

Introduction To Java Programming

You will learn about the process of creating Java programs and constructs for input, output, branching, looping, as well some of the history behind Java's development.

James Tam

Java Vs. Java Script

Java (this is what you need to know for this course)

- A complete programming language developed by Sun
- Can be used to develop either web based or stand-alone software
- Many pre-created code libraries available
- For more complex and powerful programs

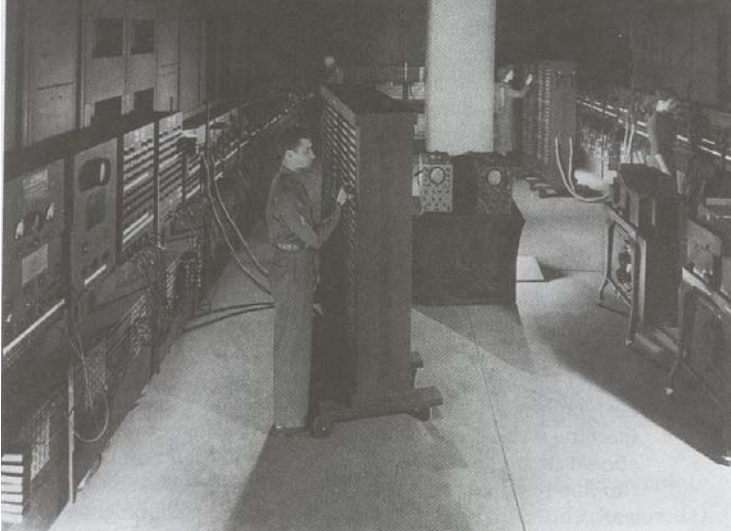
Java Script (not covered in this course)

- A small language that's mostly used for web-based applications (run through a web browser like Internet Explorer, Firefox, Safari, Chrome)
- Good for programming simple special effects for your web page e.g., roll-overs
- e.g.,
<http://pages.cpsc.ucalgary.ca/~tamj/2005/231P/assignments/assignment4/index.html>

James Tam

Java: History

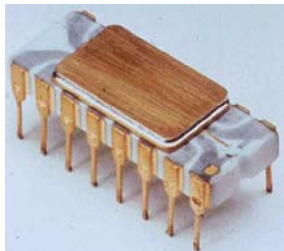
- Computers of the past



James Tam

Java: History (2)

- The invention of the microprocessor revolutionized computers



Intel microprocessor



Commodore Pet microcomputer

James Tam

Java: History (3)

- It was believed that the logical next step for microprocessors was to have them run intelligent consumer electronics



James Tam

Java History (4)

- Sun Microsystems funded an internal research project “Green” to investigate this opportunity.
 - Result: A programming language called “Oak”



Blatant advertisement: James Gosling was a graduate of the U of C Computer Science program.

Java History (5)

- Problem: There was already a programming language called Oak.
- The “Green” team met at a local coffee shop to come up with another name...
 - Java!



James Tam

Java: History (6)

- The concept of intelligent devices didn't catch on.
- Project Green and work on the Java language was nearly canceled.



James Tam

Java: History (7)

- The popularity of the Internet resulted in Sun's re-focusing of Java on computers.
- Prior to the advent of Java, web pages allowed you to download only text and images.

Your computer at home running a web browser



User clicks on a link



Images and text get downloaded

Server containing a web page



James Tam

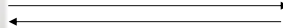
Java: History (8)

- Java enabled web browsers allowed for the downloading of programs (Applets).
- Java is still used in this context today:
 - Facebook

Your computer at home running a web browser



User clicks on a link



Java Applet downloaded

Server containing a web page



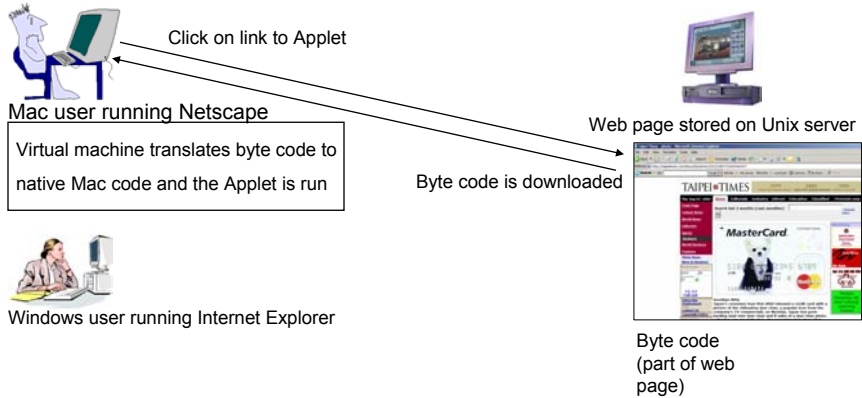
Java version of the Game of Life: <http://www.bitstorm.org/gameoflife/>

Online checkers: <http://www.darkfish.com/checkers/index.html>

James Tam

Java: Write Once, Run Anywhere

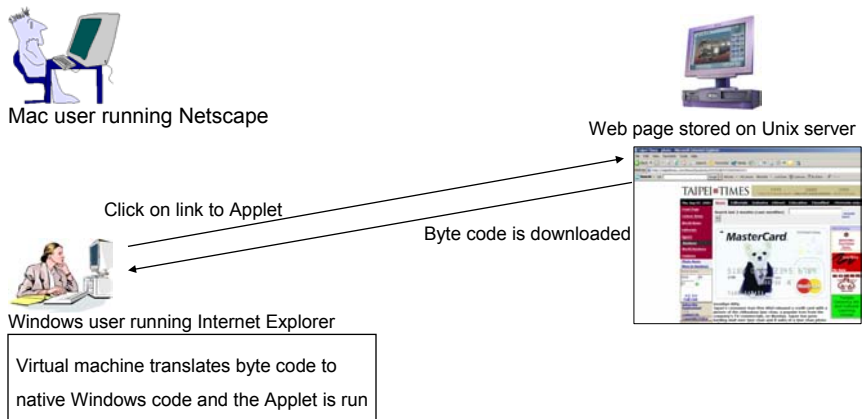
- Consequence of Java's history:
platform-independence



James Tam

Java: Write Once, Run Anywhere

- Consequence of Java's history:
platform-independent



James Tam

Java: Write Once, Run Anywhere (2)

- But Java can also create standard (non-web based) programs



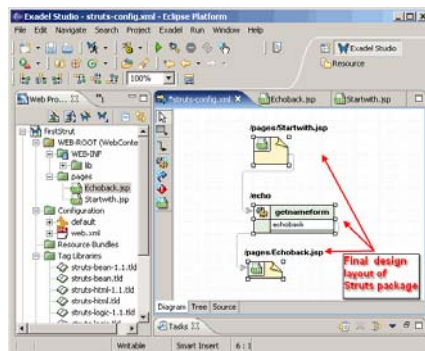
Dungeon Master (Java version)

<http://homepage.mac.com/aberfield/dmi/>

James Tam

Java: Write Once, Run Anywhere (3)

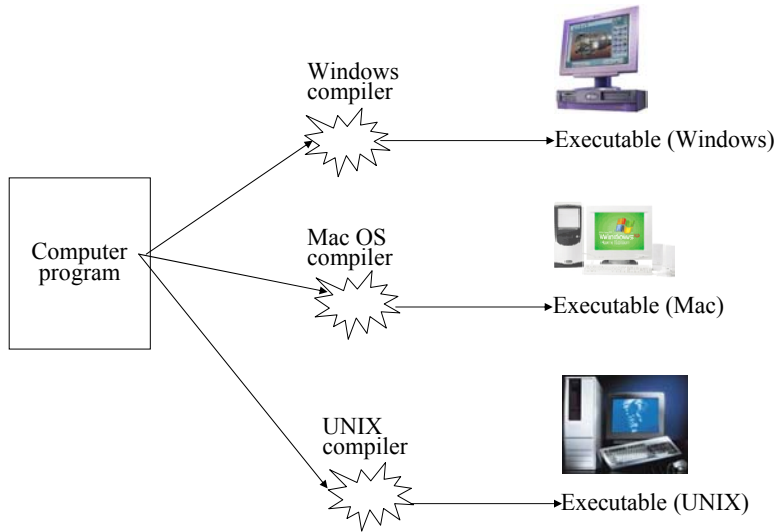
- Java has been used by large and reputable companies to create serious stand-alone applications.
- Example:
 - Eclipse¹: started as a programming environment created by IBM for developing Java programs. The program Eclipse was itself written in Java.



1 For more information: <http://www.eclipse.org/downloads/>

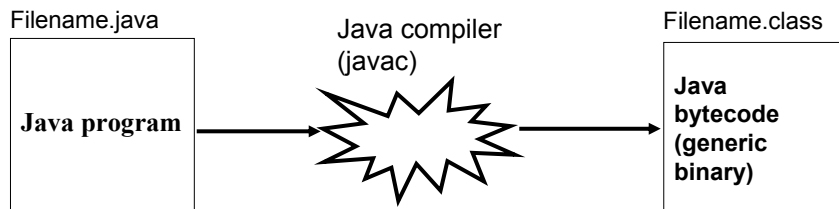
James Tam

Compiled Programs With Different Operating Systems



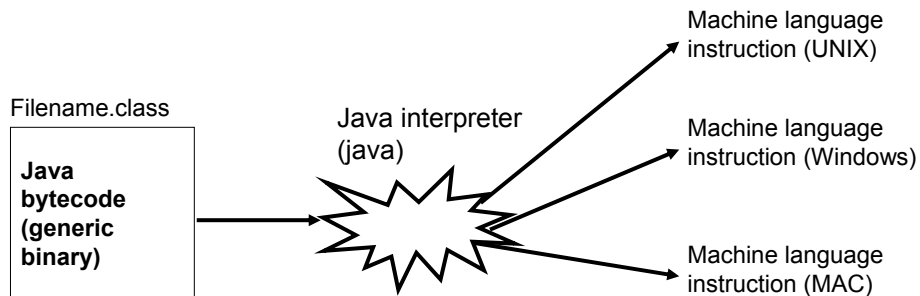
James Tam

A High Level View Of Translating/Executing Java Programs



James Tam

A High Level View Of Translating/Executing Java Programs (2)



James Tam

Which Java?

- Java 6 JDK (Java Development Kit), Standard Edition includes:
 - JDK (Java development kit) – for developing Java software (creating Java programs.
 - JRE (Java Runtime environment) – only good for running pre-created Java programs.
 - Java Plug-in – a special version of the JRE designed to run through web browsers.

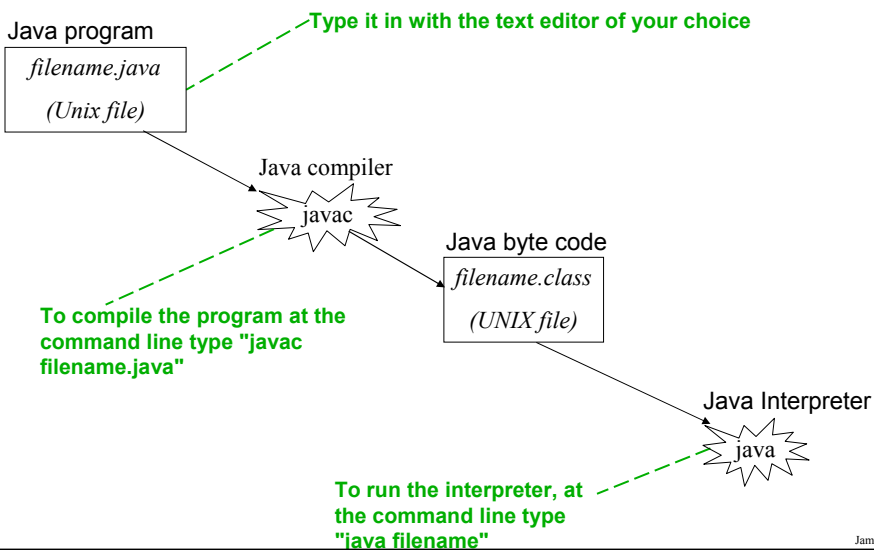
Smallest Compilable And Executable Java Program

This example can be found in UNIX under:
/home/233/examples/introJava/Smallest.java

```
public class Smallest
{
    public static void main (String[] args)
    {
    }
}
```

James Tam

Creating, Compiling And Running Java Programs On The Computer Science Network



James Tam

Compiling The Smallest Java Program

Smallest.java

```
public class Smallest
{
    public static void main (String[] args)
    {
    }
}
```

Type "javac
Smallest.java"

javac

Smallest.class
(Java byte code)

```
100001000000001000
001001000000001001
:
:
```

James Tam

Running The Smallest Java Program

Smallest.class

```
(Java byte code)
100001000000001000
001001000000001001
:
:
```

java

Type "java Smallest"

James Tam

Running The Java Compiler At Home

- After installing Java you will need to indicate to the operating system where the java compiler has been installed ('setting the path').
- For details of how to set your path variable for your particular operating system try the Sun or Java website.
- Example of how to set the path in Windows:
 - <http://java.sun.com/j2se/1.4.2/install-windows.html>

James Tam

Documentation / Comments

Multi-line documentation

```
/* Start of documentation
```

```
*/ End of documentation
```

Documentation for a single line

```
//Everything until the end of the line is a comment
```

James Tam

Java Output

- Format:**

`System.out.println(<string or variable name one> + <string or variable name two>..);`

- Examples** (Assumes a variable called 'num' has been declared.):

`System.out.println("Good-night gracie!");`

`System.out.print(num);`

`System.out.println("num=" + num);`

James Tam

Output : Some Escape Sequences For Formatting

Escape sequence	Description
<code>\t</code>	Horizontal tab
<code>\r</code>	Carriage return
<code>\n</code>	New line
<code>\"</code>	Double quote
<code>\\</code>	Backslash

James Tam

Variables

- Unlike Python variables must be declared before they can be used.
- Variable declaration:
 - Creates a variable in memory.
 - Specify the name of the variable as well as the type of information that it will store.
 - E.g. `int num;`
 - Although requiring variables to be explicitly declared appears to be an unnecessary chore it can actually be useful for minimizing insidious logic errors (more to come later).
- Using variables
 - Only after a variable has been declared can it be used.
 - E.g., `num = 12;`

James Tam

Declaring Variables: Syntax

- **Format:**
<type of information> <name of variable>;
- **Example:**
`char myFirstInitial;`
- Variables can be initialized (set to a starting value) as they're declared:
`int myFirstInitial = 'j';`
`int age = 30;`

James Tam

Some Built-In Types Of Variables In Java

Type	Description
byte	8 bit signed integer
short	16 but signed integer
int	32 bit signed integer
long	64 bit signed integer
float	32 bit signed real number
double	64 bit signed real number
char	16 bit Unicode character
boolean	1 bit true or false value
String	A sequence of characters between double quotes ("")

James Tam

Location Of Variable Declarations

```
public class <name of class>
{
    public static void main (String[] args)
    {
        // Local variable declarations occur here

        << Program statements >>
        :
        :
    }
}
```

James Tam

Java Constants

Reminder: constants are like variables in that they have a name and store a certain type of information but unlike variables they CANNOT change. (Unlike Python this is syntactically enforced).

Format:

```
final <constant type> <CONSTANT NAME> = <value>;
```

Example:

```
final int SIZE = 100;
```

James Tam

Location Of Constant Declarations

```
public class <name of class>
{
    public static void main (String[] args)
    {
        // Local constant declarations occur here
        // Local variable declarations

        < Program statements >>
        :           :
    }
}
```

James Tam

Variable Naming Conventions In Java

- Compiler requirements
 - Can't be a keyword nor can the names of the special constants: true, false or null be used
 - Can be any combination of letters, numbers, underscore or dollar sign (first character must be a letter or underscore)
- Common stylistic conventions
 - The name should describe the purpose of the variable
 - Avoid using the dollar sign
 - With single word variable names, all characters are lower case
 - e.g., double grades;
 - Multiple words are separated by capitalizing the first letter of each word except for the first word
 - e.g., String firstName = "James";

James Tam

Java Keywords

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

James Tam

Common Java Operators / Operator Precedence

Precedence level	Operator	Description	Associativity
1	expression++ expression--	Post-increment Post-decrement	Right to left
2	++expression --expression + - ! ~ (type)	Pre-increment Pre-decrement Unary plus Unary minus Logical negation Bitwise complement Cast	Right to left

James Tam

Common Java Operators / Operator Precedence

Precedence level	Operator	Description	Associativity
3	* / %	Multiplication Division Remainder/modulus	Left to right
4	+ -	Addition or String concatenation Subtraction	Left to right
5	<< >>	Left bitwise shift Right bitwise shift	Left to right

James Tam

Common Java Operators / Operator Precedence

Precedence level	Operator	Description	Associativity
6	< <= > >=	Less than Less than, equal to Greater than Greater than, equal to	Left to right
7	= = !=	Equal to Not equal to	Left to right
8	&	Bitwise AND	Left to right
9	^	Bitwise exclusive OR	Left to right

James Tam

Common Java Operators / Operator Precedence

Precedence level	Operator	Description	Associativity
10		Bitwise OR	Left to right
11	&&	Logical AND	Left to right
12		Logical OR	Left to right

James Tam

Common Java Operators / Operator Precedence

Precedence level	Operator	Description	Associativity
13	=	Assignment	Right to left
	+=	Add, assignment	
	-=	Subtract, assignment	
	*=	Multiply, assignment	
	/=	Division, assignment	
	%=	Remainder, assignment	
	&=	Bitwise AND, assignment	
	^=	Bitwise XOR, assignment	
	=	Bitwise OR, assignment	
	<<=	Left shift, assignment	
	>>=	Right shift, assignment	

James Tam

Post/Pre Operators

This example can be found in UNIX under:
/home/233/examples/introJava/Example1.java

```
public class Example1
{
    public static void main (String [] args)
    {
        int num = 5;
        System.out.println(num);
        num++;
        System.out.println(num);
        ++num;
        System.out.println(num);
        System.out.println(++num);
        System.out.println(num++);
    }
}
```

James Tam

Post/Pre Operators (2)

This example can be found in UNIX under:
/home/233/examples/introJava/Example2.java

```
public class Example2
{
    public static void main (String [] args)
    {
        int num1;
        int num2;
        num1 = 5;
        num2 = ++num1 * num1++;
        System.out.println("num1=" + num1);
        System.out.println("num2=" + num2);
    }
}
```

James Tam

Unary And Casting Operators

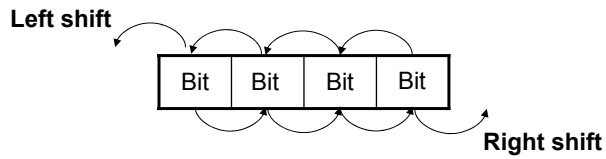
This example can be found in UNIX under:
/home/233/examples/introJava/Example3.java

```
public class Example3
{
    public static void main (String [] args)
    {
        int num = 5;
        float fl;
        System.out.println(num);
        num = num * -num;
        System.out.println(num);
        fl = num;
        System.out.println(num + " " + fl);
        num = (int) fl;
        System.out.println(num + " " + fl);
    }
}
```

James Tam

Bit Level Operations

•Bit shifting



- Format:

Variable name = Value to shift << Number of shifts

- Example:

- unsigned int num = 2;
- num = num << 2;
- num = num >> 2;

James Tam

Bit Level Operations (2)

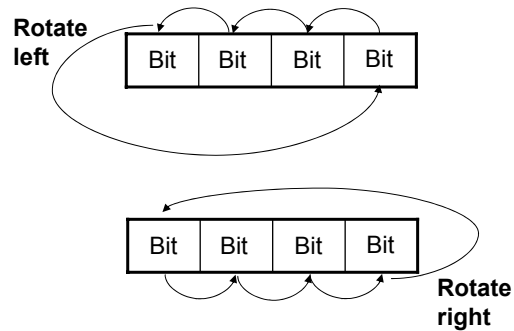
•Bitwise operations

- Bitwise AND '&'
Variable name = Value 1 & Value 2
- Bitwise OR '|'
Variable name = Value 1 | Value 2
- Bitwise NEGATION '~'
Variable name = ~Value

James Tam

Bit Level Operations (3)

- Bit rotations:



- Although it's not implemented in Java with an existing operator you can write the program code for this new operation using the existing ones.

James Tam

There Are Benefits To Using Bitwise Operations

- In some applications speed is an issue important
 - Scenarios where a guaranteed response time is mandatory
 - E.g., the software used to fly an airplane
- Large and complex programs
 - E.g., Complex simulations (Biology, Economics)
 - E.g., games that draw complex graphics

James Tam

Getting Text Input

- You can use the pre-written methods (functions) in the Scanner class.
- **General structure:**

```
import java.util.Scanner;

main (String [] args)
{
    Scanner <name of scanner> = new Scanner (System.in);
    <variable> = <name of scanner> .<method> ();
}
```

James Tam

Getting Text Input (2)

- **Example:**

This example can be found in UNIX under:
/home/233/examples/introJava/MyInput.java

```
import java.util.Scanner;
public class MyInput
{
    public static void main (String [] args)
    {
        String str1;
        int num1;
        char ch;
        Scanner in = new Scanner (System.in);
        System.out.print ("Type in an integer: ");
        num1 = in.nextInt ();
        System.out.print ("Type in a line: ");
        in.nextLine ();
        str1 = in.nextLine ();
        System.out.println ("num1:" + num1 + "\t str1:" + str1);
    }
}
```

James Tam

Useful Methods Of Class Scanner¹

- nextInt ()
- nextLong ()
- nextFloat ()
- nextDouble ()

¹ Online documentation: <http://java.sun.com/javase/6/docs/api/>

Decision Making In Java

- Java decision making constructs
 - if
 - if, else
 - if, else-if
 - switch

Decision Making: Logical Operators

Logical Operation	Python	Java
AND	and	&&
OR	or	
NOT	Not, !	!

James Tam

Decision Making: If

Format:

```
if (Boolean Expression)  
    Body
```

Example:

```
if (x != y)  
    System.out.println("X and Y are not equal");  
  
if ((x > 0) && (y > 0))  
{  
    System.out.println("X and Y are positive");  
}
```

- Indenting the body of the branch is an important stylistic requirement of Java but unlike Python it is not enforced by the syntax of the language.
- What distinguishes the body is either:
 1. A semi colon (single statement branch)
 2. Braces (a body that consists of multiple statements)

James Tam

Decision Making: If, Else

Format:

```
if (Boolean expression)
    Body of if
else
    Body of else
```

Example:

```
if (x < 0)
    System.out.println("X is negative");
else
    System.out.println("X is non-negative");
```

James Tam

If, Else-If

Format:

```
if (Boolean expression)
    Body of if
else if (Boolean expression)
    Body of first else-if
    :
    :
else if (Boolean expression)
    Body of last else-if
else
    Body of else
```

James Tam

If, Else-If (2)

Example:

```
if (gpa == 4)
{
    System.out.println("A");
}
else if (gpa == 3)
{
    System.out.println("B");
}
else if (gpa == 2)
{
    System.out.println("C");
}
```

James Tam

If, Else-If (2)

```
else if (gpa == 1)
{
    System.out.println("D");
}
else
{
    System.out.println("Invalid gpa");
}
```

James Tam

Alternative To Multiple Else-If's: Switch (2)

Format (character-based switch):

switch (*character variable name*)

```
{
  case '<character value>':
    Body
    break;

  case '<character value>':
    Body
    break;
  :
  default:
    Body
}
```

! The type of variable in the brackets can be a byte, char, short, int or long

James Tam

Alternative To Multiple Else-If's: Switch (2)

Format (integer based switch):

switch (*integer variable name*)

```
{
  case <integer value>:
    Body
    break;

  case <integer value>:
    Body
    break;
  :
  default:
    Body
}
```

! The type of variable in the brackets can be a byte, char, short, int or long

James Tam

Loops

Python loops

- Pre-test loops: for, while

Java Pre-test loops

- For
- While

Java Post-test loop

- Do-while

James Tam

While Loops

Format:

```
while (Expression)  
    Body
```

Example:

```
int i = 1;  
while (i <= 1000000)  
{  
    System.out.println("How much do I love thee?");  
    System.out.println("Let me count the ways: ", + i);  
    i = i + 1;  
}
```

James Tam

For Loops

Format:

```
for (initialization; Boolean expression; update control)  
    Body
```

Example:

```
for (i = 1; i <= 1000000; i++)  
{  
    System.out.println("How much do I love thee?");  
    System.out.println("Let me count the ways: " + i);  
}
```

James Tam

Do-While Loops

Format:

```
do  
    Body  
while (Boolean expression);
```

Example:

```
char ch = 'A';  
do  
{  
    System.out.println(ch);  
    ch++;  
}  
while (ch != 'K');
```

James Tam

Many Pre-Created Classes Have Been Created

- Rule of thumb: Before writing new program code to implement the features of your program you should check to see if a class has already been written with the features that you need.
- The Java API is Sun Microsystems's collection of pre-built Java classes:
 - <http://java.sun.com/javase/6/docs/api/>

James Tam

After This Section You Should Now Know

- How Java was developed and the impact of it's roots on the language
- The basic structure required in creating a simple Java program as well as how to compile and run programs
- How to document a Java program
- How to perform text based input and output in Java
- The declaration of constants and variables
- What are the common Java operators and how they work
- The structure and syntax of decision making and looping constructs

James Tam