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: Write Once, Run Anywhere

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

Mac user running Safari

Click on link to Applet

Byte code is downloaded

Web page stored on Unix server

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

Windows user running Internet Explorer

Byte code (part of web page)

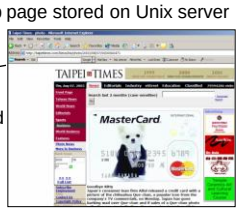
James Tam

Java: Write Once, Run Anywhere

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

Mac user running Safari

Web page stored on Unix server



Click on link to Applet

Byte code is downloaded

Windows user running Internet Explorer

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)
http://homepage.mac.com/aberfield/dmj/

Kung Fu Panda 2: THQ

Examples of mobile Java games: http://www.mobilegamesarena.net

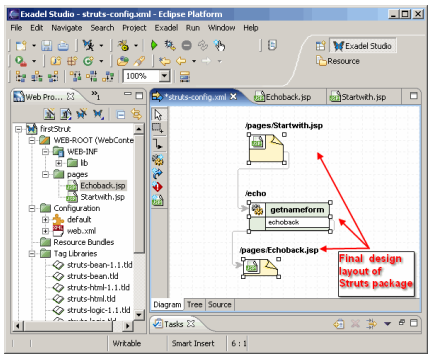
James Tam

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/

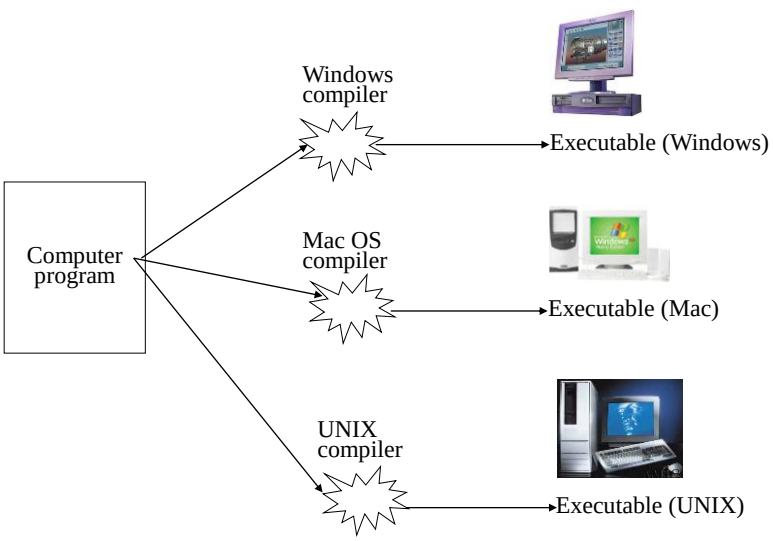
James Tam

JT's Note: IDE's

- Even more so than Python there are 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.
  - 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'.
- Bottom line: if you have problems with the IDE then you will likely be on your own.

James Tam

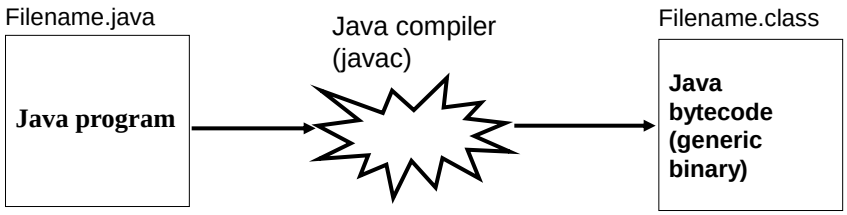
Compiled Programs With Different Operating Systems



James Tam

A High Level View Of Translating/Executing Java Programs

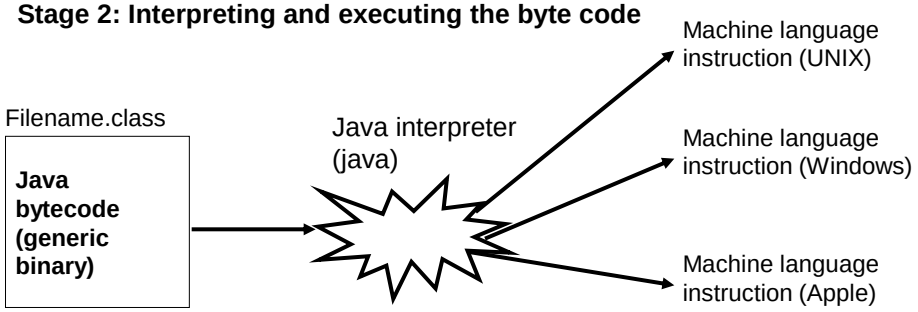
Stage 1: Compilation



James Tam

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

Stage 2: Interpreting and executing the byte code



James Tam

Which Java?

- Java 1.6 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) – only good 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 (don't use versions past 1.6).
  - 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).

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

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

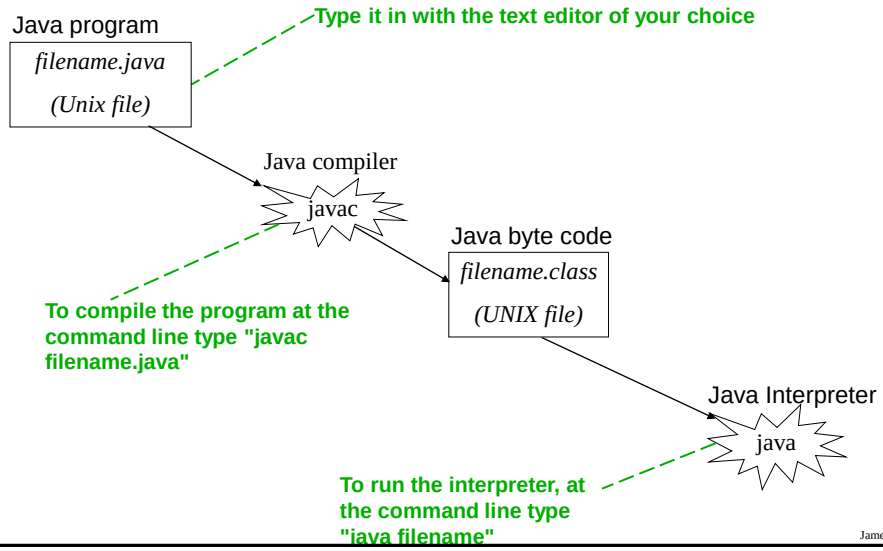
Smallest Compilable And Executable Java Program

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

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

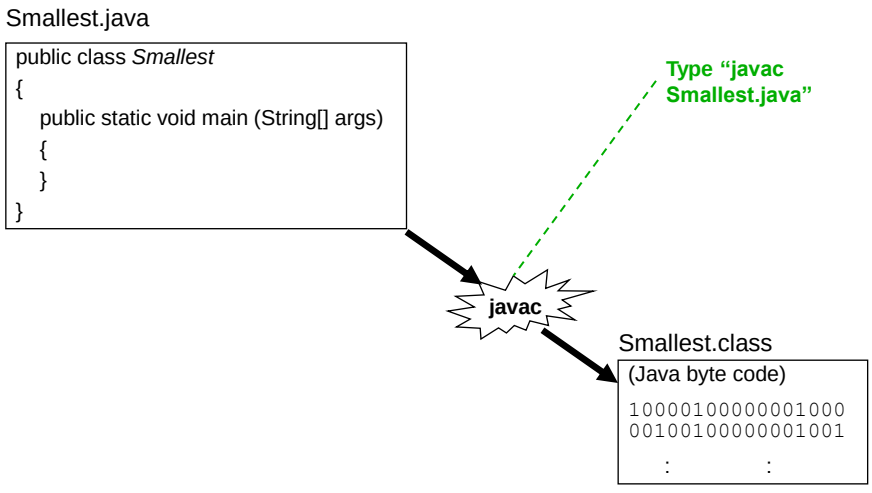
James Tam

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



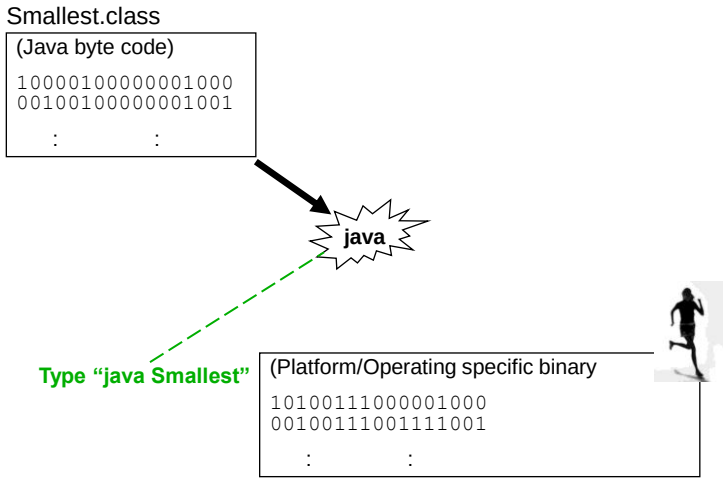
James Tam

Compiling The Smallest Java Program



James Tam

Running The Smallest Java Program



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 (see step 5 on the web link)

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

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

- **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;   // More on this shortly
        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
\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.
- Using variables
  - Only after a variable has been declared can it be used.
  - 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!

```
income = 25000
if (winLottery):
   incom = 1000000
```

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

Java

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

```
int income = 25000;
if (winLottery)
   incom = 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 myFirstInitial;

- Variables can be initialized (set to a starting value) as they're declared:
  char 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 (rare)
double	64 bit signed real number (compiler default)
char	16 bit Unicode character (ASCII values and beyond)
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

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 (under some circumstances 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

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

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

- 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)

- 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("%6d", num);
      System.out.printf("%6.2f", price);
   }
}
```

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!).

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 (more later)
         // 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

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++);
   }
}
```

James Tam

Post/Pre Operators (2)

The name of the online example is: Order2.java

```
public class Order2
{
   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 Operator/Order/Associativity

The name of the online example: Unary_Order3.java

```
public class Unary_Order3.java
{
   public static void main (String [] args)
   {
      int num = 5;
      System.out.println(num);
      num = num * -num;
      System.out.println(num);
   }
}
```

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.
- 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

```
public class Casting {
   public static void main(String [] args) {
      int num1;
      double num2;
      String str1;
      num2 = 1.9;
      str1 = "123";
      num1 = (int) num2;   // Cast needed to explicitly convert
      System.out.println(num1 + " " + num2);
      num2 = num1;  // Cast not needed: going from more to less
      System.out.println(num1 + " " + num2);
   }
}
```

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> ();
}
```

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 str1;
      int num1;
      Scanner in = new Scanner (System.in);
      System.out.print ("Type in an integer: ");
      num1 = in.nextInt ();
      in.nextLine ();
      System.out.print ("Type in a line: ");
      str1 = in.nextLine ();
      System.out.println ("num1:" +num1 +"\t str1:" + str1);
   }
}
```

James Tam

Useful Methods Of Class Scanner[1]

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

1 Online documentation: http://java.sun.com/javase/6/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

Reading A Single Character

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

```
import java.util.Scanner;
public class MyInputChar
{
   public static void main (String [] args)
   {
      final int FIRST = 0;
      String selection;
      Scanner in = new Scanner (System.in);
      System.out.println("GAME OPTIONS");
      System.out.println("(a)dd a new player");
      System.out.println("(l)oad a saved game");
      System.out.println("(s)ave game");
      System.out.println("(q)uit game");
      System.out.print("Enter your selection: ");
```

James Tam

Reading A Single Character (2)

```
        selection = in.nextLine ();
        System.out.println ("Selection: " + selection.charAt(FIRST));
    }
}
```

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

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

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");
}
```

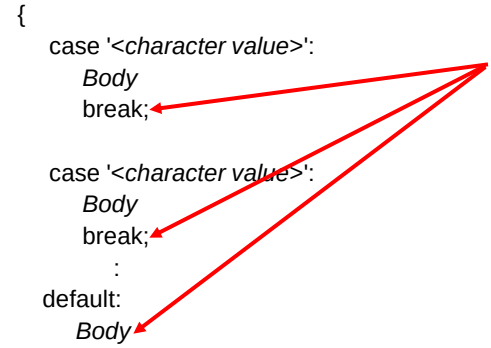
James Tam

Alternative To Multiple Else-If's: Switch

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)

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

Switch: When To Use/When Not To Use

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

James Tam

Switch: When To Use/When Not To Use (2)

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

```
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: When To Use/When Not To Use (3)

```
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: When To Use/When Not To Use (4)

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

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

  default:
    gpa = -1;

}
System.out.println("Letter grade: " + letter);
System.out.println("Grade point: " + gpa);
  }
}
```

James Tam

Switch: When To Use/When Not To Use (5)

- When a switch can't be used:
  - For data types other than characters or integers (Java 1.6 and earlier)
  - Boolean expressions that aren't mutually exclusive:
    - As shown a switch can replace an 'if, else-if' construct
    - A switch cannot replace a series of 'if' branches).
  - Example when not to use a switch:

```
if (x > 0)
    System.out.print("X coordinate right of the origin");
If (y  > 0)
    System.out.print("Y coordinate above the origin");
```

  - Example of when not to use a switch (Java 1.6):

```
String name = in.readLine()
switch (name)
{

}
```

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 (*Boolean 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

Post-Test Loop: Do-While

- Recall: Post-test loops evaluate the Boolean expression after the body of the loop has executed.
- This means that post test loops will execute one or more times.
- Pre-test loops generally execute zero or more times.

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

Common Mistake: Branches/Loops

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

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

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

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.
- The Java API is Sun Microsystems's collection of pre-built Java classes:
  - http://java.sun.com/javase/6/docs/api/

James Tam

Extras For Assignments

- Command arguments
- Getting input from files

James Tam

Command Line Arguments

- Sometimes programs can receive all input information as the program is run.
- Examples include operating system commands:

  "ls –a –l" (UNIX)

  "notepad.exe c:\temp\testfile.txt" (DOS/Windows: assuming the current directory is where Notepad resides)

Name of the program	**Inputs given to the program (command line arguments)**
ls	-a –l
notepad.exe	c:\temp\testfile.txt

James Tam

Command Line Arguments In Java

•Name of the online example: CommandLineInputs.java

```
public class CommandLineInputs
{
   public static void main(String [] args)
   {
      if (args.length > 0)
      {
         System.out.print("First input after file name: ");
         System.out.println(args[0]);
      }
      for (int i = 0; i < args.length; i++)
      {
         System.out.println(args[i]);
      }
   }
}
```

James Tam

Getting File Input, Version 1 (Just Include In Your Assignment)

•Name of the online example: FileInput1.java

```
import java.io.*;
public class FileInput1 {
   public static void main (String [] args) throws IOException {
      FileReader fr = null;
      BufferedReader br = null;
      String filename = "input.txt";
      String lineFromFile = null;
      fr = new FileReader(filename);
      br = new BufferedReader(fr);
      lineFromFile = br.readLine();     // Reads line of input
      while(lineFromFile != null) {      /* Checks for EOF  */
         System.out.println(lineFromFile);
         lineFromFile = br.readLine();
      }
   }
}
```

James Tam

Getting File Input, Version 1 (Just Include In Your Assignment)

•Name of the online example: FileInput1.java

```
import java.io.*;
public class FileInput1 {
   public static void main (String [] args) throws IOException {
      FileReader fr = null;
      BufferedReader br = null;
      String filename = "input.txt";
      String lineFromFile = null;
      fr = new FileReader(filename);
      br = new BufferedReader(fr);
      lineFromFile = br.readLine();     // Reads line of input
      while(lineFromFile != null) {      /* Checks for EOF  */
         System.out.println(lineFromFile);
         lineFromFile = br.readLine();
      }
   }
}
```

James Tam

Getting File Input, Version 2

•Name of the online example: FileInput2.java (converts from String to other types of data).

```
import java.io.*;
public class FileInput2
{
   public static void main (String [] args) throws Exception
   {
      FileReader fr = null;
      BufferedReader br = null;
      String filename = "input2.txt";
      String lineFromFile = null;
      int num1 = 0;
      double num2 = 0;
      fr = new FileReader(filename);
      br = new BufferedReader(fr);
      lineFromFile = br.readLine();
```

James Tam

Getting File Input, Version 2 (2)

```
        // Converts from String to integer (make sure String is really all integer)
        num1 = Integer.parseInt(lineFromFile);
        num1 = num1 * 2;

        lineFromFile = br.readLine();
        // Converts from String to double (make sure String is really all real)
        num2 = Double.parseDouble(lineFromFile);
        num2 = num2 + 1;

        lineFromFile = br.readLine();
        System.out.println(num1);
        System.out.println(num2);
        System.out.println(lineFromFile);


    }
}
```

James Tam

After This Section You Should Now Know

- 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
- Formatting output with the field width, precision and escape codes
- Converting between types using the casting operator
- What are the common Java operators and how they work
- The structure and syntax of decision making and looping constructs

James Tam