Overview of Source Code

- Components
  - Comments
  - Library declaration
- Classes
  - Functions
  - Variables
Comments

- Can be anywhere in the source code
- A compiler ignores the block
- Three kinds of comments:
  - /* text */
    - The compiler ignores everything from /* to */
  - /** documentation */
    - This indicates a documentation comment
  - // text
    - The compiler ignores everything from // to the end of the line
Generate Java code documentation in HTML format from Java source code

```java
/**
 * The HelloWorld program implements an application that simply displays "Hello World!" to the standard output.
 * 
 * @author Mingon Kang
 * @version 1.0
 * @since 2015-08-01
 */
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

http://www.tutorialspoint.com/java/java_documentation.htm
Library declaration

- Declare libraries that need to load in the program
  - Set of dynamically loadable libraries
  - User-defined library

```java
import package_name

// e.g.
import java.io.*;
import com.sun.*;
```

http://docs.oracle.com/javase/7/docs/api/
Identifiers

- The name of a variable or other items such as class, method, object defined in a program.
  - Unlimited-length sequence of letters and digits
  - Begin with a letter, the dollar sign "$", or the underscore character "_". No digit
  - Subsequent characters may be letters and digits, dollar signs, or underscore characters
  - Neither commas nor blanks are not permitted.
  - No special symbols other than underscore are allowed in the variable name
Identifiers

- The name of a variable or other items such as class, method, object defined in a program.
  - A variable name is **case sensitive**.
    - E.g., `name` vs `Name` vs `NAME`
  - Cannot be the same as **keywords nor reserved words**
    - E.g., `int`, `if`, `for`
Variable Naming

Questions!!

- name
- _name
- _____name
- @counter
- 1top
- _asdfadfadfaff
- csci431#2pm
- MY VARIABLE IS CORRECT
- its-a-variable
### Keywords and Reserved words

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

- Class names with an uppercase letter
  - `FirstProgram` `HelloWorld` `String`
- Variable names with a lowercase letter
- Hungarian Notation
  - Use prefix to indicate the data type of the variable

Variables

- A name used to refer to a certain location in memory
- Requires that variables must be declared with the variable type before it is used.
## Primitive Types

**Display 1.2** **Primitive Types**

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Kind of Value</th>
<th>Memory Used</th>
<th>Size Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true or false</td>
<td>1 byte</td>
<td>not applicable</td>
</tr>
<tr>
<td>char</td>
<td>single character (Unicode)</td>
<td>2 bytes</td>
<td>all Unicode characters</td>
</tr>
<tr>
<td>byte</td>
<td>integer</td>
<td>1 byte</td>
<td>−128 to 127</td>
</tr>
<tr>
<td>short</td>
<td>integer</td>
<td>2 bytes</td>
<td>−32768 to 32767</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
<td>4 bytes</td>
<td>−2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>integer</td>
<td>8 bytes</td>
<td>−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td>float</td>
<td>floating-point number</td>
<td>4 bytes</td>
<td>−3.40282347 × 10^{38} to −1.40239846 × 10^{−45}</td>
</tr>
<tr>
<td>double</td>
<td>floating-point number</td>
<td>8 bytes</td>
<td>±1.76769313486231570 × 10^{308} to ±4.94065645841246544 × 10^{−324}</td>
</tr>
</tbody>
</table>
Fundamentals of Computer Systems

- **Bit**: zero or one
  - Morse Code, Punched Card, Electrical Switch, Two distinct Voltage or current levels

- **1 Byte = 8 bits**
  - Historically for a single character
  - A unit to access a memory
  - Data can be accessed 8 bits or multiples of 8 bits at a time
  - `char` in Java takes 2 bytes for Unicode

- **16 bit vs 32 bits vs 64 bits computers?**
  - Processor registers (small amount of storage in a CPU)
    - Data, Address, General purpose, Conditional, and so on..
Data representation

- **Integer (in C/C++)**
  - 2 bytes in 16 bits systems
  - 4 bytes in 32/64 bits systems

- **Unsigned integer**: $0 \sim 2^n - 1$
- **Signed integer (two's complement)**: $-2^{n-1} \sim 2^{n-1} - 1$

Two's complement: [http://en.wikipedia.org/wiki/Two%27s_complement](http://en.wikipedia.org/wiki/Two%27s_complement)
### Data representation

- **Character (1 byte in C/C++)**
  - **ASCII code** (American Standard Code for Information Interchange)

<table>
<thead>
<tr>
<th>DEC</th>
<th>OCT</th>
<th>HEX</th>
<th>BIN</th>
<th>Symbol</th>
<th>HTML Number</th>
<th>HTML Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>040</td>
<td>20</td>
<td>00100000</td>
<td></td>
<td>&amp;#32;</td>
<td></td>
<td>Space</td>
</tr>
<tr>
<td>33</td>
<td>041</td>
<td>21</td>
<td>00100001</td>
<td>!</td>
<td>&amp;#33;</td>
<td></td>
<td>Exclamation mark</td>
</tr>
<tr>
<td>35</td>
<td>043</td>
<td>23</td>
<td>00100011</td>
<td>#</td>
<td>&amp;#35;</td>
<td></td>
<td>Number</td>
</tr>
<tr>
<td>36</td>
<td>044</td>
<td>24</td>
<td>00100100</td>
<td>$</td>
<td>&amp;#36;</td>
<td></td>
<td>Dollar</td>
</tr>
<tr>
<td>37</td>
<td>045</td>
<td>25</td>
<td>00100101</td>
<td>%</td>
<td>&amp;#37;</td>
<td></td>
<td>Procenttecken</td>
</tr>
</tbody>
</table>

ASCII code Table: [http://www.ascii-code.com/](http://www.ascii-code.com/)
## Data representation

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>060</td>
<td>30</td>
<td>00110000</td>
<td>0</td>
<td>0</td>
<td>Zero</td>
</tr>
<tr>
<td>49</td>
<td>061</td>
<td>31</td>
<td>00110001</td>
<td>1</td>
<td>1</td>
<td>One</td>
</tr>
<tr>
<td>50</td>
<td>062</td>
<td>32</td>
<td>00110010</td>
<td>2</td>
<td>2</td>
<td>Two</td>
</tr>
<tr>
<td>51</td>
<td>063</td>
<td>33</td>
<td>00110011</td>
<td>3</td>
<td>3</td>
<td>Three</td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>101</td>
<td>41</td>
<td>01000001</td>
<td>A</td>
<td>A</td>
<td>Uppercase A</td>
</tr>
<tr>
<td>66</td>
<td>102</td>
<td>42</td>
<td>01000010</td>
<td>B</td>
<td>B</td>
<td>Uppercase B</td>
</tr>
<tr>
<td>97</td>
<td>141</td>
<td>61</td>
<td>01100001</td>
<td>a</td>
<td>&amp;#97</td>
<td>Lowercase a</td>
</tr>
<tr>
<td>98</td>
<td>142</td>
<td>62</td>
<td>01100010</td>
<td>b</td>
<td>b</td>
<td>Lowercase b</td>
</tr>
</tbody>
</table>
Unicode

- Unicode (2 bytes)
  - A character set used by the Java language that includes all the ASCII characters plus many of the characters used in languages with a different alphabet from English
  - Express most of the world's writing systems

http://unicode-table.com/en/#control-character
Data representation

- **Float: single-precision floating-point format**
  - A bit for sign (0: positive, 1: negative)
  - 8 bits for exponent
  - 23 bits for fraction

Data representation

- **Double**: double-precision floating-point format

  ![Double-precision floating-point format diagram](http://en.wikipedia.org/wiki/Double-precision_floating-point_format)
System.out

- System.out.print() and System.out.println() functions

```java
public static void main(String[] args) {
    int i = 2;
    System.out.print("This is a message");
    System.out.print(i);
    i = 5;
    System.out.println("The square root of " + i + ".");
}
```
System.out

- System.out.format() functions
  - Output the formatted data

```java
public static void main(String[] args)
{
    int i = 2;
    System.out.format("The square root of %d\n", i);
}
```

https://docs.oracle.com/javase/tutorial/essential/io/formatting.html
Statement

- Smallest standalone element of programming language
- Statements control the flow of program execution.
  - Declaration statements
  - Assignment statements
  - Function Call statements
  - Compound statements
  - IF statements
  - FOR statements
  - WHILE/DO-WHILE statements
  - SWITCH statements
Every statement ends with SEMICOLON ";"

System.out.print("CSCI 431 Java Programming");
;
// null statement

Multiple statements can be in a single line

int a; int b;

Codes are processed from the top down
Variables vs Constants

- Variables
- Constants
Variable Declaration

- Reserve a space in memory
- Name that refers to the memory space
- All variables must be declared before

```
DATA_TYPE VARIABLE_NAME;
DATA_TYPE VARIABLE_NAME = INITIAL_VALUE;
e.g.,
    int counter;
    int counter = 1;

DATA_TYPE VAR1[=VALUE1],..., VAR3[=VALUE3];
e.g.,
    int counter1, count2, count3=1;
```
Assignment Compatibility

- In general, the value of one type cannot be stored in a variable of another type
  ```java
  int intVariable = 2.99; //Illegal
  ```
  - The above example results in a type mismatch because a `double` value cannot be stored in an `int` variable
- However, there are exceptions to this
  ```java
  double doubleVariable = 2;
  ```
  - For example, an `int` value can be stored in a `double` type
Assignment Compatibility

- More generally, a value of any type in the following list can be assigned to a variable of any type that appears to the right of it:
  - `byte` → `short` → `int` → `long` → `float` → `double` → `char`  

- An explicit type cast is required to assign a value of one type to a variable whose type appears to the left of it on the above list (e.g., `double` to `int`).

- Note that in Java an `int` cannot be assigned to a variable of type `boolean`, nor can a `boolean` be assigned to a variable of type `int`. 

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.
**Constants**

- **Constant (or literal):** An item in Java which has one specific value that cannot change
  - Constants of an integer type may not be written with a decimal point (e.g., 10)
  - Constants of a floating-point type can be written in ordinary decimal fraction form (e.g., 367000.0 or 0.000589)
  - Constant of a floating-point type can also be written in scientific (or floating-point) notation (e.g., 3.67e5 or 5.89e-4)
Constants

- Constants of type `char` are expressed by placing a single character in single quotes (e.g., 'Z')
- Constants for strings of characters are enclosed by double quotes (e.g., "Welcome to Java")
- There are only two `boolean` type constants, `true` and `false`
  - Note that they must be spelled with all lowercase letters
- Use ‘final’ keyword
  - final int MAX = 10;
  - final String WELCOME_MESSAGE = “Welcome to Java”;
Variable Assignment

- **Example 1**
  ```java
  final int A = 4;
  final int B = 3;
  final int C = 2;
  int nGrade;
  nGrade = A
  // the value of “nGrade”
  will be A (=4)
  ```
Variable Assignment

Example 2

```c
int a = 3;  // declare a variable “a” with the initial value of 3
int b = 4;  // declare a variable “b” with the initial value of 4
a = b;    // the value of the variable “a” is now 4
// Note that it does not mean a is equal to b
// “==“ is a comparison operator, e.g., a == b
```
Expression

- A single entity (usually a number), such as a constant or variable, OR some combination of such entities

- Examples
  - 5
  - x
  - a + b
  - x++
  - X <= y
  - x == y
Expression

- Involve multiple operands and operators
  - \( A + 4 \)
  - **Operand**: variables or constants
    - (\( A \) and 4 are both operands of the \(+\) operator)
  - **Operator**
    - Arithmetic
    - Assignment
    - Logical/relational
    - Bitwise
Arithmetic Operators

- Arithmetic operators
  - +: plus
  - -: minus
  - / : division
  - * : multiplication
  - % : modulo (remainder of after an integer division)
  - -- : decrement
  - ++ : increment
Arithmetic Operators

- **Examples**
  - $1 + 3 = 4 - 1$
  - $2 \times 4 = 4 / 2 = 5 / 0$ (you cannot divide by zero)
  - $3 \% 2 = 4 \% 1 = 5 \% 0$ (X)
  - $a++ = ++a$
  - `int a = 1;
    System.out.print("a is "+ a++);
    System.out.print("a is "+ ++a);`
  - `b-- = --b`
Assignment Operator

- Assignment Operator
  - =
  - +=
  - -=
  - *=
  - /=
  - %=
Assignment Operator

Examples

- a = 1;
- a += 1;
- a -= 1;
- a *= 1;
- a /= 1;
- a %= 1;
Logical/Relational

- `==` Equal to (Don’t confuse with the assignment “==“)
- `!=` Not equal to
- `>`
- `<`
- `>=` => gives an error
- `<=` =<= gives an error
- `&&` Logical AND; ‘&’ is a bitwise operator
- `||` Logical OR; ‘|’ is a bitwise operator
- `!` Logical NOT
Type Casting

- Type casting is to convert a variable from one data type to another data type.

- For instance, if you want to store a long value into a simple integer then you can type cast long to int explicitly using the cast operator as follows:

  \[(\text{type\_name})\text{ expression}\]

E.g. \((\text{int}) 13.2\)

\((\text{float}) 11/5\)
# Operator Precedence

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp++ exp--</td>
<td>Suffix/postfix increment and decrement</td>
<td>Left-to-right</td>
</tr>
<tr>
<td>++exp --exp</td>
<td>Prefix increment and decrement</td>
<td>Right-to-left</td>
</tr>
<tr>
<td>+ -</td>
<td>Unary plus and minus</td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>Logical NOT</td>
<td></td>
</tr>
<tr>
<td>(type_cast)</td>
<td>Type cast</td>
<td></td>
</tr>
<tr>
<td>new</td>
<td>Object creation</td>
<td></td>
</tr>
<tr>
<td>* / %</td>
<td>Multiplication, division, and remainder</td>
<td>Left-to-right</td>
</tr>
<tr>
<td>+ -</td>
<td>Addition and subtraction</td>
<td></td>
</tr>
<tr>
<td>&lt; &lt;=</td>
<td>For relational operators &lt; and &lt;=</td>
<td></td>
</tr>
<tr>
<td>&gt; &gt;=</td>
<td>For relational operators &gt; and &gt;=</td>
<td></td>
</tr>
</tbody>
</table>
## Operator Precedence

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>6  ==  !=</td>
<td>For relational = and ≠</td>
<td>Left-to-right</td>
</tr>
<tr>
<td>7  &amp;&amp;</td>
<td>Logical AND</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9  =</td>
<td>Simple assignment</td>
<td>Right-to-left</td>
</tr>
<tr>
<td>+=  -=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*= /= %=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operator Precedence

Example 1)

```c
int a, b, c = 1;

a = b = c;
```

1) `(a = b) = c;`

2) `a = ( b = c);`
Operator Precedence

Example 2)

```plaintext
int a = 1, b = -1;
!a + b

1) (!a) + b
2) !(a + b)
```
Operator Precedence

Example 3)

```c
int a = 1, b = 4;
a = a+++b;
1) (a++) + (b)
2) (a) + (++b)
```