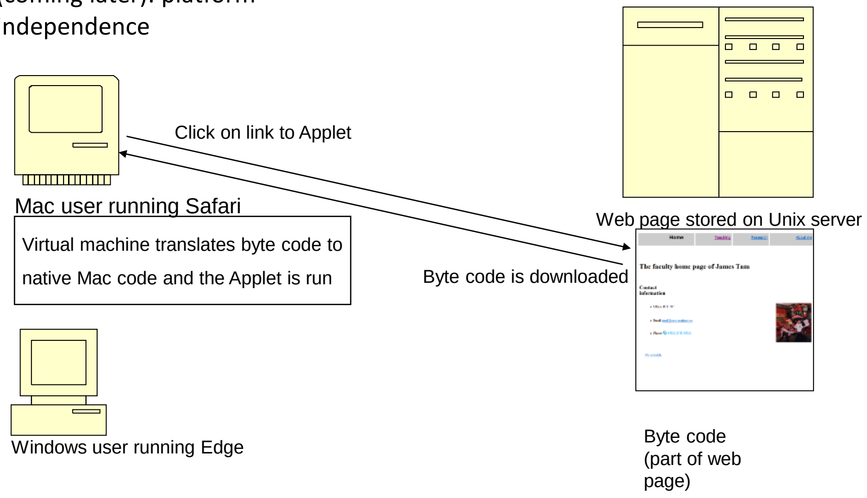
Introduction To Java Programming

You will learn the basics of creating a Java program: writing/translating a program, writing documentation, declaring variables, constants, getting input, displaying output.

James Tam

Java: Write Once, Run Anywhere

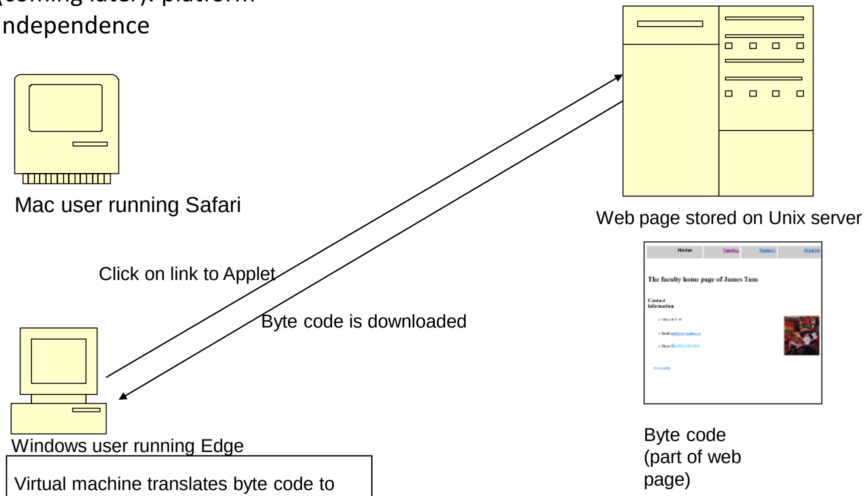
- Consequence of Java's history (coming later): platform-independence


Click on link to Applet
Mac user running Safari
Virtual machine translates byte code to native Mac code and the Applet is run
Byte code is downloaded
Web page stored on Unix server
Windows user running Edge
Byte code (part of web page)

James Tam

Java: Write Once, Run Anywhere

- Consequence of Java's history (coming later): platform-independence



Virtual machine translates byte code to native Windows code and the Applet is run

James Tam

Java: Write Once, Run Anywhere (2)

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

Dungeon Master (Java version)
Accessed Jan. 2021
Dungeon Master Java Download (2001 Role playing Game) (old-games.com)

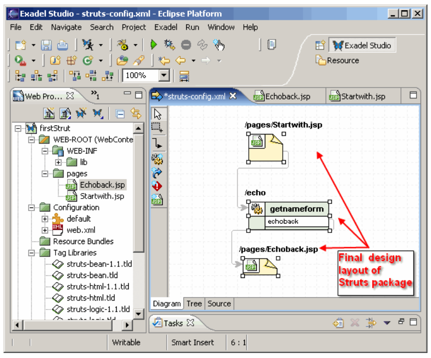
Kung Fu Panda:
Accessed 2013
screen grab from www.kunfupanda.com

Some examples of mobile Java games: http://www.mobilegamesarena.net

James Tam

Java: Write Once, Run Anywhere (3)

- Java has been used by large and reputable companies to create serious stand-alone applications.
- Example:
  - Eclipse[1]: started as a programming environment created by IBM for developing Java programs. The program Eclipse was itself written in Java.



1 For more information (last accessed 2015): http://www.eclipse.org/downloads/

James Tam

JT's Note: IDEs

- There are many graphical development environments available for Java (e.g., Eclipse).
- Learning one or more these environments prior to embarking on employment would be a valuable experience.
- However it is not recommended that you use them for this course.
  - You may have drastic problems configuring the environment (e.g., if you have to use example starting code).
  - It's easier programming without an IDE and then learning one later than the opposite (not all development teams can/will use them).
  - With the size of the programs you will see in this class it would be a good learning experience to 'work without a net'.
    - Because you have to do it all yourself you will likely learn things better.

James Tam

IDEs: Bottom Line

- Assignments must be submitted in the form of `.java` text files that will compile and run on the computer science network.

Remote learning version: program needs to work in the latest version of Java when run compiled and run through a command line

- If you have problems with the IDE or getting your programs to work on our network then you will likely be on your own.

James Tam

Official Online Java Documentation

- "Getting started" tutorials (last accessed 2021):
  - http://docs.oracle.com/javase/tutorial/

James Tam

Which Java?

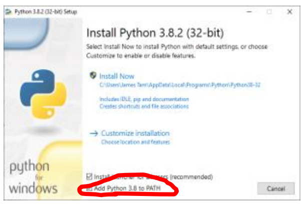
- Download link:
  - https://www.oracle.com/java/technologies/javase-downloads.html
- Java JDK (Java Development Kit), Standard Edition includes:
  - JDK (Java development kit) – for *developing* Java software (creating Java programs).
  - JRE (Java Runtime environment) –for *running* pre-created Java programs.
    - Java Plug-in – a special version of the JRE designed to run through web browsers.
- For consistency/fairness: Your graded the version of Java installed on the CP
  - Only run your program using a remote connecti Linux computer) or test your code periodically o compatible.
  - It's your responsibility to ensure compatibility.
  - If the program doesn't work on the Lunix computers in the lab then it will only receive partial marks (at most).

Remote learning version: program needs to work in the latest version of Java when run compiled and run through a command line

James Tam

Getting Java Setup

- Unlike installing python the java install leaves out one small but critical step:
  - (Python):



Python 3.8.2 (32-bit) Setup

Install Python 3.8.2 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\James Tam\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.8 to PATH

Cancel

  - (The 'path' specifies to your operating system the location or 'path' to where the translation program 'python' or 'javac/java' resides on your computer).
  - JT's editorial opinion: since Java is also used as a language for beginners this was a real dumb omission. (Don't blame JT!)
    - The benefits of leaving out the option provided with Python don't outweigh the costs.

James Tam

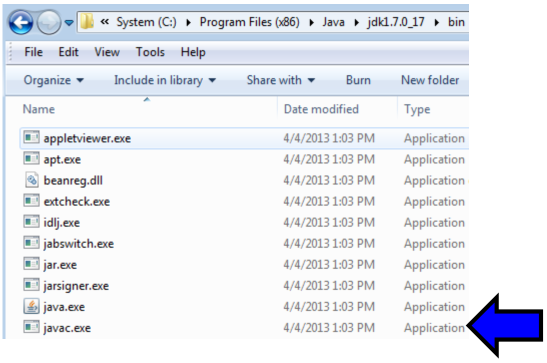
Setting The Path For Java

- Windows help document (Step II specifies how to set the path): https://www.oracle.com/webfolder/technetwork/tutorials/OracleCode/Windows-HOL-setup.pdf
- Help link for other operating systems (MAC-OS, UNIX):
  - https://www.oracle.com/java/technologies/installation-solaris2-009.html
- Web search terms if you don't like these tutorials: '`set`' or '`setting`', '`java`', '`classpath`'

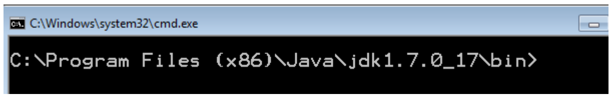
James Tam

Alternative: Simple But A Hack

- Create your Java programs in the same location as the **Java compiler** (you should remember where you installed it).


« System (C:) ▸ Program Files (x86) ▸ Java ▸ jdk1.7.0_17 ▸ bin
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

Name	Date modified	Type
appletviewer.exe	4/4/2013 1:03 PM	Application
apt.exe	4/4/2013 1:03 PM	Application
beanreg.dll	4/4/2013 1:03 PM	Application
extcheck.exe	4/4/2013 1:03 PM	Application
idlj.exe	4/4/2013 1:03 PM	Application
jabswitch.exe	4/4/2013 1:03 PM	Application
jar.exe	4/4/2013 1:03 PM	Application
jarsigner.exe	4/4/2013 1:03 PM	Application
java.exe	4/4/2013 1:03 PM	Application
javac.exe	4/4/2013 1:03 PM	Application



```
C:\Program Files (x86)\Java\jdk1.7.0_17\bin>
```

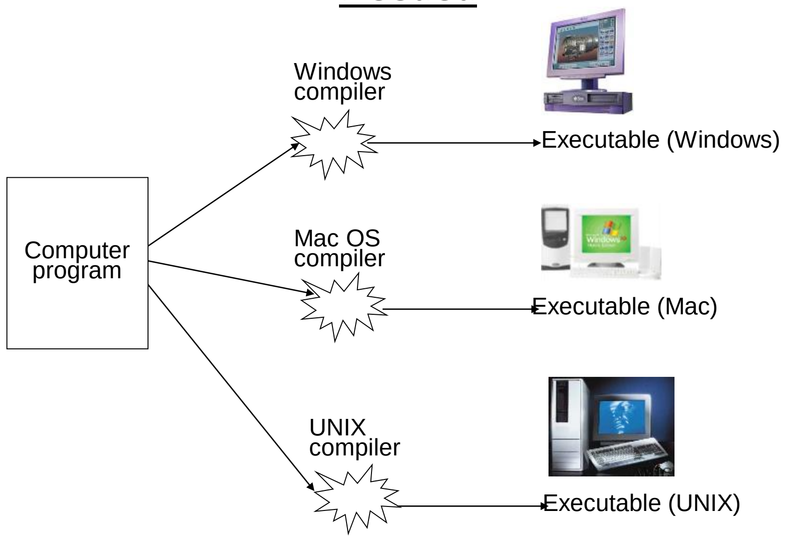
James Tam

Compilation

- Translating from a high level programming language such as Java or C++ to low level machine language (binary).
- Python:
  - One stage translation process: from Python to machine occurs each time the program runs.
  - The translated instructions remain in memory.
- Java
  - Two stage process:
    1. A one time translation occurs from Java to a generic binary that is common to many computers and many electronic devices (this creates a 'byte code' file)
    2. Each time the program is run the generic binary is translated to machine language which is specific to the computer or device.

James Tam

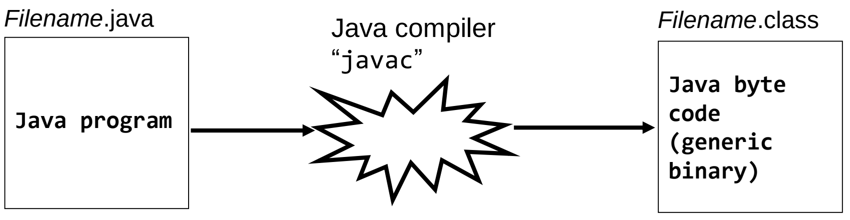
Compiled Programs With Different Operating Systems: Multiple Compilers Needed



James Tam

A High Level View Of Translating/Executing Java Programs

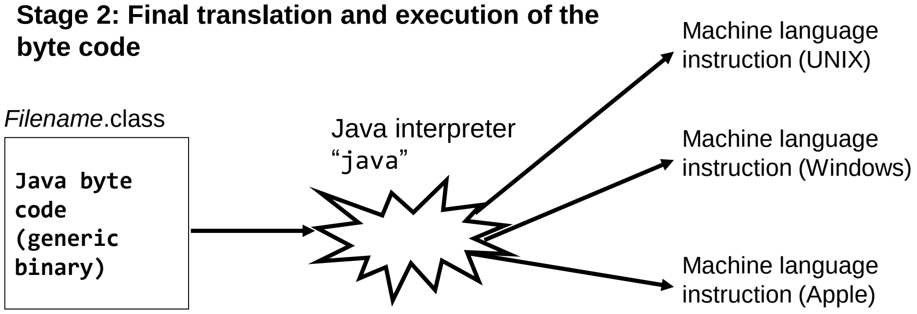
Stage 1: Compilation



James Tam

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

Stage 2: Final translation and execution of the byte code


Filename.class
Java byte code (generic binary)
Java interpreter
“java”
Machine language instruction (UNIX)
Machine language instruction (Windows)
Machine language instruction (Apple)

James Tam

Java Syntax Requirements To Compile A Program

- Type the following into a text file called “Smallest.java”:

```
public class Smallest
{
    public static void main(String[] args)
    {
        System.out.println("Small program running");
    }
}
```

James Tam

Compiling And Running A Java Program (Windows)

- These steps assume that you already have Java installed on your computer and the path has already been properly set!

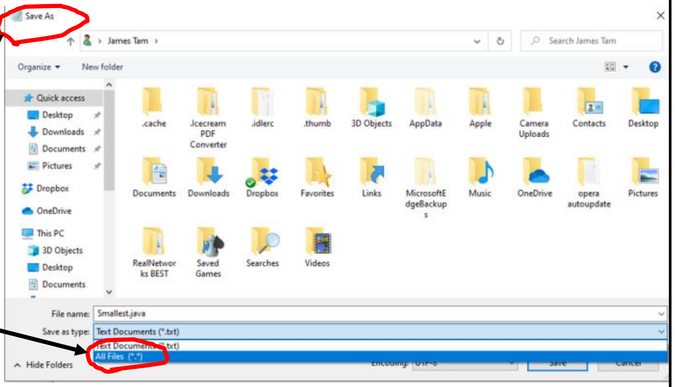
1. Type in your program using a text editor (e.g. Notepad, WordPad, Notepad++).
   a) Save the program as a ‘text’ file (for now try it with the program called `Smallest.java`. Make sure the program ends in the suffix **`.java`** and **NOT** anything else e.g. **`.txt`**.
   b) For now save the program under `C:\users\<`*`Your Windows user name`*`>`
2. Open a command line (‘`cmd`’ in Windows or ‘`terminal`’ with a computer running OS X).
3. If the command line shows the location as `C:\users\<`*`Your Windows user name`*`>` then type the following. If the command line opens elsewhere then move your Java program to this location.
   a) Compile the program by typing the following at the command line: `javac Smallest.java`
   b) If your program has no error messages specifying syntax problems then the command line it will then allow you to enter new input. Run the interpreter (and run the program) by typing: `java Smallest`

James Tam

Step 1: Example Of Saving (Notepad)

"Save As" allows the file suffix to be changed from the '.txt' default

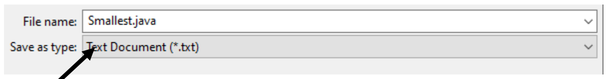
Select "All Files" and then you can add the suffix '.java' to the file name.



File name: Smallest.java
Save as type: Text Documents (*.txt)
Text Documents (*.txt)
All Files (*.*)

James Tam

Step 1: Example Of Saving (WordPad)



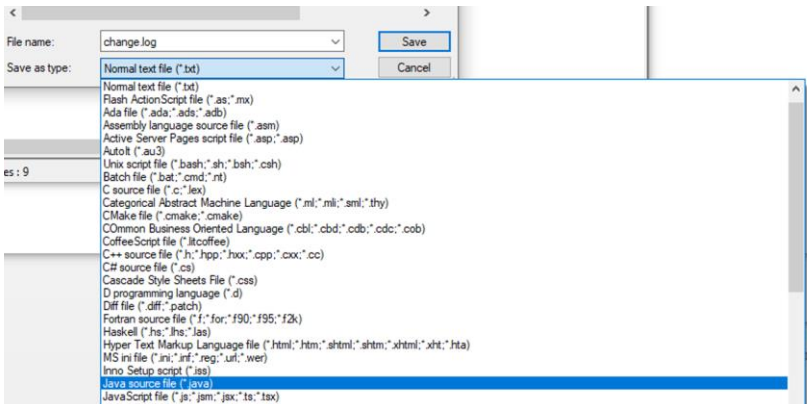
File name: Smallest.java
Save as type: Text Document (*.txt)

Select "Save As" from the menu and then the file type as "Text document" but change the suffix from 'txt' to 'java'

James Tam

Step 1: Example Of Saving (Notepad++)

- “Save as” type: Java source file.



File name: change.log

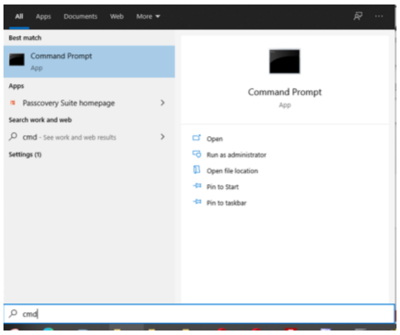
Save as type: Normal text file (*.txt)

Save
Cancel
es : 9
Normal text file (*.txt)
Flash ActionScript file (*.as;*.mx)
Ada file (*.ada;*.ads;*.adb)
Assembly language source file (*.asm)
Active Server Pages script file (*.asp;*.asp)
AutoIt (*.au3)
Unix script file (*.bash;*.sh;*.bsh;*.csh)
Batch file (*.bat;*.cmd;*.nt)
C source file (*.c;*.lex)
Categorical Abstract Machine Language (*.ml;*.mli;*.sml;*.thy)
CMake file (*.cmake;*.cmake)
COmmon Business Oriented Language (*.cbl;*.cbd;*.cdb;*.cdc;*.cob)
CoffeeScript file (*.litcoffee)
C++ source file (*.h;*.hpp;*.hxx;*.cpp;*.cxx;*.cc)
C# source file (*.cs)
Cascade Style Sheets File (*.css)
D programming language (*.d)
Diff file (*.diff;*.patch)
Fortran source file (*.f;*.for;*.f90;*.f95;*.f2k)
Haskell (*.hs;*.lhs;*.las)
Hyper Text Markup Language file (*.html;*.htm;*.shtml;*.shtm;*.xhtml;*.xht;*.hta)
MS ini file (*.ini;*.inf;*.reg;*.url;*.wer)
Inno Setup script (*.iss)
Java source file (*.java)
JavaScript file (*.js;*.jsm;*.jsx;*.ts;*.tsx)

James Tam

Step 2: Starting A Command Line (Windows Cmd)

- Click on the `Start` button and type ‘`cmd`’

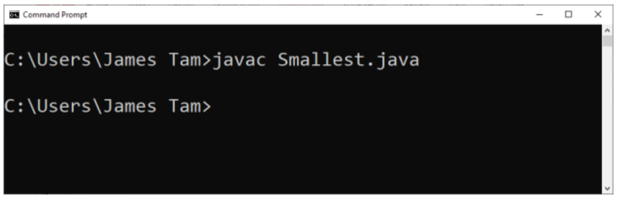


- Click on the ‘Open’ option for the Command Prompt (you may need to click on “Run as Administrator” if you are not currently running an administrator account).

James Tam

Step 3A: Compile The Program

- What's needed: The location opened by the prompt matches the recommended location for this example (for me its `C:\Users\James Tam`).
- Compile the program (translate to machine/binary) by typing the following at the command line and then hit enter: `javac Smallest.java`
- In this case there were no syntax errors and after compilation the command prompt returns.



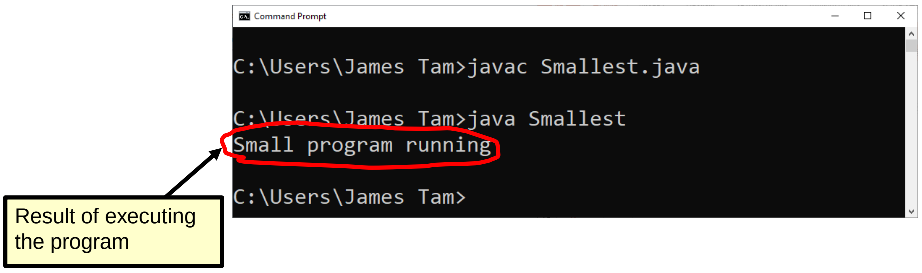
```
C:\Users\James Tam>javac Smallest.java

C:\Users\James Tam>
```

James Tam

Step 3B: Run The Program

- Run the Java interpreter (which runs your program) by typing the following at the command line and hitting enter: `java Smallest`



```
C:\Users\James Tam>javac Smallest.java

C:\Users\James Tam>java Smallest
Small program running

C:\Users\James Tam>
```

Result of executing the program

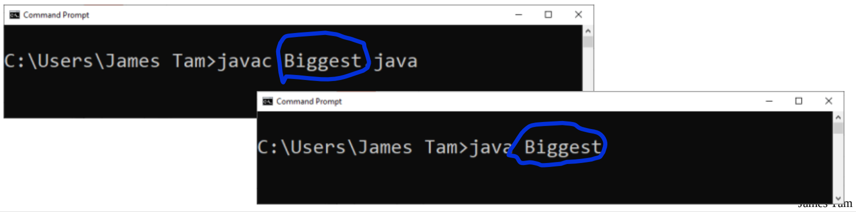
```
public class Smallest
{
    public static void main(String[] args)
    {
        System.out.println("Small program running");
    }
}
```

James Tam

General Rule: Creating/Running Programs

- Very important Java requirement: the **name after the word class** must match the name of the file AND it is what you type in when compile (ending in `.java`) and run your program (no suffix).

```
public class Biggest
{
    public static void main(String[] args)
    {
        System.out.println("Big program");
    }
}
```



```
C:\Users\James Tam>javac Biggest.java
```

```
C:\Users\James Tam>java Biggest
```

James Tam

Documentation / Comments

Multi-line documentation

```
/*   Start of documentation
*/   End of documentation
```

- Don’t nest this form of documentation (results in a syntax error)

Documentation for a single line

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

James Tam

Review: What Should You Document

- Program author
- What does the program as a while do e.g., tax program.
- What are the specific features of the program e.g., it calculates personal or small business tax.
- What are its limitations e.g., it only follows Canadian tax laws and cannot be used in the US. In Canada it doesn't calculate taxes for organizations with yearly gross earnings over $1 billion.
- What is the version of the program
  - If you don't use numbers for the different versions of your program then consider using dates
  - Tie versions with program features: for each version list the features completed.

James Tam

Important Note

- Each Java instruction must be followed by a semi-colon!

General format

```
Instruction1;
Instruction2;
Instruction3;
   :       :
```

Examples

```
int num = 0;
System.out.println(num);
  :    :
```

James Tam

Java Output: Common Methods (~Function)

- Print only the output specified (*no other formatting*: spaces, tabs, newlines)

  (Java)

  `System.out.print();`

  (Python)

  `print(…, end="")`

- Print the output specified *followed by a newline*.

  (Java)

  `System.out.println();`

  (Python)

  `print()`

James Tam

Java Output: Specifics

- **Format:**

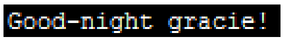
  System.out.print(<*string or variable name one*> + <*string or variable name two*>..);

  OR

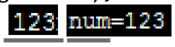
  System.out.println(<*string or variable name one*> + <*string or variable name two*>..);

- **Complete online example:** `OutputExample.java`

```
public class OutputExample
{
    public static void main(String [] args)
    {
        int num = 123;  // Details coming
        System.out.println("Good-night gracie!");
        System.out.print(num);
        System.out.println("num="+num);
    }
}
```



```
Good-night gracie!
```



```
123num=123
```

James Tam

Output : Some Escape Sequences For Formatting

- The escape sequence is placed between the quotes in `print()` or `println()` e.g., `System.out.print("hi\tthere");`

Escape sequence	Description
`\t`	Horizontal tab
`\n`	New line
`\"`	Double quote
`\\`	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 (example to follow shortly).
- Using variables
  - Only after a variable has been declared can it be used (e.g., assignment)
  - E.g., `num = 12;`

James Tam

Using Variables: A Contrast

Python

- Variables do not need to be declared before being used.
- Easy to start programming.
- Easy to make logic errors!

```
incomeTam = 25000
if (winLottery):
    incomeTan = 1000000
```

BAD, Logic error: can be tricky to catch in a real (large and complex) program

Java

- Syntax rule: variables must always be declared prior to use.
- A little more work to get started.
- Some logic errors may be prevented.

```
int incomeTam = 25000;
if (winLottery)
    incomeSmith = 1000000;
```

BETTER, Syntax error: compiler points out the source of the problem

James Tam

Declaring Variables: Syntax

- **Format**:
  `<type of information> <name of variable>;`
- **Example**:
  `char firstInitial;`
- Variables should be initialized (set to a starting value) as they're declared:

```
char firstInitial = 'j';
String firstName = "James";
int age = 30;
```

Note: In Java Strings MUST be enclosed in double quotes. Single quotes are used for characters (which are not the same as a Java String).

James Tam

Some Built-In Types Of Variables In Java

Type	Description
byte	8 bit integer
short	16 bit integer
int	32 bit integer
long	*64 bit integer*
float	32 bit real number (rare)
double	*64 bit real number (default for many operations)*
char	16 bit Unicode character (stores ASCII values and values beyond). **Use single quotes**.
boolean	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 Strings

- Similar to Python strings: a sequence of characters indexed from zero to (length – 1)
  - Don't try to directly access elements via the index e.g., `string1[0];`
  - Elements can be accessed via a method `charAt(int)`
- Unlike Python strings Java Strings **only use double quotes**
- (In Java single quotes encloses a single character)
- **Format** (creating string variable):
  `String <string name> = "<value>";`

- **Example** (creating string variable):
  `String username = "tamj";`

James Tam

Common String Methods

- **Examples useful methods**:

Method	Description
`string.charAt(int)`	Retrieves character at the specified index
`string.compareTo(String s)`	Compares string with parameter: • Zero returned if string and parameter equal • Less than zero if the string comes before the parameter • Greater than zero if the string comes after parameter
`string.compareToIgnoreCase (String s)`	As `compareTo()` but case insensitive
`string.length()`	Returns the length of the string
`string.toLowerCase()`	Converts alphabetic characters to lower case
`string.toUpperCase()`	Converts alphabetic characters to capitals

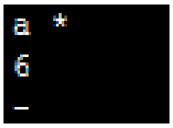
For more info look under "class String"
http://docs.oracle.com/javase/8/docs/api/

James Tam

A String Example

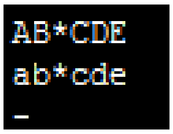
- **Name of the complete online example:** `StringEg.java`

```
String myString = "ab*cde";
System.out.println(myString.charAt(0) +
  " " + myString.charAt(2));
System.out.println(myString.length());
System.out.println("-");
```



```
a *
6
-
```

```
myString = myString.toUpperCase();
System.out.println(myString);
myString = myString.toLowerCase();
System.out.println(myString);
System.out.println("-");
```

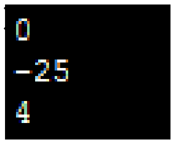


```
AB*CDE
ab*cde
-
```

James Tam

A String Example (2)

```
// recall myString = "ab*cde"
System.out.println
  (myString.compareToIgnoreCase("ab*cde"
System.out.println
  (myString.compareToIgnoreCase("zzz"));
System.out.println
  (myString.compareToIgnoreCase("ab"));
```



```
0
-25
4
```

ab*cde(zzz)

ab*cde(ab)

James Tam

Style Hint: Initializing Variables

- Always initialize your variables prior to using them!
  - Do this whether it is or is not required by the syntax of the language.
- Example: A how not to approach (with some languages you'll get a logic and not a syntax error as you do with Java).
- **Name of the complete online example:** `OutputExample.java`

```
public class OutputExample1
{
   public static void main(String [] args)
   {
       int num;
       System.out.print(num);
    }
}
```

OutputExample1.java:7: error: variable num might not have been initialized System.out.print(num);

James Tam

Formatting Output: Elective Topic

- It's somewhat similar to Python.
- The field width and places of precision (float point) can be specified.
- **Format ('System.out.' requirement excluded for brevity)**:

```
printf("%<field width>d", price);      // Integer
printf("%<field width>s", price);      // String
printf("%<field width>.<precision>f", price);   // Floating point
```

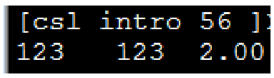
- If field width greater than the size of the data:
  - A **positive field width** will result in **leading spaces** (right justify).
  - A **negative field width** will result in **trailing spaces** (left justify).

James Tam

Formatting Output (2): Elective Topic

- **Name of the complete online example**: `FormattingExample.java`

```
public class FormattingExample
{
    public static void main(String [] args)
    {
        String str = "123";
        int num = 123;
        double price = 1.999;
        System.out.printf("%-4s", str);
        System.out.printf("%5d", num);
        System.out.printf("%6.2f", price);
    }
}
```



```
[csl intro 56 ]
123    123   2.00
```

James Tam

Java Constants ("`Final`")

- 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…that's a good thing!).
- The syntactically enforced unchanging nature of constants is specified with the '`final`' key word.
- Stylistic reminder: constants must be capitalized.

Format:

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

Example:

```
    final int SIZE = 100;
    SIZE = 1000;  // Syntax error
```

James Tam

Location Of Constant Declarations

```
public class <name of class>
{
    public static void main(String[] args)
    {
       // Local constant declarations occur here (for now)
       // 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";`
  - To avoid confusion don't use method names
    - E.g. `String println;`

James Tam

Some Java Keywords (Avoid Using As Identifiers)

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 Operators

Operation	Operator	Example usage
Assignment	=	num = 123;
Addition	+	num = 2 + 2;
Subtraction	-	num = 5 - 2;
Multiplication	*	num = num * 2;
Division	/	num = 9 / 3;
Remainder	%	num = 9 % 2
Negation	-	-num;

James Tam

Post/Pre Operators

- **Post**/**Pre** Increment
- A common shorthand notation used in several languages (e.g., Java, C, C++) which will increase a variable by one.
- **Post-increment**

```
num++;
```

- **Pre-increment**

```
++num;
```

- Pre vs. post operators will produce identical results if the increment is the only operation (two previous examples):
- The specific difference between ‘post’ vs. ‘pre’ will be coming up shortly

James Tam

Post/Pre Decrement

- Operates in a similar fashion to post/pre decrement except that a variable is decreased by one.
- Post decrement

```
num--;
```

- Pre decrement

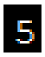
```
--num;
```

James Tam

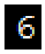
Post/Pre Operators

Name of the complete online example: `Order.java`

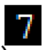
```
public class Order
{
    public static void main(String [] args)
    {
        int num = 5;
        System.out.println(num);
        num++;
        System.out.println(num);
        ++num;
        System.out.println(num);
        System.out.println(++num);
        System.out.println(num++);
    }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



```
5
```



```
6
```



```
7
```

```
8
```

```
8
```

James Tam

Casting: Converting Between Types

- Casting: the ability to convert between types.
  - Of course the conversion between types must be logical otherwise an error will result e.g., multiplication on a String is a nonsensical operation
- In Java unlike Python the conversion isn't just limited to a limited number of functions.
  - Consequently Python doesn't have true 'casting' ability.
- **Format**:
  `<`*`Variable name`*`> = (`*`type to convert to`*`) <`*`Variable name`*`>;`

James Tam

Casting: Structure And Examples

Name of the complete online example: `Casting.java`

Converting/casting types:
- Simple but important concept
- Going from 'more' to 'less' and 'less' to 'more': we'll return back to this in the 'hierarchies' section (inheritance)

```
public class Casting {
    public static void main(String [] args) {
        int num1;
        double num2;
        num2 = 1.9;

        // Cast needed to explicitly convert (going from more to less)
        num1 = (int) num2;
        System.out.println(num1 + " " + num2);
        // Cast not needed: going from less to more
        num2 = num1;
        System.out.println(num1 + " " + num2);
    }
}
```

```
1 1.9
```

```
1 1.0
```

James Tam

Accessing Pre-Created Java Libraries

- It's accomplished by placing an '`import`' of the appropriate library at the top of your program.
- **Syntax:**

  `import <`*`Full library name`*`>;`

- **Example:**

  `import java.util.Scanner;`

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>();
}
```

Creating scanner entity (object)

Getting user input with a method

Getting Text Input (2)

Name of the complete online example: `MyInput1.java`

```
import java.util.Scanner;

public class MyInput1
{
    public static void main(String [] args)
    {
        String name;
        int age;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter your age: ");
        age = in.nextInt();
        in.nextLine();
        System.out.print("Enter your name: ");
        name = in.nextLine();
        System.out.println("Age: " +age +"\t Name:" + name);
    }
}
```

Useful Methods Of Class Scanner[1]

- `nextInt()`
- `nextLong()`
- `nextFloat()`
- `nextDouble()`
- `nextLine()`

1 Online documentation: http://docs.oracle.com/javase/8/docs/api/

James Tam

Reading A Single Character

- Text menu driven programs may require this capability.
- Example:

```
GAME OPTIONS
(a)dd a new player
(l)oad a saved game
(s)ave game
(q)uit game
```

- There's different ways of handling this problem but one approach is to extract the first character from the string.
- Partial example:

```
String s = "boo";
System.out.println(s.charAt(0));
```

James Tam

Getting Input: A Common Issue

- Many methods or functions that get non-String input (e.g. numeric) will leave the "End of Line" <EOL> character in the input buffer. (Not Java specific).
  - Example when you type in a number you must signal the end of the input by hitting enter.
  - The enter character signifies that the entry of input has occurred ("end of the line" has been reached) and is a valid String.
- This can be a problem if there's a need to get String input immediately afterward.
  - Example: get the user to enter a number and then a String.
- Solution:
  - After getting non-String input call a function or method immediately afterward that gets String input to remove the EOL from the buffer.
  - Then call the function/method a second time to prompt for the String.

James Tam

Getting Input: Example Solution

- **Name of the complete online example**: `MyInput2.java`

```
public class MyInput2
{
    public static void main(String [] args)
    {
        String name;
        int age;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter your age: ");
        age = in.nextInt();
        in.nextLine(); //Solution
        System.out.print("Enter your name: ");
        name = in.nextLine();
        System.out.println("Hi " + name + " you're "
         + age);
    }
}
```

James Tam

Copyright Notification

slide 61

James Tam