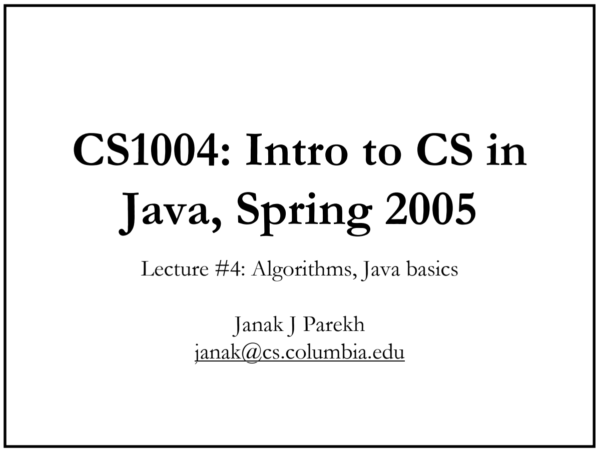
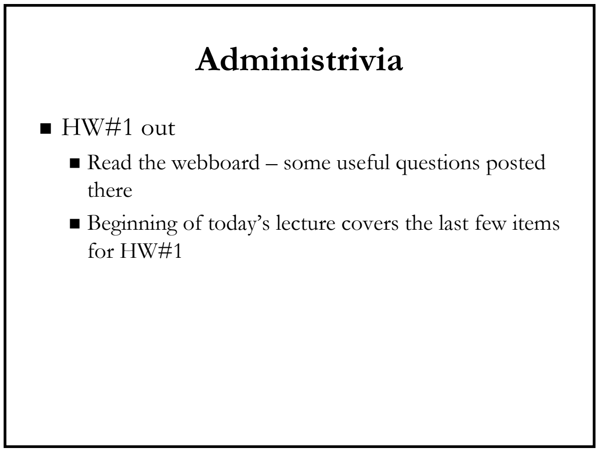


CS1004: Intro to CS in Java, Spring 2005

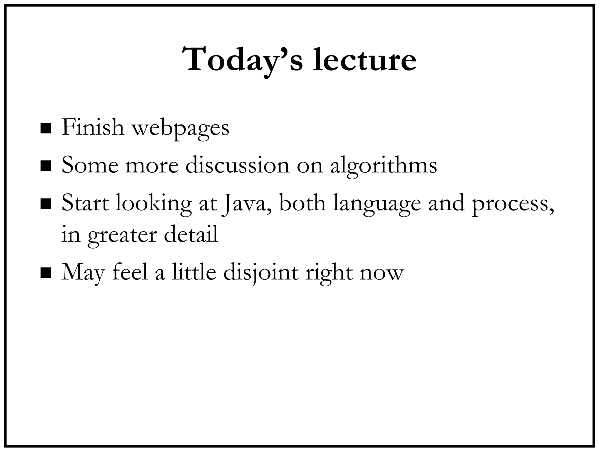
Lecture #4: Algorithms, Java basics

Janak J Parekh
janak@cs.columbia.edu



Administrivia

- HW#1 out
  - Read the webboard – some useful questions posted there
  - Beginning of today's lecture covers the last few items for HW#1



Today's lecture

- Finish webpages
- Some more discussion on algorithms
- Start looking at Java, both language and process, in greater detail
- May feel a little disjoint right now

Transferring files, revisited

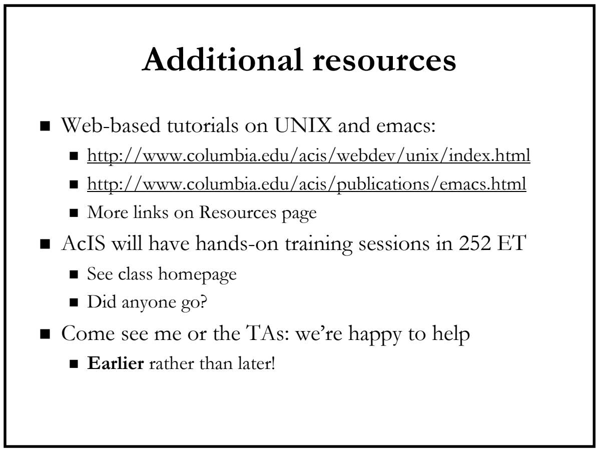
- How to use WinSCP?
  - Look at the step-by-step guide on the WinSCP download page
  - http://www.columbia.edu/acis/software/winscp/
  - Let's take a quick look

Webpage permissions revisited

- chmod a+r index.html
  - Gives everyone permission to read the file
  - Can also use *filename globbing:* chmod a+r *.html
  - Filename globbing works for other commands, too
- chmod a+rx public_html
  - Gives everyone permission to enter and list the contents of public_html

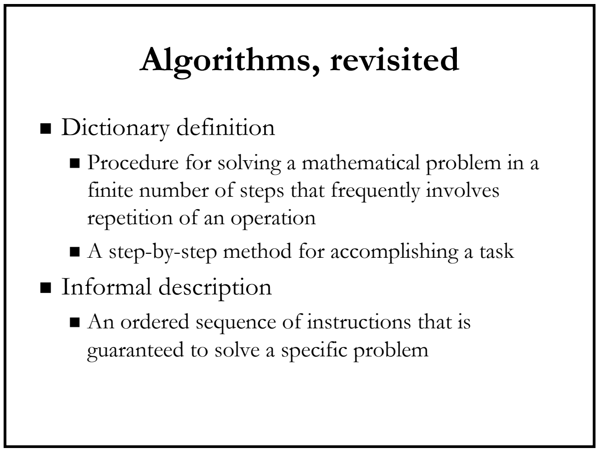
Useful HTML tags

- Hyperlinks: <A HREF="http://www.google.com">Click here to visit Google</A>
  - "Local" links: <A HREF="friends.html">Here's a list of my friends</A>
- Images: <IMG SRC="test.gif">
- Lots more; check out w3schools.com or some other site
- Let's add a little to my page



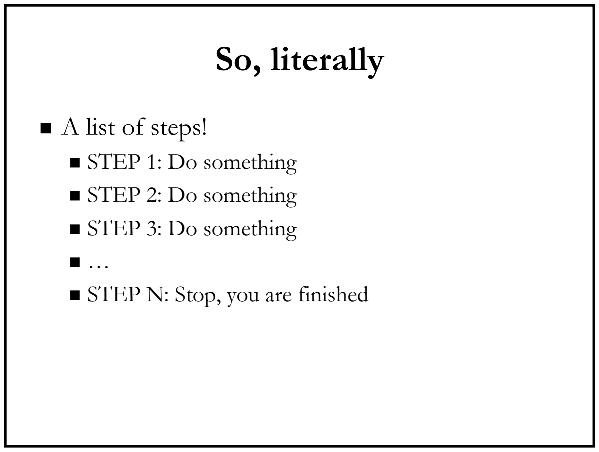
Additional resources

- Web-based tutorials on UNIX and emacs:
  - http://www.columbia.edu/acis/webdev/unix/index.html
  - http://www.columbia.edu/acis/publications/emacs.html
  - More links on Resources page
- AcIS will have hands-on training sessions in 252 ET
  - See class homepage
  - Did anyone go?
- Come see me or the TAs: we’re happy to help
  - **Earlier** rather than later!



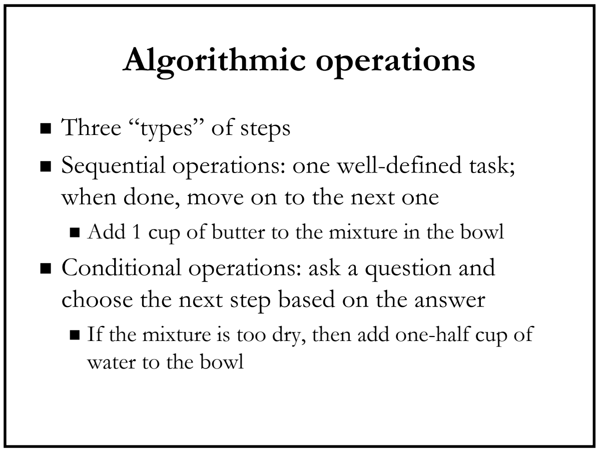
Algorithms, revisited

- Dictionary definition
  - Procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation
  - A step-by-step method for accomplishing a task
- Informal description
  - An ordered sequence of instructions that is guaranteed to solve a specific problem



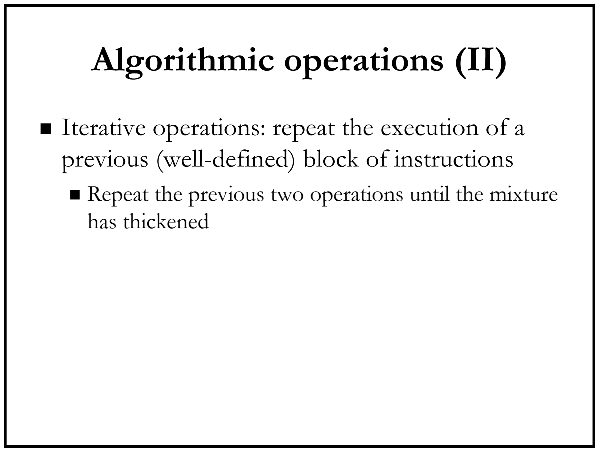
So, literally

- A list of steps!
  - STEP 1: Do something
  - STEP 2: Do something
  - STEP 3: Do something
  - …
  - STEP N: Stop, you are finished



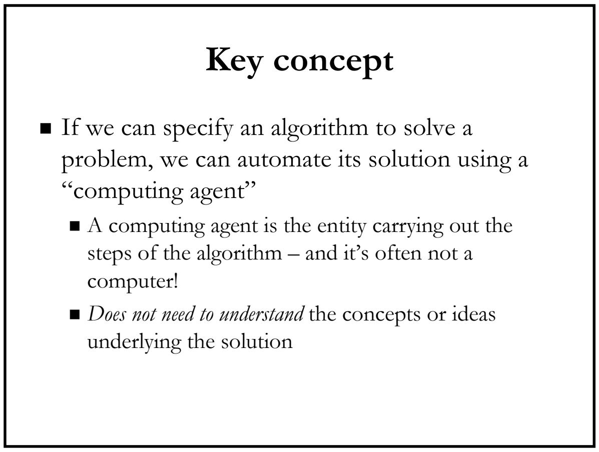
Algorithmic operations

- Three "types" of steps
- Sequential operations: one well-defined task; when done, move on to the next one
  - Add 1 cup of butter to the mixture in the bowl
- Conditional operations: ask a question and choose the next step based on the answer
  - If the mixture is too dry, then add one-half cup of water to the bowl



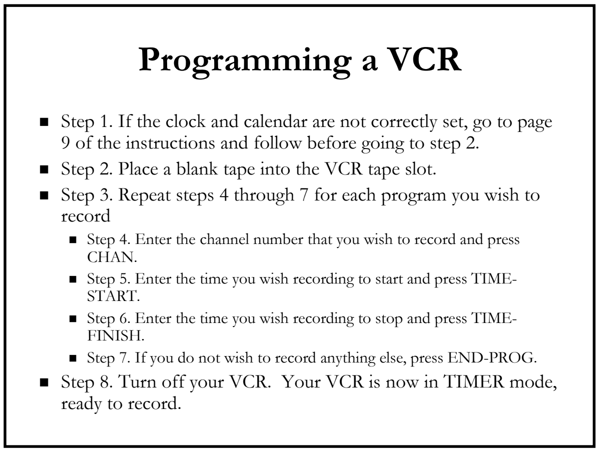
Algorithmic operations (II)

- Iterative operations: repeat the execution of a previous (well-defined) block of instructions
  - Repeat the previous two operations until the mixture has thickened



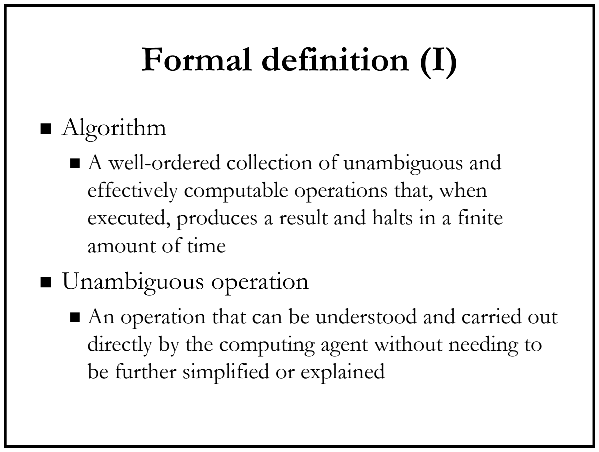
Key concept

- If we can specify an algorithm to solve a problem, we can automate its solution using a "computing agent"
  - A computing agent is the entity carrying out the steps of the algorithm – and it's often not a computer!
  - *Does not need to understand* the concepts or ideas underlying the solution



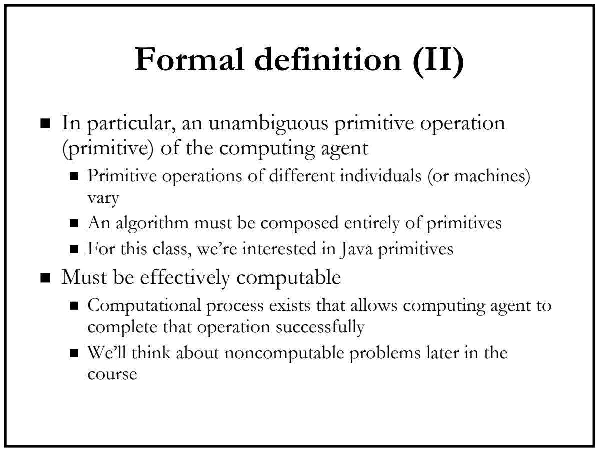
Programming a VCR

- Step 1. If the clock and calendar are not correctly set, go to page 9 of the instructions and follow before going to step 2.
- Step 2. Place a blank tape into the VCR tape slot.
- Step 3. Repeat steps 4 through 7 for each program you wish to record
  - Step 4. Enter the channel number that you wish to record and press CHAN.
  - Step 5. Enter the time you wish recording to start and press TIME-START.
  - Step 6. Enter the time you wish recording to stop and press TIME-FINISH.
  - Step 7. If you do not wish to record anything else, press END-PROG.
- Step 8. Turn off your VCR. Your VCR is now in TIMER mode, ready to record.



Formal definition (I)

- Algorithm
  - A well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time
- Unambiguous operation
  - An operation that can be understood and carried out directly by the computing agent without needing to be further simplified or explained



Formal definition (II)

- In particular, an unambiguous primitive operation (primitive) of the computing agent
  - Primitive operations of different individuals (or machines) vary
  - An algorithm must be composed entirely of primitives
  - For this class, we're interested in Java primitives
- Must be effectively computable
  - Computational process exists that allows computing agent to complete that operation successfully
  - We'll think about noncomputable problems later in the course

Formal definition (III)

- The result of the algorithm must be produced after the execution of a finite number of operations
- Otherwise, you have the dreaded "infinite loop"
  - The algorithm has no provisions to terminate
  - Common error in the designing of algorithms (or typos in your code)

Brief history of Computer Science

- Started long before it was called CS – "computing machines"
- 3,000 years ago: Mathematics, logic, and numerical computation
- 1614: Logarithms
- Around 1622: First slide rule created
- 1672: The Pascaline – Pascal's mechanical calculator
- 1674: Leibnitz's Wheel
  - Could do addition, subtraction, multiplication, and division

History (cont'd.)

- 1801: The Jacquard loom
  - Automated loom – used punch cards
- 1823: Babbage's Difference Engine
  - Addition, subtraction, multiplication, and division to 6 significant digits
  - Solved polynomial equations and other complex mathematical problems
- 1830s: Babbage's Analytic Engine
  - Designed, but never actually implemented
  - Had primitive concepts of arithmetic/logic, memory, processor, and I/O
  - Assistant, Ada Lovelace, widely considered one of the first women in computing

History (cont'd.)

- 1890: U.S. census carried out with Herman Hollerith's programmable card processing machines
  - These machines could automatically read, tally, and sort data entered on punched cards
  - His company, Tabulating Machine Corporation, became IBM in 1924

Electronic computers

- Began after 1940, fueled in large part by needs of World War II
- Early computers included the Mark I, ENIAC, ABC system, Colossus, Z1
- Vacuum tube-based switches (memory)
- Programs were still generally physically hardwired

Stored programs

- The idea of actually storing the program in computer memory itself was proposed by John Von Neumann in 1946
  - Is known as the Von Neumann architecture, we'll study it next week
  - Modern computers remain, fundamentally, Von Neumann machines
  - First stored program computers
    - EDVAC
    - EDSAC

The shrinking of machines

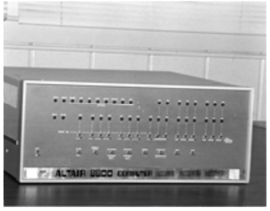
- 1950s: Still vacuum tubes, computers multiple rooms in size, failures frequent
- Early 1960s: Replaced vacuum tubes with transistors and magnetic cores
  - Computer could fit into a single room, higher reliability, reduced cost
  - Rise of high-level programming languages
    - Fortran first popular one, started in late 50s
    - The programmer occupation was born

Minicomputers

- Late 1960s: use of IC instead of individual transistors, evolution of desk-sized computer
  - Software industry formed
  - Still only for larger businesses
  - Minicomputers still exist today (although much smaller; http://www.as400.ibm.com)

Microcomputers

- 1975-1985: Continued miniaturization
  - Reduced to the size of a typewriter (microcomputer)
  - Desktop and personal computers common
  - Appearance of networks, email, GUIs, embedded systems
- Bill Gates and Paul Allen wrote a port of BASIC for the Altair 8800 (pictured at right)



Modern computing

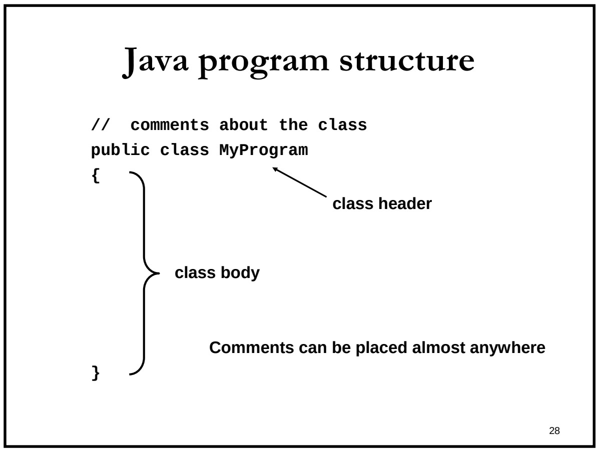
- Recent developments
  - Massively parallel processors
  - Handheld devices/PDAs
  - High-resolution graphics
  - Powerful multimedia user interfaces
  - Integrated global telecommunications, wireless data
  - Massive storage devices
  - Ubiquitous computing
- What's the next big thing?

Programming languages, redux

- A *programming language* specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*

Java program structure

- In the Java programming language:
  - A program is made up of one or more *classes*
    - Basic building blocks of a Java program
  - A class contains one or more *methods*
    - Think of them as well-defined "functions"
  - A method contains program *statements*
  - *Comments* are textual documentation
- Curly braces ({, }) used to "enclose" methods and statements
- A Java application always contains at least one class, with a method called main



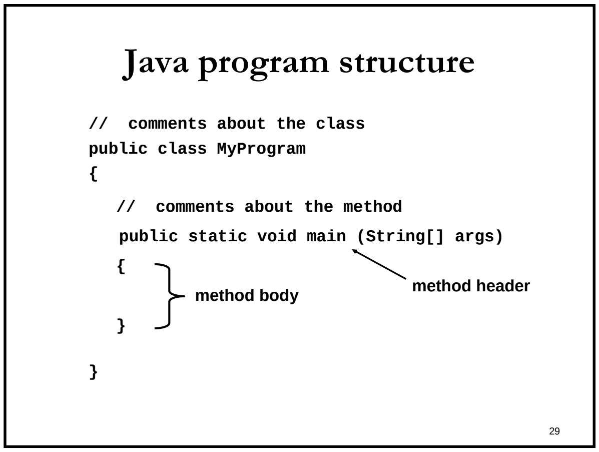
Java program structure

```
//  comments about the class
public class MyProgram
{



}
```

class header

class body

Comments can be placed almost anywhere

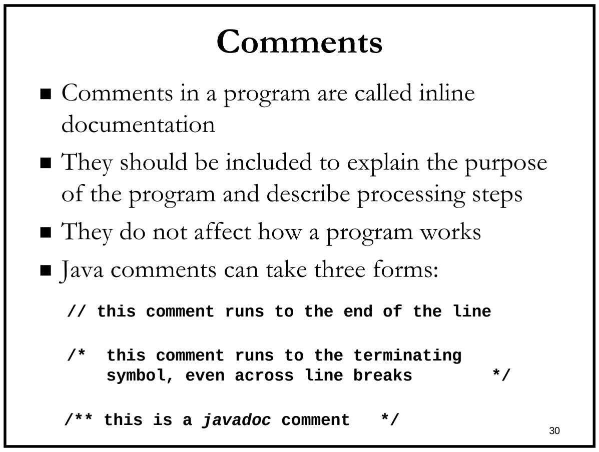


Java program structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

method header

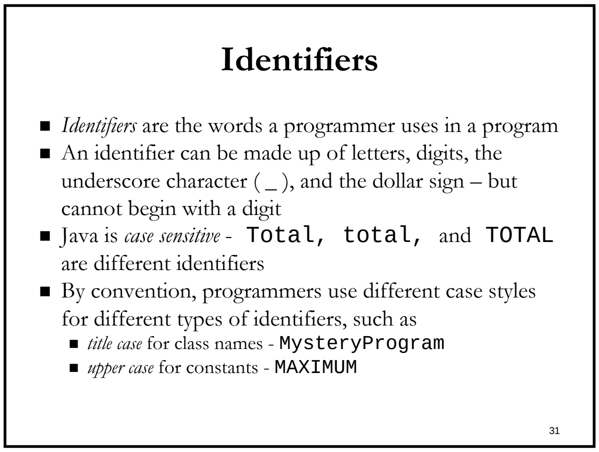
method body



Comments

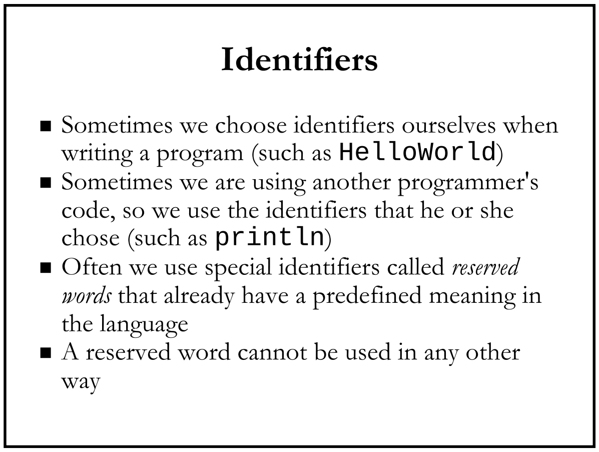
- Comments in a program are called inline documentation
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Java comments can take three forms:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks        */

/** this is a javadoc comment   */
```



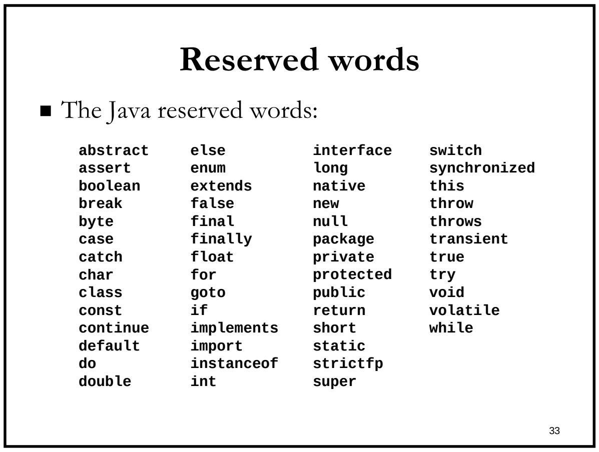
Identifiers

- *Identifiers* are the words a programmer uses in a program
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign – but cannot begin with a digit
- Java is *case sensitive* - `Total`, `total`, and `TOTAL` are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - `MysteryProgram`
  - *upper case* for constants - `MAXIMUM`



Identifiers

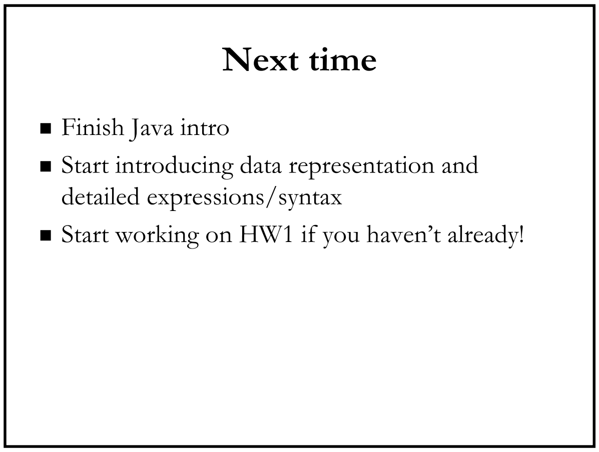
- Sometimes we choose identifiers ourselves when writing a program (such as `HelloWorld`)
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way



Reserved words

- The Java reserved words:

```
abstract    else         interface   switch
assert      enum         long        synchronized
boolean     extends      native      this
break       false        new         throw
byte        final        null        throws
case        finally      package     transient
catch       float        private     true
char        for          protected   try
class       goto         public      void
const       if           return      volatile
continue    implements   short       while
default     import       static
do          instanceof   strictfp
double      int          super
```



Next time

- Finish Java intro
- Start introducing data representation and detailed expressions/syntax
- Start working on HW1 if you haven’t already!