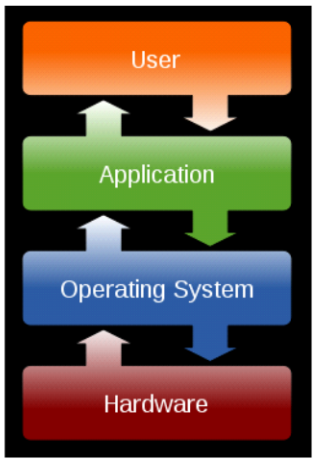
Fundamentals of Java Coding

Friday, January 08, 2010
12:48 PM



Hardware and Software

Hardware
- Actual physical item you see before you (monitor, CPU, keyboard, mouse, speakers)

Software
- Programs that tell these pieces how to function (Applications and Operating Systems)

Operating System
- Code that tells the hardware how to do something
- Examples: how to add, how to move the cursor, etc.
- Vista, Windows 7, Linux, Mac DOS

Application
- Program that tells the Operating System which command to execute.
- Examples: to add 2 numbers, to close a window, etc.
- Microsoft Word, Internet Explorer, Paint

Input / Output

Input Devices
- Ways the use communicates with the computer
- Keyboards, Mouse, Microphones, etc.

Output Devices
- Ways the computer communicates with the user
- Speakers, Screen

Memory Types

Primary Storage
- Extremely fast since it interacts directly with the CPU
- Built into the computer -- you can't remove primary storage
- Anything stored in primary storage is lost after the computer powers down

Secondary Storage
- Relatively slower than Primary Storage
- Tangible objects are used for secondary storage (hard drive, CD-ROM, Memory Stick, Floppy Disk, etc)
- Storage is permanent and will be retained even the computer powers down

RAM vs. ROM
- RAM, or **Random Access Memory**, is the most common type of primary storage.
- ROM, or **Read-Only Memory**, is the kind of secondary storage used on CD-ROMs
  - ROM cannot be altered once it has been written to memory -- a file saved on a CD-ROM can never be removed from the disk.
  - CD-RWs, or CD-Rewriteable, is a kind of secondary storage that is saved after a computer powers down but can still be overwritten. This is also the kind of storage used on memory sticks.

Memory Unit Table

Unit	Number of Bytes
Bit	0.125 (1/8 of a byte)
Byte	1 (2^0)
Kilobyte	2^{10}
Megabyte	2^{20}
Gigabyte	2^{30}
Terabyte	2^{40}
Petabyte	2^{50}
Exabyte	2^{60}
Zettabyte	2^{70}
Yottabyte	2^{80}

After this there was mathematical and consumer confusion. Terms sprang up like "kilobyte" that people thought just meant 1,000 bytes but really meant 1024. Although this wasn't so significant in lower memory values, when you got into massive memory amounts the distinctions were huge. Thus, computer scientists created a new set of terms for to be more precise, and the old terminology was transformed to meet consumer expectations.

Kibibyte	2^{10}

Kilobyte	10^3

Mebibyte	2^{20}
Gibibyte	2^{30}
Tebibyte	2^{40}
Pebibyte	2^{50}
Exbibyte	2^{60}
Zebibyte	2^{70}
Yobibyte	2^{80}

Megabyte	10^{6}
Gigabyte	10^{9}
Terabyte	10^{12}
Petabyte	10^{15}
Zettabyte	10^{21}
Yottabyte	10^{24}
Exabyte	10^{18}

Base Conversions

- Because we have 10 fingers, we use Base-10, or decimal. A base is basically the column system for recording the numbers. Base-10 means that the only possible digits are 0-9, and once we reach a multiple of 10 we shift a column.
- Computers read all data in machine language, represented by 1s and 0s alone. The mathematical base that uses only 1s and 0s is called "binary code," which uses multiples of 2 to shift columns.
- Two other common bases are octal (Base-8) and hexidecimal (Base-16). In order to represent numbers 10-15 in a single hexidecimal column letters A-F are used.

Basic Conversion

- Create a table with columns representing each power of the base
- Start with the left-most column that a non-zero value will fill and divide the number by that column's value.
- Record the quotient in the column.
- Take the remainder and continue to divide it among the columns until it is 0.
- Any remaining column is set to 0.
- The left-to-right number is your new converted number.

Shorthand Conversion

- Divide your number by 2 and get a quotient and remainder.
- Repeat this step by continuing to divide the quotient by 2 to get a new quotient and remainder.
- When the quotient reaches 0, read the remainders from the last one to the first one. The bottom-to-top reading is your new converted number.

Proof of Shorthand Method

If you multiply substitute the expressions for each quotient back into its original equation, and continue to stuff back all the way to the top expression, you will find a string that repeats the syntax "2 x (something)." If you multiply that string out you will get the powers of 2 that you would have gotten in the basic conversion method.

Language Levels

- A **programming language** is a language through which you speak to the computer and communicate what you want it to do. The computer learns the language instantly with a single download, but it takes much longer for humans to learn to speak it.
- Programming languages all build off each other. The more advanced languages (high level) are far more understandable but you can do less with them -- the less understandable the language (lower level) the more potential it has to manipulate your computer.

Machine Language

- Composed of only 1s and 0s
- Nearly impossible to code
- Can manipulate bits of computer memory

Assembly (Low-level) Language

- Composed of some numbers and characters
- Very difficult language to code
- Advanced viruses are often written in it

High-Level Language

- English words are used in the code
- Code is far easier and understandable
- Less you can do with it (but you can still do a lot)

Why Java?

- If something goes wrong in Java you can always just exit the program -- it has its own safety net
- The **Java Virtual Machine** interprets the code the same on all computers, unlike other languages that need to be recompiled

Developing Code

Stage 1: Flow Chart

- If you can't figure out where to begin, draw a diagram that will basically depict what your code will do and the general way to tackle it
- Usually this step is only needed for really complicated codes

Stage 2: Pseudo-code

- Outlining the actual algorithm of the code in simple English
- Most pseudo-code can be translated into any language -- this enables programmers of all

languages to code your algorithm

Stage 3: Coding
- Actually implementing the code
- Often people jump to this stage and get bogged down in the language if they are inexperienced

Stage 4: Compiling
- The code is compacted into **byte code** by the compiler which will be interpreted by the computer
- Basically takes the java understandable to you and transforms it to java understandable to the computer
- **Compile-time** errors are caught at this point (mostly syntax errors). Your code will not compile until all the compile-time errors are resolved.

Stage 5: Running
- The code is interpreted by the JVM and is run by the computer
- **Runtime** errors occur here (bugs) and can sometimes crash the program
- If all goes well, your code is complete!

Downloading the Software

Java Development Kit
- Created by Sun Microsystems for java programming
- Always needed to run java on the computer
- If you are using an Apple computer, the second download will include the JDK, so you can skip this part.

1. Go to http://java.sun.com/javase/downloads/index.jsp
2. Click "Download JDK" on the first option. Be sure that it isn't JavaEE or anything else -- just the JDK.
   As of 2/15/10, the most updated version is JDK 6 Update 18
3. Select a platform and click download. If you are using an Apple computer, the platform you need is not listed. As I said, you can skip this part.

BlueJ Compiler
- This is the compiler that will check for errors and enable you to run the code.
- It also functions as an editor with features that will be convenient as you begin coding.

1. Go to http://www.bluej.org/download/download.html
2. Click the appropriate download link and follow the standard installation procedure.

The HelloWorld Breakdown

Friday, January 08, 2010
12:48 PM

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

Reading Code

- All programs start with the "main" line and run until they reach the bottom
- Languages are read like English -- Left to Right, Top to Bottom

Comment Styles

- A **comment** is a line of English that helps to explain the Java code so you don't forget it in the future
- **Single-Line** commenting
  If you just have one line of code you want to comment on with a small note, just end the line with 2 slashes followed by whatever you want to say. When the computer sees the two slashes, it jumps to the next line.
- **Block** commenting
  If you want to comment on a larger piece, like a description of what the code does as a whole, just start the comment with a slash and a star. The computer will jump to the end of the comment, which is noted by a star and slash.

Classes

- File where the java information will be kept (redefined in Java Course II)
- Everything in the class is embedded in a block
- Convention: class names should be CamelCase.

Main Method

- Where the program execution begins.
- Basically the beginning of the line is the "on" button of the program, and the end of the main - block is the "off" button where the program shuts down.
- Everything in the method is embedded in a block

Arguments and Parameters

- An **argument** is a statement or command you give to the computer
- Each argument ends with a ';'
- Some arguments take **parameters** as input so to execute -- for instance, the command "Print to the screen!" needs to have the information of what to print.

Printing and Printing Lines

- System.out.println("some string") takes a string of text as a parameter and prints it to the screen, and then jumps to the next line so the next time you print something it will be on a new line.
- System.out.print("some string") takes a string of text as a parameter and prints it to the screen, so the next time you print something it will be exactly where you left off.

Text Formatting Table

Character	Function
\\	Backslash printed
\"	Quotation mark printed

\n	Enter
\r	Carriage Return
\f	Form-Feed / Clear

Whitespace

- Java does not read whitespace
- Convention:
  - Blocks are tabbed inward
  - Right margin is maintained
  - Block commenting has text in lines between the /* and the */ rather than on the same line, unless it is a single-line comment that appears in the middle of the line so it can't use the //

Variable Introduction

Friday, January 08, 2010
1:10 PM

Declaration

DataType VariableName;

Declaration can only be done once -- If you say "I have variable X. I have variable X." the computer will get confused.

Assignment

VariableName = Value;

Note: The Mathematical = means "the two sides of the equation have equal values." However, the Computational = means "Take the value of the right side and store it in the variable on the left. The Algebraic = is denoted in Java as ==, but that comes later.

Initialization

Datatype VariableName = Value;

This is basically an assignment immediately at Declaration. Declaration says "I have a variable X." Assignment says "The variable X holds the value 3." Initialization says "I have a variable X that holds the value 3."

- This system ensures that no equation is performed that uses a variable with no value.
- Convention: initialize to 0 unless there is a particular reason to initialize to another number.
- Initialization can only be done once, because it is part declaration.

Naming Variables

1. Variable names cannot start with a number
2. Names cannot be symbols
3. Names cannot have spaces, but can have underscores
4. Names cannot be any of the java keywords (basically, if the word changes color, don't use it as a variable name!)
   Convention: variable names should be mixedCase.

Variable Manipulation

Operator	Function
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulus

Printing Variables

To print a variable, simply type the variable name inside the print function to print it. To attach it to a string, separate the string parameter and the variable with a + sign -- the function will automatically concatenate it into one string before printing.

Data Types

Saturday, January 09, 2010
9:14 AM

Differences

- A **floating point** number can hold decimal values and thus has more precision
- If it is holding an integer, it adds a .0 at the end of the number (important for printing purposes)
- Something of type "float" has the same capacity as type "int"
- Something of type "double" has the same capacity as type "long"
- Convention: Integers use type "int" and floats use type "double" -- unless absolutely necessary that another type be used.

Type Name	Kind of Value	Memory Used	Size Range
byte	integer	1 byte	–128 to 127
short	integer	2 bytes	–32768 to 32767
int	integer	4 bytes	–2147483648 *to* 2147483647
long	integer	8 bytes	–9223372036854775808 *to* 9223372036854775807
float	floating-point number	4 bytes	$\pm 3.40282347 \times 10^{+38}$ *to* $\pm 1.40239846 \times 10^{-45}$
double	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ *to* $\pm 4.94065645841246544 \times 10^{-324}$
char	single character (Unicode)	2 bytes	all Unicode characters
boolean	true or false	1 bit	not applicable

Display 2.2
Primitive Types

Division Distinction

- An integer divided by an integer drops the remainder in the answer, because the / operator only returns the quotient. So, 5/2 would return 2.
- When a floating point number is used in the division, the division is done with total precision and it returns a floating value. So, 5.0/2 would return 2.5.

Type Casting to Integers

- Setting an integer value to a double poses no problem, because precision is being increased
- Setting a double value to an integer is problematic, because precisions is being lost

Exercise: Create a program with the following main function:

```
double x = 10.56;
int y = (int)x;
System.out.println(y);
```

- **Type casting** truncates the end of a value and basically says to the computer "I know what I am doing, trust me on this one" and the computer will store the double in the integer. No rounding will be done -- the end will simply be cut.

Data Type: char

- Holds a single character

Data Type: boolean

- Holds a single true or false value

Constants

- In math, a constant is a numerical value that cannot be changed
- In computer science, a constant is a variable that is set in stone and cannot be changed in the program. Assigning a new value to a constant after it has been initialized will result in an error.
- A variable can be declared constant by using the keyword final before the initialization.

  final Datatype VARIABLE_NAME = Value;
- Convention: constants should be written in UPPER_CASE with underscores where spaces should be.

Importing Packages

Saturday, January 09, 2010
3:40 PM

The Java API

http://java.sun.com/j2se/1.5.0/docs/api/

- Java tools created by Sun Microsystems for programming
- Can be used in your code, but first you need to specify which **package** you will be using

Importing (specifying the package)

- On the right hand side of the API site there is a long list of programs people have written -- you need to specify which of those you want. For our intentions right now, scroll down to "Scanner".
- At the top, above where it says "Class Scanner," you should see the line "java.util" -- this is the package in which the class Scanner can be found
- To add a scanner to your code, above everything (except the header comment which specifies what your code will do), type "**import** java.util.Scanner;" -- this means "I would like to use the Scanner tool in the API package titled "java.util"

  Note: If you know which package you want but you will be using a lot of tools from it, you can just specify the package and then say "as many tools as I need" -- this is done by placing an asterisk in the place where you identify the tool name: "import java.util.*;"

The Scanner Tool

- A scanner is a tool created by Sun Microsystems that allows the programmer to request input from the user
- At the top you specified that you will be using a Scanner. You first need to place the scanner into your code. This is done quite similarly to the variable declaration we studied before:
  ObjectType ObjectName = new ObjectType(necessary parameters);
  Scanner input=new Scanner(System.in); //System.in specifies that the source will be the keyboard input.
- Go to the Method Summary section to see what a scanner can do
- Scanner = an object that scans in data from a stream of input
- nextInt() and nextDouble() methods are **called** by the Scanner object, so the notation ObjectName.ObjectMethod(), or in this case input.nextInt() is used.
- nextInt() and its sister methods all retrieve the first "word" inputted by the user -- all the characters up to the space or enter. If a user enters "5 4", the first nextInt() would retrieve the 5, and the next call would automatically retrieve the 4 instead of waiting for the user to input another value. nextLine() clears everything in the waiting queue.

The Math Tool

- Imported by default
- Contains many useful functions

Function	Usage
random()	Returns a random double between 0 (inclusive) and 1 (exclusive). Useful Formula: **((int)(Math.random()*(range of possible integers)))+(minimum value)** Example: (int)(Math.random()*10)+1 -->returns a random number between 1 and 10
pow(x,y)	Returns x raised to the y as a double
sqrt(x)	Returns the square root of x as a double
abs(x)	Returns the absolute value of x as a double
floor(x)	Returns the greatest integer less than or equal to x as a double
ceil(x)	Returns the least integer greater than or equal to x as a double
round(x)	Returns the closest **long** to x if x is of type **double**, and returns the closest **int** to x if x is of type **float**. Note that you can typecast the returned long into an int.

Basic Conditions

Sunday, January 10, 2010
1:26 PM

Boolean Data Type

- Holds either the value true or false
- Basically contains a bit of information

Flip-Flop Gates

- Binary 1s and 0s are really translated into a series of on-off switches that send signals through the wires of a chip
- Different gates can redirect those signals and flip a 1 to a 0 or vice versa
- A 0 denotes a false value while a 1 denotes a true value

Gate	Return Value
And	Returns true only if both values are true
Or	Returns true if at least one value is true
Not	Returns the opposite of the inputted value
Nand	Returns true if at least one value is false
Nor	Returns true only if both values are false
Xor	Returns true only if only one of them is true
Xnor	Returns true only if the two are the same

Branching

- Depending on a condition, the code could take different turns.
- Ultimately the code will run to the bottom -- the branched paths inevitably lead to the same end

Scanner Function: nextBoolean()

- input must either true or false (lowercase, no quotation marks)

If Statement

```
if (condition)
    Do something;
```

Note: If there is more than one command under the condition, place them in a block.
Warning: A semicolon after the condition line will not result in the condition having any effect. Make sure that the line ends after the command, not after the condition.

Block Rule

- An if-statement with only one argument after it needs no block. However, if multiple arguments fall under the if-statement, a defined block is needed.

Comparative Operators

- Expressions can be evaluated to true or false

Operator	Test
==	Are the values equal?
!=	Are the values unequal?
>	Is the first value greater than the second?
>=	Is the first value greater than or equal to the second? **Note: Do not use =>**
<	Is the first value less than the second?

<=	Is the first value less than or equal to the second? **Note: Do not use =<**

If-Else Clause

- If the condition is met, the commands in its block are executed
- If the condition is not met, the commands in the else block are executed

Else if

- Useful for stringing multiple conditions together

Advanced Conditions

Sunday, January 10, 2010
4:17 PM

Conjoined Conditions

- **Unary Operator**: !
- **Bitwise Binary Operators**: &, |, ^ (not commonly used)
- **Short-Circuit Binary Operators**: && and | |

Gates Rewritten:

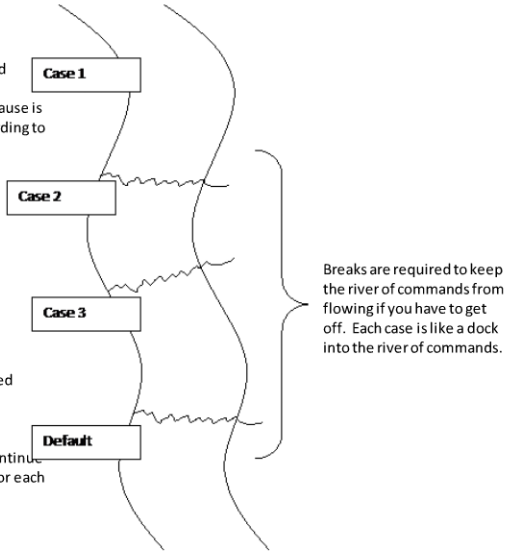
Gate	Boolean Expression
And	x && y
Or	x | | y
Not	!x
Nand	!(x && y)
Nor	!(x | | y)
Xor	((x && (!y)) | | (y && (!x)) Can also be expressed as x^y
Xnor	(((!x) && (!y)) | | (x && y)) Can also be expressed as !(x^y)

Order Precedence

Precedence	Operator	Type
12	()	Parenthesis
11	(Type), !, ~	Typecast, Logical NOT, Bitwise NOT
10	*, /, %	Multiplication, Division, and Modulus
9	+, -	Addition and Subtraction
8	<, <=, >, >=	Relations
7	==, !=	Equality
6	&	Bitwise AND
5	^	Bitwise XOR
4	|	Bitwise OR
3	&&	Logical AND
2	| |	Logical OR
1	=	Assignment

Cascading If-Statements

- Branching trees can be constructed
- Inside each if-statement you can place more if-statements as needed
- Enables a clear organization of the code instead of a string of if-statements with convoluted conjoined conditions in a single "if - else if - else" chain
- If-statements can be cascaded without brackets if they are single lines -- an entire if-else clause is treated as a single argument, so if only an if-else clause is inside an if-statement then according to the block rule no braces are needed.



Logical Contrapositive

- "If a then b" is the logical equivalent to "If not be then not a"
- Used in evaluating logical deductions

Web Searching Techniques

- AND, OR, NOT, NEAR
- Parenthesis and conjoined conditions apply
- +require, -exclude

Switch Statements

- Used as another method of chaining a series of if-statements together that can't be cascaded
- Breaks down the possible conditions with a series of cases in a single block
- Uses a default case for any situation that falls outside of the specified cases
- Every time the case is completed, the program must **break** from the switch
- If no break is executed, the computer will simply execute the tasks for the next case and continue all the way down until it sees a break. Basically you need to switch on and off the current for each case instance (unless you intentionally want it to run all the way through).
- A switch can only be used with integers and characters

```
switch (x)
{
    case 1: Do something; break;
    case 2: Do something; break;
    default: Do something;
}
```

Looping

Monday, January 11, 2010
10:39 AM

Concept

- Repeated action until a condition becomes false
- Does not reverse Top-Down design, but returns the program to a previous line, often with new parameters.

The While Loop

```
while (condition)
{
    execute…
}
```

Note: Do not use the = sign in the while loop! It sets the value of the variable rather than checking it

The Do-While Loop

```
do
{
    execute…
}
while (condition);
```

Note: the do-while ALWAYS executes once, and it has a semicolon after the condition

The For Loop

```
for (initialization; condition; update)
{
    execute…
}
```

Note: the for loop is often used when you know when you are going to terminate the loop -- you know the number of times you want it to cycle.

Nested Loops

- Same idea as cascading if-statements
- Often convenient for dual-counting, like with rows and columns

Validating Input

- Continues to loop until input meets preconditions
- System.err --> Error Message Output
- System.exit(0) terminates the program
- Using the hasNextDataType() method set, we can determine if a certain data type can be read before it actually is read by the nextDataType() method set, avoiding potential crashes
- If the input is incorrect, it will still be there, so nextLine() will have to be called to flush the queue each time, or else you will end up in an infinite loop.
- An alternative method can be found here

Increments and Decrements

Assignment Shorthand

- +=, -=, *=, /=, %=
- Example: x += y is the same as x = x + y

Pre-Increment Operators

- ++x, --x
- Increments first, evaluates second

Post-Increment Operators

- x++, x--
- Evaluates first, increments second

Break and Continue

- Break keyword can also be used to break from a loop without checking the condition
- Continue keyword terminates this iteration of the loop and jumps back to the condition to evaluate it
- Both keywords work, but they are considered inelegant and "noobish"

Sentinel Values

- If you want the user to enter a series of values of unknown length, i.e. if you want to find the sum of every value entered, you need a **sentinel value** that will signal the program to stop looping
- Often the sentinel value is -1, 0, or a ridiculously large number.

Infinite Loops and Computer Crashing

- A loop that has a condition that is never violated will run endlessly forever
- Condition "true" is acceptable -- this will inevitably cause an infinite loop without a break
- An infinite loop that runs too long or a loop that runs to a ridiculous number can consume all the RAM if left untouched and can make the computer run insanely slow
- The computer will be back to normal after it is shut down

    Note: If you programmed it in C, this stuff could really do serious damage -->this is the advantage of java programming -- it is far more secure

Debugging

- Compile time = Time during code writing -- Syntax errors are caught
- Runtime = Time during code execution -- Bugs are caught
- Planting stops
- Stepping and Continuing with iterations

TI Programming

- Programming on the TI-Calculators can be done with current skills
- Look at syntax online -- same coding principles apply

Characters and Strings

Monday, January 11, 2010
11:36 AM

New Data Type

- char --> Holds a single character
- Characters include enter, shift, space, etc.
- Characters are denoted by 'single quotes'

New Object: String

- A collection of characters
- Characters are like beads and a String is the thread that joins them together
- Strings are denoted by "double quotes"

Scanner Function: nextLine()

- Takes the entire line of input, including the "\n," as input
- If you used a scanner method before this one, you need to call nextLine() twice. This will be further explained in a later section.

toUpperCase() and toLowerCase() methods

- Converts the entire string to either upper or lower case

charAt(), indexOf(), and length() methods

- These two methods are "inverses" of each other
- **charAt(int index)** Returns the character at the specified index
  Note: Characters on a string begin with index 0
- **indexOf(character c)** Returns the index of the first occurrence of the specified character
  Note: Returns -1 if the character is not present
- **length()** Returns the length of this string.

substring() and concat() methods

- **substring(int start, int end)** Returns the string of characters from the start index (inclusive) to the end index (exclusive)
- **concat(String s)** Attaches string s to the end of the string that called the method
- Just like in the println() method, you can also concatenate strings simply with the + operator

Looping Concatenation

- Concatenation is particularly useful when adding to a string over and over through a loop.
- Kind of used like the integer incrementing except with a string

equals() and equalsIgnoreCase() methods

- Compares the content of one string to another and returns true if they are the same
- IgnoreCase version disregards case differences, so "yes" and "Yes" would still be equal

compareTo() method

- Used for alphabetizing the strings
- If the calling string comes before the parameter string, it returns a negative value
- If the parameter string comes before the calling string, it returns a positive value
- If the strings are equal, it returns a value of 0

Parsing Values

Integer.parseInt(str) grabs the next integer from the string by **parsing** the string into two components -- the integer and the remainder. This method will fail if the next piece of the string is not an integer.

Note: This method will be explained more in-depth in Java Course II.

Null

- Just like "nothingness" for integers is 0, "nothingness" for an object is **null**

Null Pointer Exception

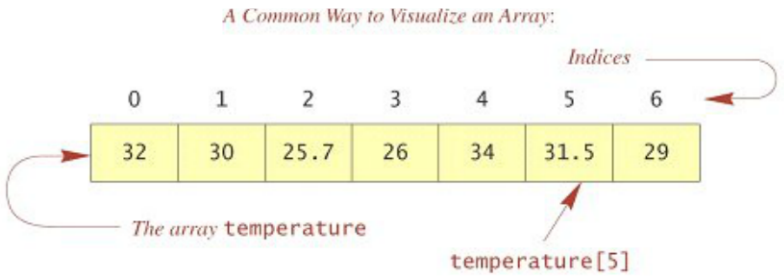
- An **Exception** is an error message that is caused by a situation outside the norm
- No "has not been initialized" error will be caught by the computer during compile time, so if you try to manipulate the string with a value of null the program will crash.
- Known as an infamous error message that frustrates **everyone**

Arrays

Monday, January 11, 2010
6:29 PM

Definition and Usefulness of an Array

- Arrays are an efficient tool for storing information that would need to be stored in a series of variables in an otherwise inefficient and confusing way.
- It is basically a set of **elements** stored together in a linear formation according to **index**
  - Imagine an answer sheet -- you have a question number (**index**) and an answer (element value)
- Arrays are particularly useful when dealing with long lists of data, such as grades for a student



Redefinition of Strings and charAt() method

- A string is a list of characters sorted by indices, no? Strings are indeed arrays of characters.
- You can also explicitly make an array of characters, but strings are easier to manipulate for most purposes.
- **charAt(int index)** basically a method that finds the element with the specified index.
- Arrays, like strings, begin at index 0

Declaration

```
DataType[] name = new DataType[ArraySize];
DataType name[] = new DataType[ArraySize];
```

Assignment

- To access a value, use the syntax ***name[index]*** in place of where you would use a variable name.

Initialization

- Values in an array are all empty values that need to be initialized
- Arrays can be initialized with curly braces containing the contents after the declaration statement:

```
DataType[] name = new DataType[ArraySize] {element0, element1, element2, etc};
```

Arrays & Loops Combo

- Since arrays use a counter to access the data, it is easy to perform functions with them such as initialization of values or adding the values together with a for loop
- Counter for the for loop functions as an index counter

String[] args

- Another look at the main method should give us understanding of what *String[] args* represents.
- An array of arguments can be passed into the method as strings.

Typecasting Array Contents

Syntax: (DataType[])arr;

Shortcomings of Arrays

- Size is immutable -- does not shrink or expand
- Values cannot be removed or added, only changed.

Array Index Out Of Bounds Exception

- If you try to retrieve a value from the array from a non existent index (a negative one or one that exceeds the array size), the program will crash because that index does not exist.
- Known as an infamous error message that frustrates everyone

Matrices and the XY Plane

- Arrays can have multiple dimensions
- To add a new dimension, simply add another bracket set in the array usage:

  DataType[][] name = new DataType[first dimension size][second dimension size];
- Usually the first bracket represents the row, and the second bracket represents the column
- You can add array dimensions to an array even if you can't map it (like a 4D array).

  Warning: the memory space consumed is the volume of the figure, so the size of every measurement has to be multiplied together.
- In order to properly use methods of arrays, such as the .length property, treat the matrix as an "array of arrays." In this sense:

  arr.length returns the length of the first dimension

  arr[0].length returns the length of the second dimension

Method Decomposition

Tuesday, January 12, 2010
10:50 AM

Uses of Decomposition

- Breaks down a long code into distinct sections
- Allows a piece of code to be used several times without writing it over and over again
- Makes the code easier to comment, follow, and read
- Has other functions later that will be covered in Java Course II

Method Heading / Signature

propertyList returnType name (parameterList)

- Methods are like number machines
- Can be thought of as f(x) in Algebra 2
- Parameter list a list of parameters that the method requires to use
- These parameters are known as **Formal Parameters**, are placeholders for values they will assume
- Return type is what kind of value leaves the function
- Convention: methods should be written in mixedCase.

Method Body

- Looks similar to what goes inside the main method
- Basically defines what occurs inside the method / machine
- Whatever manipulation occurs to the parameters or the code is executed here
- The method body is embedded within a block directly beneath the signature

Calling a Method

methodName(parameters)

- Parameters being passed in do not need the type in front of them to identify them
- These are the actual parameters that are passed to the formal parameters

Query Methods and Returning Values

- A **Query Method** is basically an investigating method that will determine what a value is, whether something is true, etc.
- Example: "isPrime" "isPositive"
- To return an answer, use the **return** keyword followed by the value you wish to return.

Action Methods and Returning Void

- An Action Method is basically a method that will perform a task it is given
- Functions like a soldier -- It executes a task that it is told to do.
- Returns a **void** value, which is basically an empty return.
- To return void, simply use the return keyword without any value after it. This is an **explicit return**.
- You can also not return anything and just let the code finish the method and then by default it will return void when it reaches the bottom. This is an **implicit return**.

> *Note: Proper coding breaks all methods down into action and query methods, but a method that does both is still allowed. This is much like the "break" and "continue" keyword usage -- it is considered "noobish" but it is technically ok.*

Variable Lifespan

Demonstration: Use a formal and actual parameter of X and change the value of X within the function. Does the actual parameter change?

Demonstration 2: Use a formal and actual parameter of X[] and change the value of X[0] within the function. Does the actual X[0] change?

- **Local variables** (declared inside blocks or methods) only last until the block or method reaches its end
- **Field variables** (declared outside all blocks and methods) can be used for the entire class and can be accessed by all methods.

- Array difference will be explained more in-depth in **Java Course II**

Non-Static Variable X cannot be referenced from a static context

- You are bound to come across this error message as you begin to write your own methods
- It means you forgot the static keyword before the function name
- Always include static until you learn how exactly to use it in Java Course II

Definition of Recursion

- A repeated action that happens again and again until a condition is met to stop it
- The process is executed on a different value each time
- Kind of like a jackhammer which continues pulverizing each piece until it becomes dust.

Recursion v. Loops

- Any looping action can be done with recursion, and vice versa
- Recursion is considered more elegant than a loop, but it is not "noobish" to use a loop.
- Recursion is slower than looping, but it is an important skill because it often makes loops make sense

Recursive Stacks and Overflowing

- A recursive function can be broken down into windows in a **stack**
- Arrow climbs upward as windows open and then descends as windows close
  If too many windows open up and it consumes too much memory the program will crash because of a **stack overflow**. This is known as **memory leakage**.

  Example of Stack Overflow:

```
public boolean willFail()
{
    return willFail();
}
```

Advanced Commenting

Tuesday, January 12, 2010
1:39 PM

Preconditions

- Requirements before a method can be executed
- The method does not check these preconditions, but it states that they must be met before the method is run.

Postconditions

- Description of what will have changed after the method is run
- Basically describes what the method does

Java Documentation (Javadoc)

- Creates a window like the kinds on the Java API for your code
- Makes the code easier to understand and use

Javadoc Tag List

* @author (classes and interfaces only)
* @version (classes and interfaces only)
* @param (methods and constructors only)
* @return (methods only)
* @see (creates a see also section with a link)
* @deprecated (indicates a deprecated method)

Note: If with javadoc the method's purpose and usage is very clear, preconditions and postconditions do not need to be specified. However, if the method is an action method rather than a query one, usually preconditions and postconditions should be present. Use your best judgment.

Dealing with Exceptions

Tuesday, January 12, 2010
1:48 PM

Making your Program Robust

- Anticipate any error that could possibly exist and circumvent it with awesome coding
  Exercise: Write a program that asks for 2 integers to divide and you enter a divisor of 0?
- This can be prevented by making your program robust

Try-Catch Clause

- The **try clause** performs the code. If everything goes correctly, the try clause is executed and the code moves on. If an error occurs, instead of crashing the code jumps to the **catch clause** and executes the code there.
- Should not be used to cover for your own errors that you don't understand.
- Prevents **runtime** errors

Validating Input (continued)

- If you loop a try-catch statement until a correct value is inputted you can ensure that the program does not crash and that you have a value that works before the loop is terminated
- This is an alternative to the hasNextDataType() methods.

Finally Clause and its Tricks

- The finally clause is a piece of code which will be executed regardless of whether the code failed.
- If a method returns a value in the try or catch block, the finally clause is executed before the value is actually returned. Thus, if the value is affected in the finally clause, the return ***will*** be affected.

Dodging the Error

- If you don't want your program to crash when something goes wrong but you realize that there are potential errors, you can have part of your program dodge the error and have another piece of code actually catch it (kind of like in tennis).
- If all the code pieces dodge the error, the program will crash (both players watch the ball go out of bounds because neither goes after it).
  [method signature] throws Exception
  Kind of appropriate that an error that is not caught is thrown, no?
- A method that throws an exception should have the exception it throws denoted in the javadoc with the @throws or @exception tag. Both do the same thing.