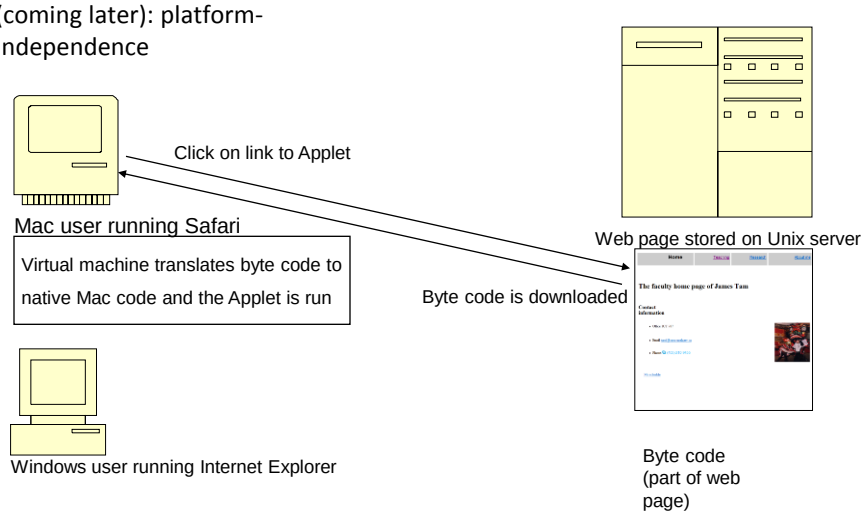
Introduction To Java Programming

You will learn about the process of creating Java programs and constructs for input, output, branching, looping and arrays.

James Tam

Java: Write Once, Run Anywhere

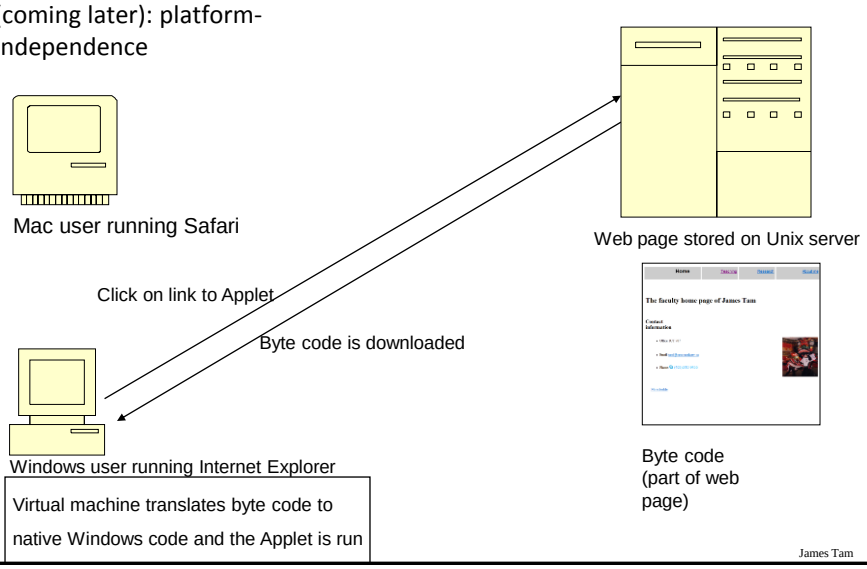
- Consequence of Java's history (coming later): platform-independence



James Tam

Java: Write Once, Run Anywhere

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



James Tam

Java: Write Once, Run Anywhere (2)

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

Dungeon Master (Java version)
Accessed 2013
http://homepage.mac.com/aberfield/dmj/

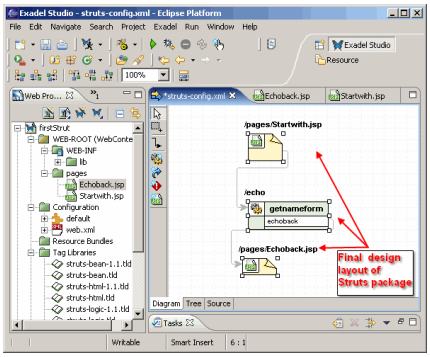
Kung Fu Panda:
Accessed 2013
screen grab from www.kunfupanda.com

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: http://www.eclipse.org/downloads/

JT's Note: IDE's

- 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.

IDE's: Bottom Line

- Assignments must be submitted in the form of `.java` text files that will compile and run on the computer science network.
- 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:
  - http://docs.oracle.com/javase/tutorial/

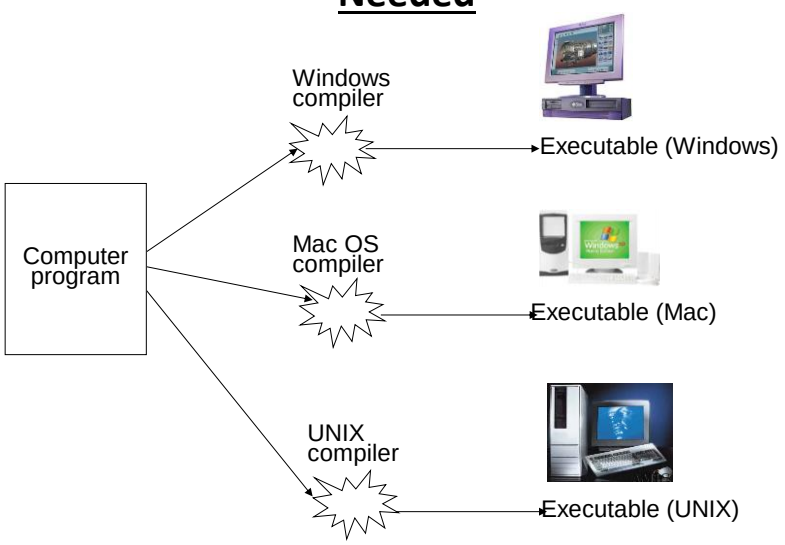
James Tam

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.
  - The translated instructions remain in memory.
- Java
  - Two stage process: 1) one time translation occurs Java to a generic binary that is common to many computers and many electronic devices (this creates a file) 2) when the program is run the generic binary is translated to machine language that is specific to the device.

James Tam

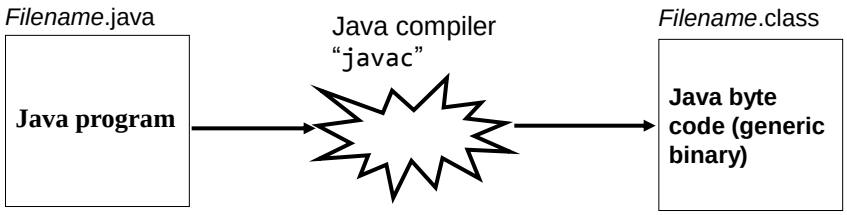
Compiled Programs With Different Operating Systems: Multiple Compilers Needed



James Tam

A High Level View Of Translating/Executing Java Programs

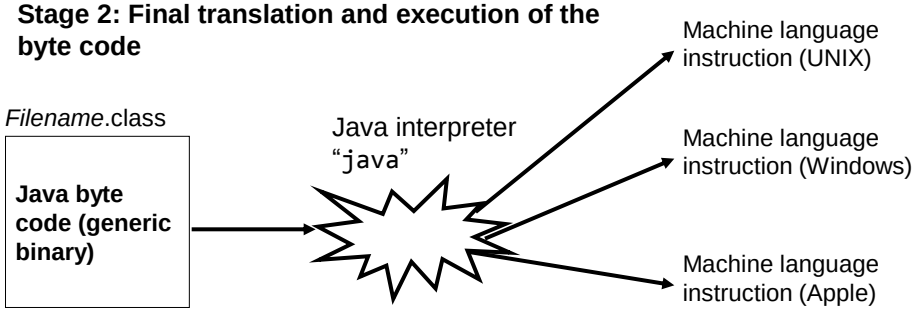
Stage 1: Compilation



James Tam

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

Stage 2: Final translation and execution of the byte code



James Tam

http://java.sun.com/javase/downloads/index.jsp

Which Java?

- Java JDK (Java Development Kit), Standard Edition includes:
  - J*D*K (Java development kit) – for *developing* Java software (creating Java programs).
  - J*R*E (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 work will be based on the version of Java installed on the CPSC network
  - Only run your program using a remote connection program (e.g., SSH to a CPSC Linux computer) or test your code periodically on the network to make sure it's 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).

James Tam

Location Of Online Examples For This Section

- Course website:
  `-www.cpsc.ucalgary.ca/~tamj/233/examples/intro`
- UNIX directory:
  `-/home/233/examples/intro`

James Tam

Smallest Compilable And Executable Java Program

The name of the online example is: `Smallest.java` (*Important note: the file name must match the word after the keyword 'class' below*).

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

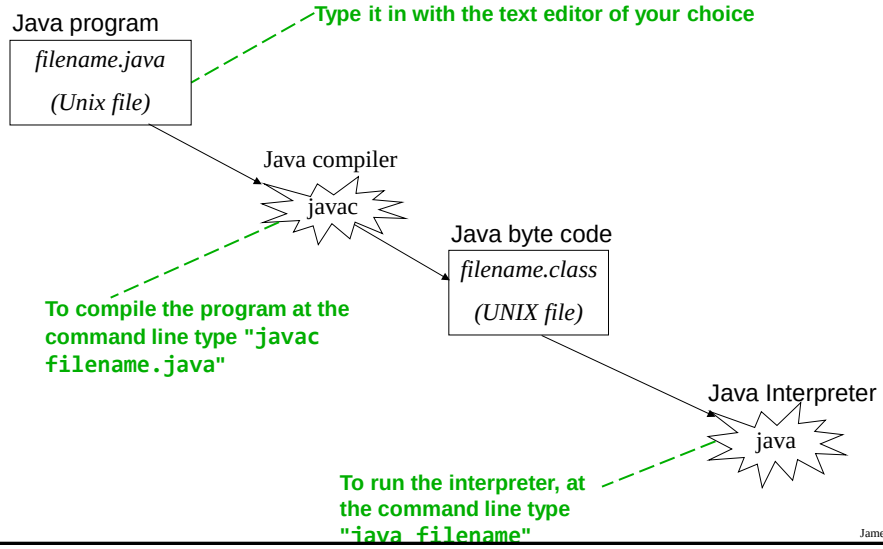
Smallest.java

```
public class
Smallest
{

}
```

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)

```
10000100000001000
00100100000001001
   :          :
```

James Tam

Running The Smallest.java Program

Smallest.class

(Java byte code)

```
10000100000001000
00100100000001001
   :          :
```

java

Type "java Smallest"

(Platform/Operating specific binary)

```
10100111000001000
00100111001111001
   :          :
```

James Tam

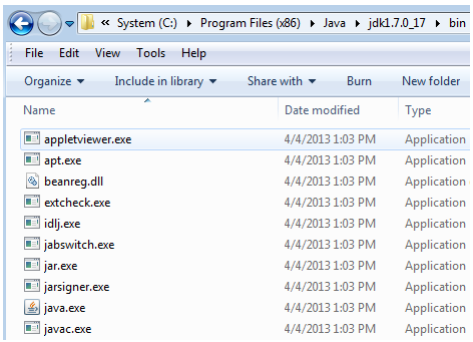
Running The Java Compiler At Home

- After installing Java you will need to indicate to the operating system where the java compiler has been installed ('setting the path').
  - This is similar to Python.
- 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 (see step 5 on the web link)

James Tam

Alternative: Simple But A Hack

- Create your Java programs in the same location as the Java compiler (example)



James Tam

Documentation / Comments

Multi-line documentation

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

- Don't nest this form of documentation (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 (or that portion of the 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 it's 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).

James Tam

Important Note

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

General format	Examples
`Instruction1;`	`int num = 0;`
`Instruction2;`	`System.out.println(num);`
`Instruction3;`	: :
: :	

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>..);
```

- **Examples (online program called "`OutputExample1.java`")**

```
public class OutputExample1
{
    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!
123 num=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):
    incomeSmith = 1000000
```

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;
```

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 can be initialized (set to a starting value) as they're declared:
  ```
  char firstInitial = 'j';
  String firstName = "James";
  int age = 30;
  ```

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 (ASCII values and beyond)
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];`
- 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

- **The complete online program is called "`String1.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 syntactically required or not.
- Example how not to approach (with some languages it's a logic and not a syntax error):

```
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 online example: `FormatttingOutput.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…hurrah!).
- The syntactically enforced unchanging nature of constants is specified with the '`final`' key word

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";`

James Tam

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;
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

The name of the online example is: `Order1.java`

```
public class Order1
{
    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

The name of the 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

James Tam

Getting Text Input (2)

The name of the online example: `MyInput.java`

```
import java.util.Scanner;

public class MyInput
{
    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);
    }
}
```

James Tam

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

Decision Making In Java

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

James Tam

Decision Making: Logical Operators

Logical Operation	Python	Java
AND	`and`	`&&`
OR	`or`	`||`
NOT	`not`	`!`

James Tam

Java Relational Operators

```
if (operand     relational operator     operand)
```

Java operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
==	=	Equal to	5 == 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
!=	≠	Not equal to	x != 5

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 single or multiple statements)

James Tam

The 'Body'

- Single statement body

```
if (num > 0)
    System.out.println("Part of body");
    System.out.println("Not part of body");
System.out.println("Still not part of body");
```

- Multi statement (compound) body

```
if (num > 0)
{
    System.out.println("Part of body");
    System.out.println("Part of body");
}
```

James Tam

Checking For Equality: Strings

- Never use the equality operator when comparing objects (e.g., Strings)
  - More on this later
- Instead the `equals()` and `equalsIgnoreCase()` methods should be used instead.
  - These methods return a true or false value
- Quick example:

```
String s1 = "abc";
String s2 = "abc";
if (s1.equals(s2))
    ...
if (s1.equals(s2) == true)
    ...
```

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 (Java)
If, Elif (Python)

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)

Complete example: SwitchExample.java

```
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 if (gpa == 0)
{
    System.out.println(“F");
}
else
{
    System.out.println("Invalid gpa");
}
```

James Tam

Alternative To Multiple Else-If’s: Switch

- Use when checking for equality of: integer numbers or characters (safest to check only for these types even if later versions of Java support additional types).

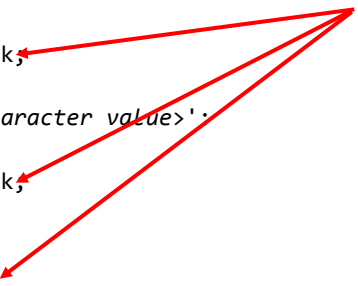
James Tam

Alternative To Multiple Else-If's: Switch

Format (character-based switch):

```
switch (character variable name)
{
     case '<character value>':
          Body
          break;

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



Important! The break is mandatory to separate Boolean expressions (must be used in all but the last).

The break transfers execution out of the switch construct, otherwise cases will 'fall-through'

1 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
}
```

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

James Tam

The 'Break' Statement

- 'Break's is mandatory if cases are to be separated.
- Example:

```
int gpa = 3;
char letter = ' ';
switch (gpa) {
    case 4:
      letter = 'a';
    case 3:
      letter = 'b';
    case 2:
      letter = 'c';
    case 1:
      letter = 'd';
    case 0:
      letter = 'f';
    // Student receives an 'f'!
}
```

As mentioned without a break the switch will execute the first true case and all other cases will 'fall through'

James Tam

Switch: Benefit (Cleaner Code)

- Benefit (when to use):
  - It may produce simpler code than using an if, else-if (e.g., if there are multiple compound conditions)
  - Contrast

```
// Using if
If ((menu == 'a') ||
    (menu == 'A') ||
    (menu == 'N') ||
    (menu == 'n'))
    System.out.println("New
player added");
else if ((menu == 'q') ||
         (menu == 'Q'))
```

```
switch(menu)
{
  case 'a':
  case 'A':
  case 'N':
  case 'n':
    System.out.println("New player \
       added");
    break;

  case 'Q':
  case 'q':
```

James Tam

Switch: Mix and Match Use Of 'Break'

- Name of the online example: `SwitchExample.java` (When to use a `switch`)

```
import java.util.Scanner;

public class SwitchExample
{
    public static void main (String [] args)
    {
        final int FIRST = 0;
        String line;
        char letter;
        int gpa;
        Scanner in = new Scanner (System.in);
        System.out.print("Enter letter grade: ");
```

James Tam

Switch: Mix and Match Use Of 'Break' (2)

```
        line = in.nextLine ();
        letter = line.charAt(FIRST);
        switch (letter)
        {
           case 'A':
           case 'a':
              gpa = 4;
              break;

           case 'B':
           case 'b':
              gpa = 3;
              break;

           case 'C':
           case 'c':
              gpa = 2;
              break;
```

James Tam

Switch: Mix and Match Use Of 'Break' (3)

```
            case 'D':
            case 'd':
               gpa = 1;
               break;

            case 'F':
            case 'f':
               gpa = 0;
               break;

            default:
               gpa = -1;

         }  // End of switch (determining GPA)
         System.out.println("Letter grade: " + letter);
         System.out.println("Grade point: " + gpa);
    }
}
```

James Tam

Loops

Python loops
- for
- while

Java loops
- for
- while

James Tam

While Loops

Format:

```
while (Boolean expression)
{
    Body
}
```

Example:

```
int i = 1;
while (i <= 10)
{
    System.out.println(i);
    i = i + 1;
}
```

```
i = 1
while (i <= 10):
    print(i)
    i = i + 1
```

James Tam

For Loops

Format:

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

Example

```
for (i = 1; i <= 10; i++)
{
    System.out.println(i);
}
```

```
for i in range (1, 11, 1):
   print(i)
```

James Tam

For Loops: Java Vs. Python

- Unlike Python with most languages for loops are generally used as counting (e.g., up down).
- Iterating through other series (such as lines in a file) is not possible.
- Python example not possible in other languages

```
inputFile = open("input.txt", "r")
for line in inputFile:
    print(line)
```

- In Java however the **loop control update** can be most any mathematical expression (even randomly assigned).

```
for (i = 1; i <= 100; i = i * 5)
```

James Tam

For Loops: Java Vs. Python (2)

- Also note in Java that the **stopping boundary** is explicit.

```
for (i = 1; i <= 10; i++)
-Vs.
for i in range (1, 11, 1):
```

James Tam

Common Mistake: Branches/Loops

- Forgetting braces and that single statement bodies are specified by the first semi-colon.
- (Partial) examples:

```
while (i < 10)
    System.out.println(i);
    i = i + 1;
```

Body (logic error)

```
while (i < 10);
{
    System.out.println(i);
    i = i + 1;
}
```

Body

James Tam

Many Pre-Created Classes Have Been Created

- Rule of thumb of real life: 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.
- Note: for some assignments you may have to implement all features yourself rather than use pre-written code.
  - You may receive little or no credit otherwise.
- The Java API is Sun Microsystems's collection of pre-built Java classes:
  - http://java.sun.com/javase/8/docs/api/

James Tam

Example: Generating Random Numbers (Probabilities)

- Name of the (more complete example): `DiceExample.java`

```
import java.util.Random;
public class DiceExample
{
    public static void main(String [] args)
    {
        final int SIDES = 6;
        Random generator = new Random();
        int result = -1;
        result = generator.nextInt(SIDES) + 1;
        System.out.println("1d6: " + result);

        result = generator.nextInt(SIDES) + 1;
        result = result + generator.nextInt(SIDES) + 1;
        result = result + generator.nextInt(SIDES) + 1;
        System.out.println("3d6: " + result);
    }
}
```

James Tam

Arrays

- They are similar to Python lists.
  - Specified with square brackets
  - Indexed from 0 to (number elements-1)
- Some differences:
  - All elements must be of the same type e.g., array of integers cannot mix and match with floats
  - Python has methods associated with lists although an array in Java has a '`length`' attribute associated with it.
  - Arrays cannot be dynamically resized (new array must be created).

James Tam

Creating An Array

- **Format**:
  - <*type*> []$_1$ <*name*> = new <*type*> [<*Number of elements*>];

- **Example (common approach)**:

```
final int MAX = 100;
int [] grades = new int[MAX];
```

- **Example (Fixed size array declared and initialized – rarely used approach)**:

```
int [] array = {1,2,3};
```

1 Each dimension must be specified by a set of square brackets e.g., two dimensional array requires two sets of brackets

James Tam

Arrays: Complete Example

- Name of the (more complete example): `GradesExample.java`

```
public class GradesExample
{
    public static void main(String [] args)
    {
        final int MAX = 10;
        int [] grades = new int[MAX];
        int i = 0;
        Random generator = new Random();
```

James Tam

Arrays: Complete Example (2)

```
        for (i = 0; i < MAX; i++)
        {
            grades[i] = generator.nextInt(101);
        }

        for (i = 0; i < grades.length; i++)
        {
            System.out.println("Element #" + i + " grade " +
              grades[i]);
        }
    }
}
```

Unlike Python lists you cannot pass an entire Java array in order to display the elements:

```
System.out.println(grades);
```

```
[I@c351f6d
```

James Tam

After This Section You Should Now Know

- The basic structure required for 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
- Formatting output with the field width, precision and escape codes (elective)
- Converting between types using the casting operator
- What are the common mathematical and logical operators and how they work
- The structure and syntax of decision making and looping constructs

James Tam

After This Section You Should Now Know (2)

- How to generate random numbers
- How to create and work with Java arrays

James Tam

Copyright Notification

- "Unless otherwise indicated, all images in this presentation are used with permission from Microsoft."

James Tam