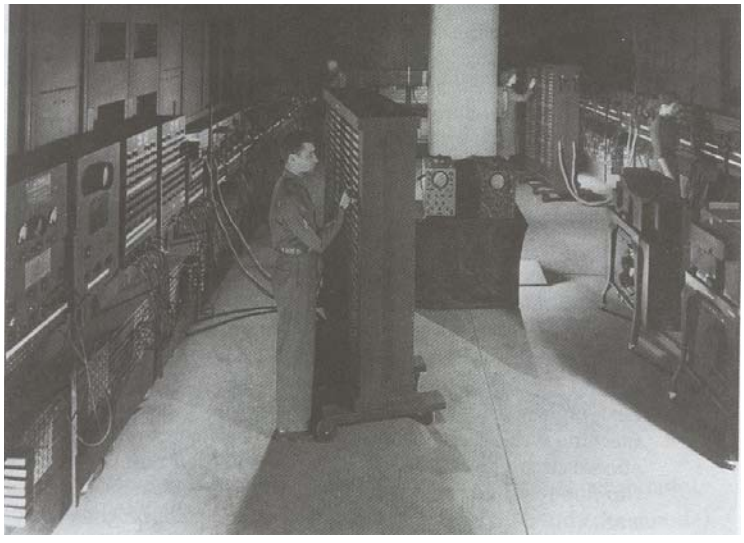
Introduction To Java Programming

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

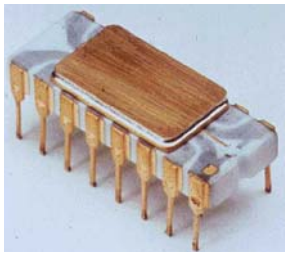
Java: History

•Computers of the past



Java: History (2)

•The invention of the microprocessor revolutionized computers



Intel microprocessor

Commodore Pet microcomputer

Java: History (3)

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

Java History (4)

•Sun Microsystems funded an internal research project "Green" to investigate this opportunity.

- Result: A programming language called "Oak"

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

Wav file from "The Simpsons" © Fox, Image from the website of Sun Microsystems

Java History (5)

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

Java: History (6)

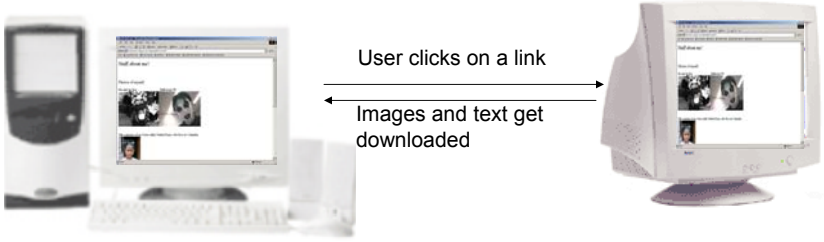
- The concept of intelligent devices didn't catch on.
- Project Green and work on the Java language was nearly canceled.

Java: History (7)

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

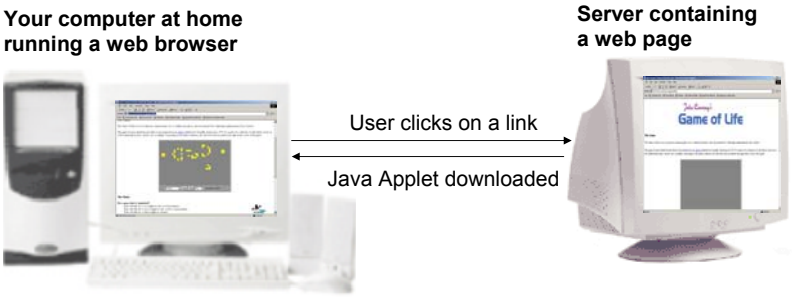
Your computer at home running a web browser

Server containing a web page



Java: History (8)

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

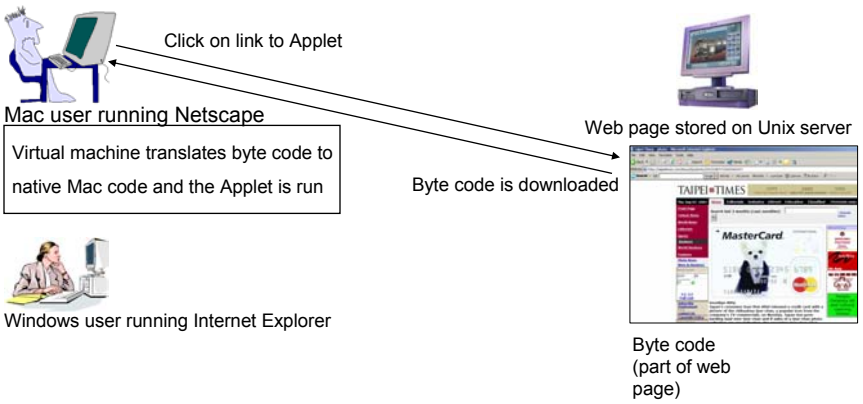


Java version of the Game of Life: http://www.bitstorm.org/gameoflife/

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

Java: Write Once, Run Anywhere

- Consequence of Java's history: platform-independence



Java: Write Once, Run Anywhere

•Consequence of Java's history: platform-independent

Mac user running Netscape

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

Java: Write Once, Run Anywhere (2)

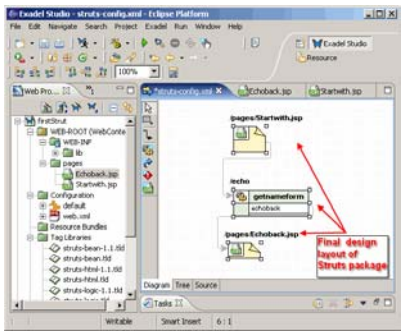
•But Java can also create standard (non-web based) programs

Dungeon Master (Java version)

http://www.cs.pitt.edu/~alandale/dmjava/

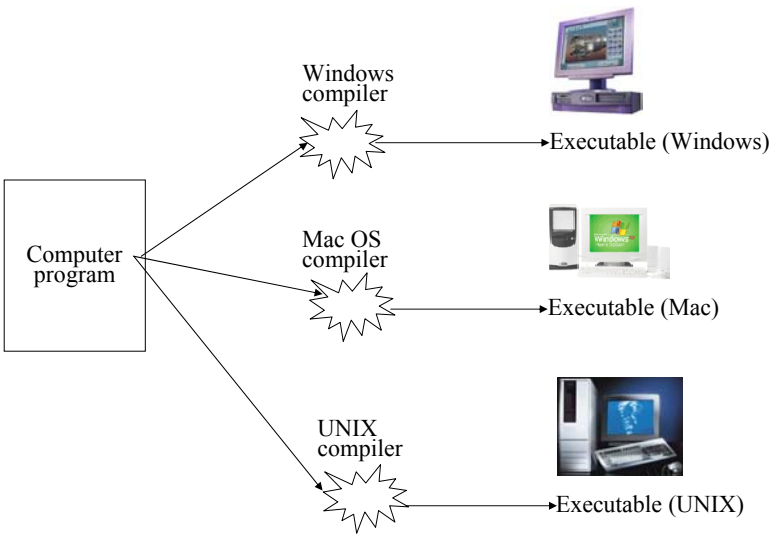
Java: Write Once, Run Anywhere (3)

- 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.

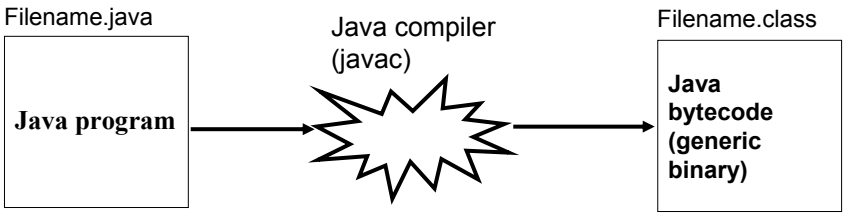


1 For more information: http://www.eclipse.org/downloads/

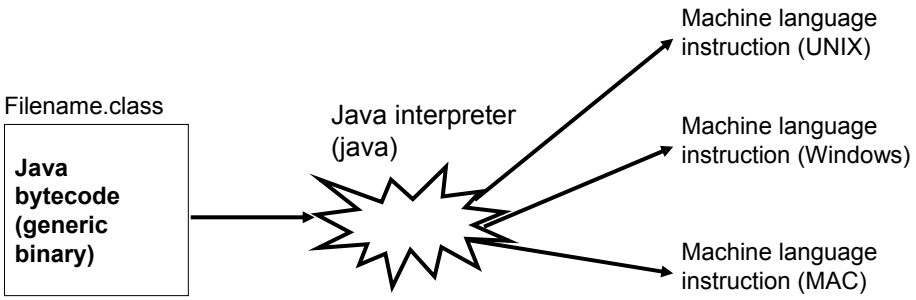
Compiled Programs With Different Operating Systems



A High Level View Of Translating/Executing Java Programs



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



Which Java?

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

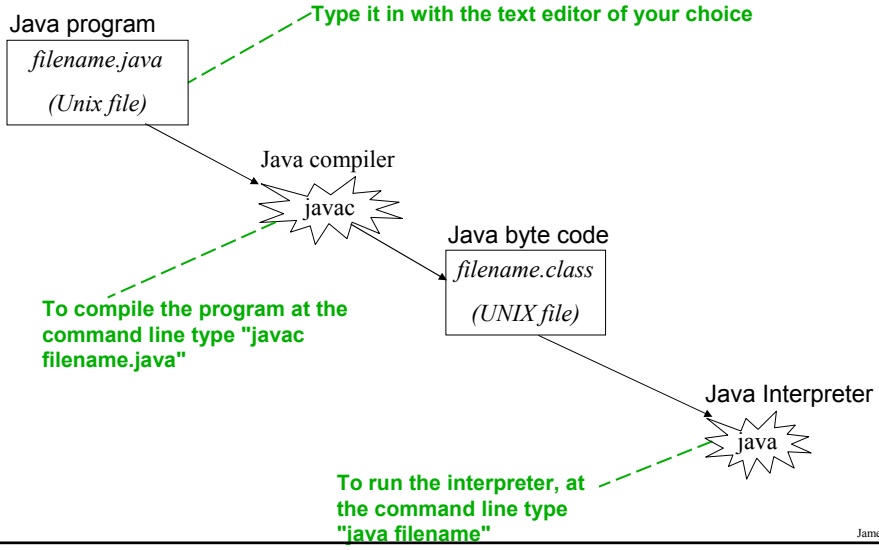
http://java.sun.com/javase/downloads/index.jsp

Smallest Compilable And Executable Java Program

The name of the online example is: Smallest.java

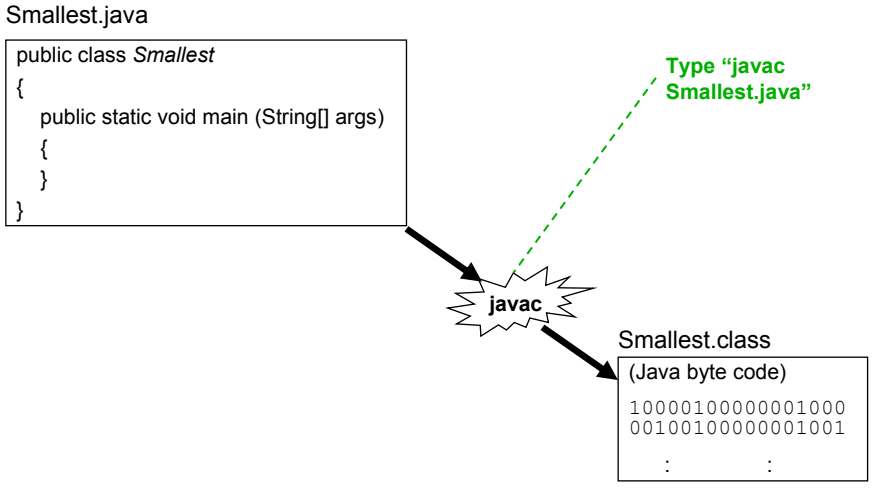
```
public class Smallest
{
   public static void main (String[] args)
   {
   }
}
```

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



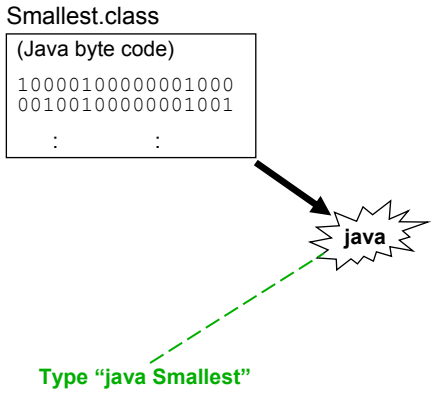
James Tam

Compiling The Smallest Java Program



James Tam

Running The Smallest Java Program



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 (see step 5 in the link below):
  - http://java.sun.com/j2se/1.4.2/install-windows.html

Documentation / Comments

Java

- Multi-line documentation
  - /* Start of documentation
  - */ End of documentation
- Documentation for a single line
  - //Everything until the end of the line is a comment

Review: What Should You Document

- 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).

Java Output

- **Format:**

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

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

```
System.out.println("Good-night gracie!");
System.out.print(num);
System.out.println("num=" +num);
```


Output : Some Escape Sequences For Formatting

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

Declaring Variables

- **Format:**
  - It's the same structure that's used with 'C' variables.

Some Built-In Types Of Variables In Java

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

Location Of Variable Declarations

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

         << Program statements >>
               :          :

   }
}
```


Java Constants

Format:

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

Example:

final int SIZE = 100;

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 >>
                        :          :

   }
}
```


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";

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		

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

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

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

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

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	

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++);
   }
}
```


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);
   }
}
```

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> ();
}
```


Getting Text Input (2)

- **Example: The name of the online example is MyInput.java**

```
import java.util.Scanner;

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

Useful Methods Of Class Scanner[1]

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

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

Decision Making In Java

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

Decision Making: Logical Operators

Logical Operation	C	Java
AND	&&	&&
OR	||	||
NOT	!	!

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");
}
```

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");
```


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
```

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");
}
```


If, Else-If (2)

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

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

Format (character-based switch):

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

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

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

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 variablein the brackets can be a byte, char, short, int or long

Loops

Java Pre-test loops

- For
- While

Java Post-test loop

- Do-while

While Loops

Format:

```
while (Expression)
   Body
```

Example:

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

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);
}
```


Do-While Loops

Format:

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

Example:

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

Many Pre-Created Classes Have Been Created

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

After This Section You Should Now Know

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