();           // "H"
int len = s.length();          //  1
char first = s.charAt(0);      // 'H'
```

- A char is primitive; you can't call methods on it.

```
char c = 'h';
c = c.toUpperCase();                // ERROR
s = s.charAt(0).toUpperCase();      // ERROR
```

  - What is s + 1? What is c + 1?
  - What is s + s? What is c + c?

Character methods

Method	Description
`getUnmericValue(`**Ch**`)`	converts a “numeric” character into number
`isDigit(`**Ch**`)`	whether or not a character is one of digits ‘0’ through ‘9’
`isLetter(`**Ch**`)`	whether or not a character is one of letters
`isLowerCase(`**Ch**`)`	whether or not a character is a lowercase letter
`isUpperCase(`**Ch**`)`	whether or not a character is a uppercase letter
`toLowerCase(`**Ch**`)`	converts a character into the lowercase version
`toUpperCase(`**Ch**`)`	converts a character into the uppercase version

```
Character.getNumericValue(‘6’)   returns 6
Character.isDigit(‘C’)           returns false
Character.isLowerCase(‘h’)       returns true
Character.toUpperCase(‘e’)       returns ‘E’
```

`string` and `Character` Methods Exercises

Assume that the following variables have been declared:

```
String a = "Ready, Set, Go!";
String b = a.substring(5, 10);
char b1 = b.charAt(2);
```

Evaluate the following expressions:

```
#1. Character.isLowerCase(b1)
#2. Character.toLowerCase(b1)
#3. a.charAt(2 + a.indexOf("e"))
#4. b + 5
#5. b1 + 5
```

Cumulative text algorithm examples

- accepts a `string` and a `char` and returns the number of times the character occurs in the string.

```
int found = 0;
for (int i = 0; i < text.length(); i++) {
    if (text.charAt(i) == 'i') {
        found++;
    }                                   // "Winnie" returns 2
}
return found;
```

- accepts a `string` and returns the same `char` in the reverse order.

```
String result = "";
for (int i = 0; i < phrase.length(); i++) {   // "stressed"
    result = text.charAt(i) + result;         // returns
}                                             //
"desserts"
return result;
```

System.out.printf

CS 210

Formatting text with `printf`

CS 210

`System.out.printf(`**`"format string"`**`, `**`parameters`**`);`

- A format string can contain *placeholders* to insert parameters:

`%d`	integer
`%f`	real number
`%s`	string

  - NOTE: these placeholders are used instead of + concatenation

- Example:

```
int x = 3;
int y = -17;
System.out.printf("x is %d and y is %d!\n", x, y);
                    // x is 3 and y is -17!
```

  - NOTE: `printf` does not drop to the next line unless you write `\n`

`printf` width

`%`**W**`d`	integer, **W** characters wide, right-aligned
`%-`**W**`d`	integer, **W** characters wide, *left*-aligned
`%`**W**`f`	real number, **W** characters wide, right-aligned
`...`	

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.printf("%4d", (i * j));
    }
    System.out.println();   // to end the line
}
```

Output:

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
```

`printf` precision

CS 210

- `%.`**D**`f` real number, rounded to **D** digits after decimal
- `%`**W**`.`**D**`f` real number, **W** chars wide, **D** digits after decimal
- `%-`**W**`.`**D**`f` real number, **W** wide (left-align), **D** after decimal

```
double gpa = 3.253764;
System.out.printf("your GPA is %.1f\n", gpa);
System.out.printf("more precisely: %8.3f\n", gpa);
```

Output:

```
your GPA is 3.3
more precisely:    3.254
```

$\underbrace{\texttt{\ \ \ 3.}\overbrace{\texttt{254}}^{3}}_{8}$

`printf` question

- Modify our `Receipt` program to better format its output.
  - Display results in the format below, with `$` and 2 digits after `.`

- Example log of execution:

```
How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 25.0
Person #4: How much did your dinner cost? 10.00

Subtotal:  $70.00
Tax:       $5.60
Tip:       $10.50
Total:     $86.10
```

printf answer (partial)

```
    ...
    // Calculates total owed, assuming 8% tax and 15% tip
    public static void results(double subtotal) {
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        // System.out.println("Subtotal: $" + subtotal);
        // System.out.println("Tax: $" + tax);
        // System.out.println("Tip: $" + tip);
        // System.out.println("Total: $" + total);

        System.out.printf("Subtotal: $%.2f\n", subtotal);
        System.out.printf("Tax:      $%.2f\n", tax);
        System.out.printf("Tip:      $%.2f\n", tip);
        System.out.printf("Total:    $%.2f\n", total);
    }
}
```

The End

CS 210

CHAPTER 4

CONDITIONAL EXECUTION

Winnie Li

if/else, return question

- Write a method `countFactors` that returns the number of factors of an integer.
  - `countFactors(24)` returns `8` because 1, 2, 3, 4, 6, 8, 12, and 24 are factors of 24.

- Solution:

```
// 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;
}
```

Nested `if/else` example

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 = 4.383104571428572
```

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");
        }
    }
}
```

Scanners as parameters

- If many methods need to read input, declare a `Scanner` in `main` and pass it to the other methods as a parameter.

```
public static void main(String[] args) {
    Scanner console = new Scanner(System.in);
    int sum = readSum3(console);
    System.out.println("The sum is " + sum);
}

// Prompts for 3 numbers and returns their sum.
public static int readSum3(Scanner console) {
    System.out.print("Type 3 numbers: ");
    int num1 = console.nextInt();
    int num2 = console.nextInt();
    int num3 = console.nextInt();
    return num1 + num2 + num3;
}
```