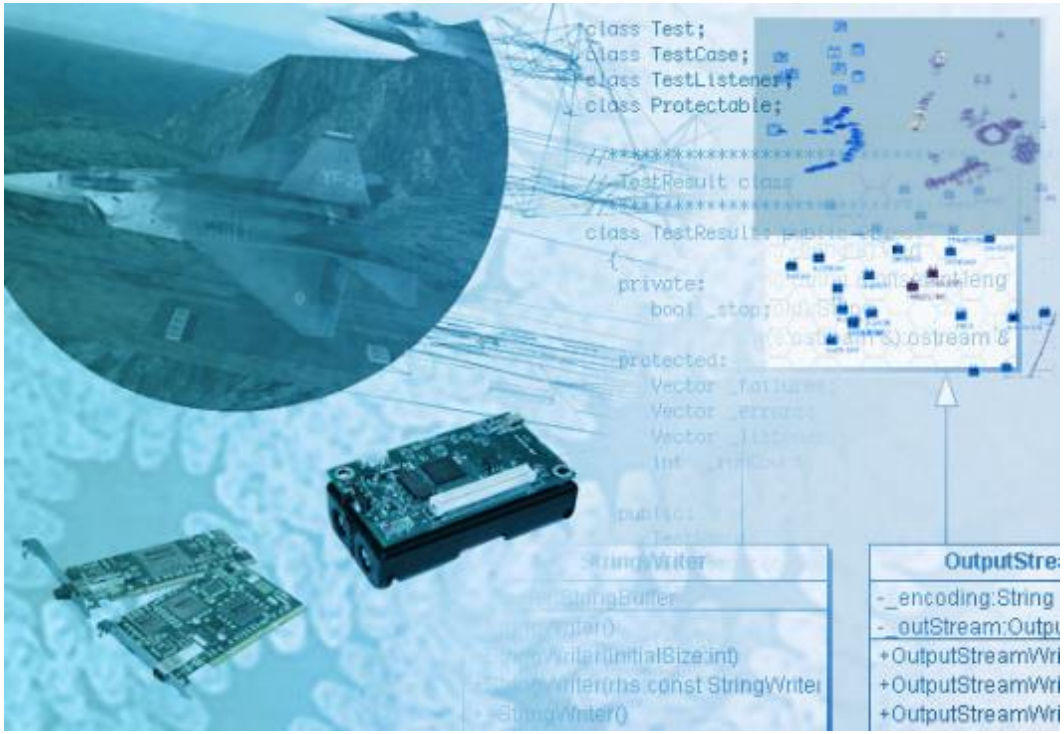
CSYE 6200

CONCEPTS OF OBJECT-ORIENTED DESIGN
FALL 2016

MARK G. MUNSON



WHY ARE WE HERE?

- Provide an introduction to Object Oriented Design (OOD) concepts
- To build a solid understanding of software development using Java as an OO language
- Know when Java is the best design choice (and when it isn't)
- Understand the tradeoffs of building complex systems using Java (and/or some other languages)

ADMINISTRATION

- **What do you need?**
  - Computer (Windows, Mac, or Linux)
  - Books – see the syllabus
  - Prior programming experience is not required

ADMINISTRATION

- **Assistants - Each section has an assistant. Assistants will be available on-campus to answer quick questions.**
  - Ranjan Ramanujam Vijayaraghavan (Ranjan R.V.)
    - ramanujamvijayara.r@husky.neu.edu
  - Christoper Dsouza
    - dsouza@husky.neu.edu

ADMINISTRATION

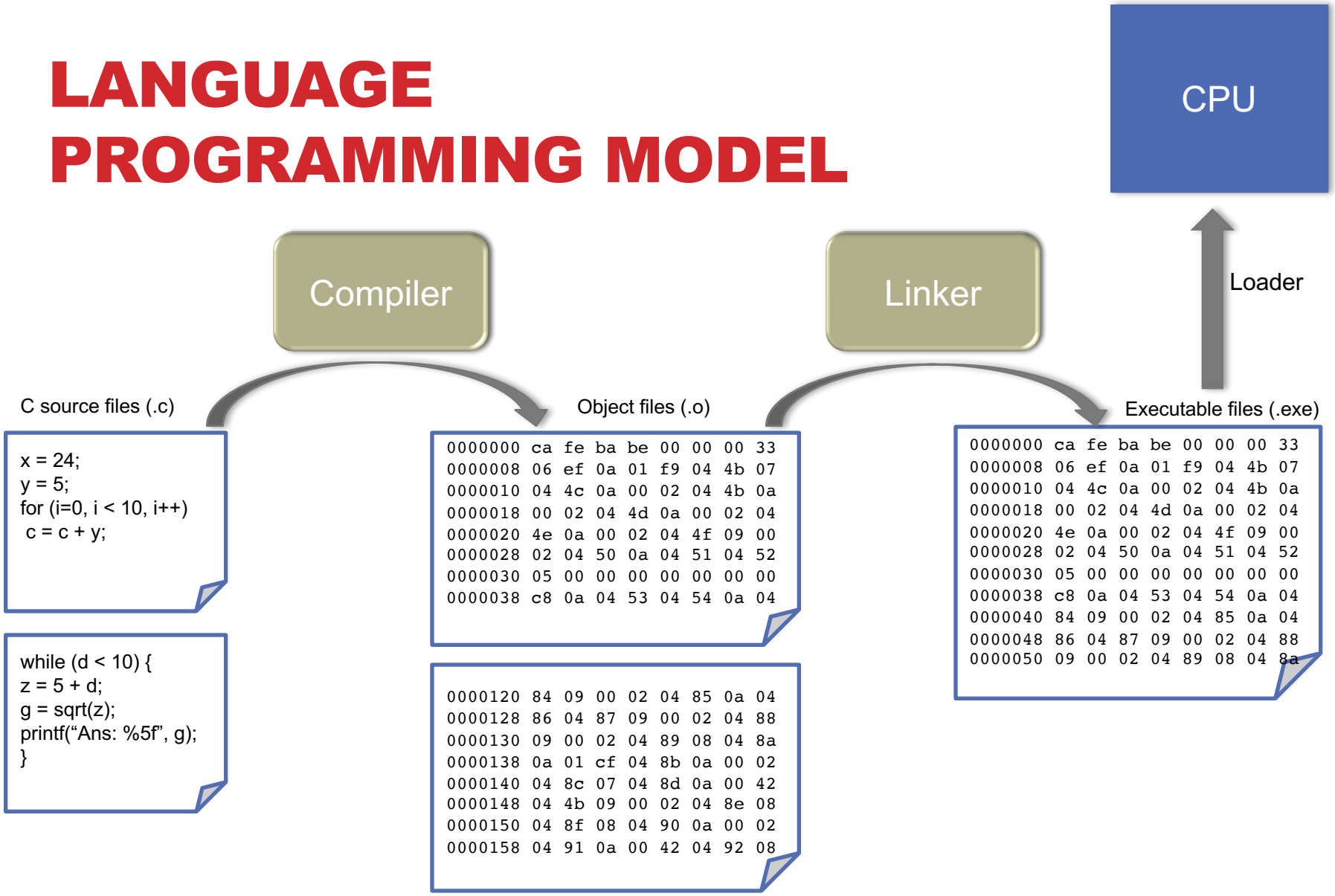
- **Expectations**
  - Full attendance
  - Auditing is not allowed
  - Complete assignments – there will be several
  - What is the Final Challenge?

THE LECTURE

- **Why Java**
- **Installing Java / Other tools**
- **Coding**
  - Hello World
- **Variables**
- **Control Statements**
  - For-demo
- **UML**
  - Use Cases

WHY JAVA?

LANGUAGE PROGRAMMING MODEL

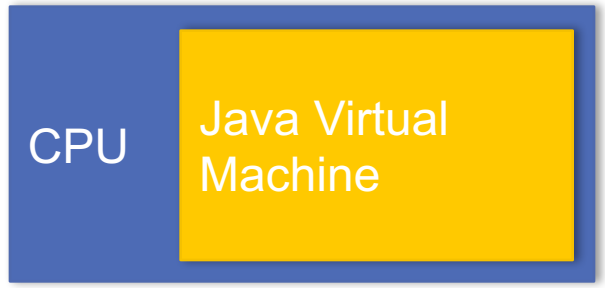


A MODEL PROBLEM

Problem: Too many processors, operating systems, and architectures – each platform requires customization – and more work to test, package and distribute

- **Chipsets**
  - Intel x86, Motorola 68K, WD 65C816
  - AMD Opteron, Sun SPARC, IBM PowerPC
  - ARM Atom
- **Operating Systems (Minicomputers and Personal Computers)**
  - Windows DOS/NT, Apple MacOS
  - IBM AIX, HP HP/UX, Sun Solaris, SGI Irix
  - Linux

JAVA LANGUAGE PROGRAMMING MODEL



Java Compiler

Class Archive

Loader

Java source files (.java)

```
class dog {
  doCalc() {
   int c = 0;
   for (int i=0, i < 10, i++)
   c = c + y;
  return c;
  }
}
```

```
class cat {
public void main(String arv[]) {
   double d = 3;
   while (d < 10) {
   double z = 5 + d;
   double g = Math.sqrt(z);
   System.out.printf("Ans: %5f", g);
}
```

class files (.class)

```
0000000 ca fe ba be 00 00 00 33
0000008 06 ef 0a 01 f9 04 4b 07
0000010 04 4c 0a 00 02 04 4b 0a
0000018 00 02 04 4d 0a 00 02 04
0000020 4e 0a 00 02 04 4f 09 00
0000028 02 04 50 0a 04 51 04 52
0000030 05 00 00 00 00 00 00 00
0000038 c8 0a 04 53 04 54 0a 04
```

```
0000120 84 09 00 02 04 85 0a 04
0000128 86 04 87 09 00 02 04 88
0000130 09 00 02 04 89 08 04 8a
0000138 0a 01 cf 04 8b 0a 00 02
0000140 04 8c 07 04 8d 0a 00 42
0000148 04 4b 09 00 02 04 8e 08
0000150 04 8f 08 04 90 0a 00 02
0000158 04 91 0a 00 42 04 92 08
```

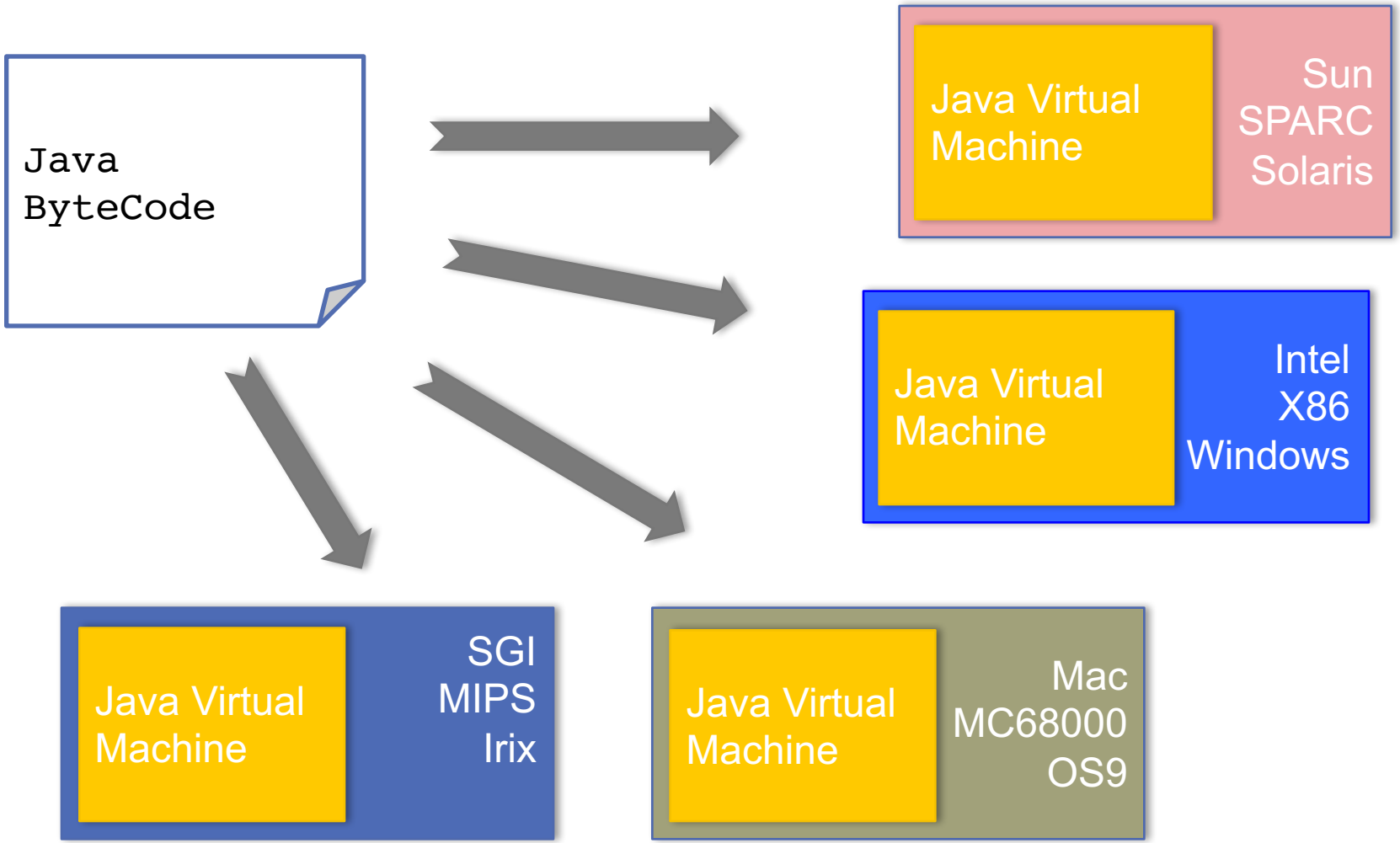
ByteCode

Executable files (.jar)

```
0000000 ca fe ba be 00 00 00 33
0000008 06 ef 0a 01 f9 04 4b 07
0000010 04 4c 0a 00 02 04 4b 0a
0000018 00 02 04 4d 0a 00 02 04
0000020 4e 0a 00 02 04 4f 09 00
0000028 02 04 50 0a 04 51 04 52
0000030 05 00 00 00 00 00 00 00
0000038 c8 0a 04 53 04 54 0a 04
0000040 84 09 00 02 04 85 0a 04
0000048 86 04 87 09 00 02 04 88
0000050 09 00 02 04 89 08 04 8a
```

A Zip Archive

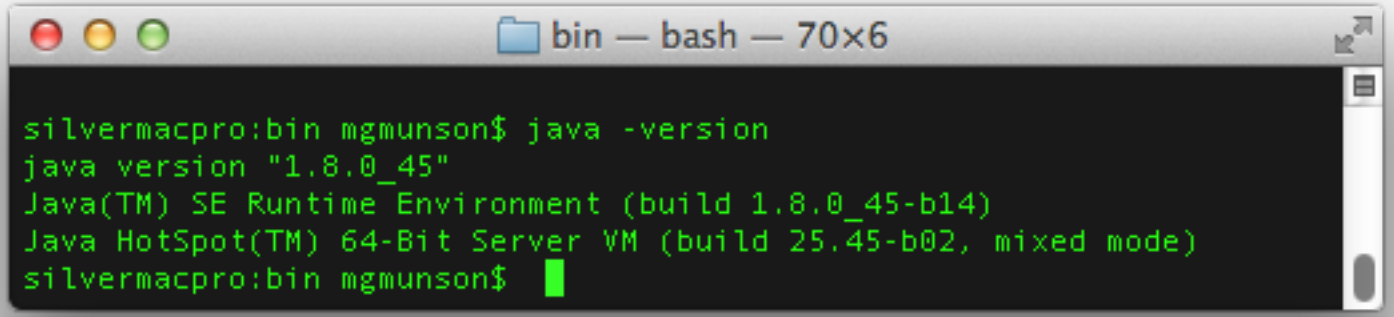
JAVA PORTABILITY : A SINGLE FILE SOLUTION



INSTALLING JAVA

INSTALLING JAVA

- Go to the Sun Java Developer site to install the latest 64-bit version of the Java Software Developer's Kit (JDK)
- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- You may install the NetBeans Integrated Development Environment (IDE) as well
- Run the installer, and use the default options
- Check your installation by running the 'java' command from a terminal window



INSTALLING JAVA (CONT.)

Java SE – Downloads | Oracle Technology Network | Oracle

www.oracle.com/technetwork/java/javase/downloads/index.html

ORACLE

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partners About OTN

Oracle Technology Network > Java > Java SE > Downloads

- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Downloads

Java DOWNLOAD — Java Platform (JDK) 8u60

NetBeans DOWNLOAD — NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 8u60
This releases includes support for ARMv8 processors, Nashorn enhancements, and improvements to Deployment Rule Set functionality.
JDK for ARM releases are now available on the same page as the downloads for other platforms.
Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations
- Readme Files
  - JDK ReadMe

JDK DOWNLOAD

Server JRE DOWNLOAD

JRE DOWNLOAD

Java SDKs and Tools

- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

Java Resources

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net
- Developer Training
- Tutorials
- Java.com

OTHER TOOLS

- **Windows users:**
  - Install Cygwin to emulate a Unix-style console
- **Version control:**
  - Install a git client
  - On Windows: TortiseGit allows git commands to be used directly from the GUI

LET'S CODE

HELLO WORLD

Example.java UNREGISTERED

Example.java

```
1   /*
2   This is an Example program
3   */
4
5   class Example {
6       // A Java program begins with a call to main()
7
8       public static void main(String args[]) {
9           System.out.println("Hello World");
10      }
11  }
```

Line 11, Column 2 Tab Size: 4 Java

Sublime Text 2

SOME CLASS RULES

- **Java is Object Oriented, so procedures must live within a 'class'**
- **Class names start with an upper case letter (i.e. Example)**
- **Each class is stored in a file with the same name**
  - Example.java contains the Example class definition
- **Each class may contain a start point called 'main'**

BUILD IT

Use ‘dir’ command with Windows DOS
Use ‘ls’ with OSX and Linux

```
> ls
Example.java
> java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
> javac Example.java
> ls
Example.class  Example.java
> java Example
Hello World
```

WHAT WE DID

Hello World

CPU

Java Virtual Machine

Java Compiler

javac

java

Example.java

Example.class

```
/*
This is an Example program
*/

class Example {
    // A Java program begins with a call to main()

    public static void main(String args[]) {
        System.out.println("Hello World");
    }
}
```

```
0000000 ca fe ba be 00 00 00 34 00 1d 0a 00 06 00 0f 09
0000010 00 10 00 11 08 00 12 0a 00 13 00 14 07 00 15 07
0000020 00 16 01 00 06 3c 69 6e 69 74 3e 01 00 03 28 29
0000030 56 01 00 04 43 6f 64 65 01 00 0f 4c 69 6e 65 4e
0000040 75 6d 62 65 72 54 61 62 6c 65 01 00 04 6d 61 69
0000050 6e 01 00 16 28 5b 4c 6a 61 76 61 2f 6c 61 6e 67
0000060 2f 53 74 72 69 6e 67 3b 29 56 01 00 0a 53 6f 75
0000070 72 63 65 46 69 6c 65 01 00 0c 45 78 61 6d 70 6c
0000080 65 2e 6a 61 76 61 0c 00 07 00 08 07 00 17 0c 00
0000090 18 00 19 01 00 0b 48 65 6c 6c 6f 20 57 6f 72 6c
00000a0 64 07 00 1a 0c 00 1b 00 1c 01 00 07 45 78 61 6d
00000b0 70 6c 65 01 00 10 6a 61 76 61 2f 6c 61 6e 67 2f
00000c0 4f 62 6a 65 63 74 01 00 10 6a 61 76 61 2f 6c 61
00000d0 6e 67 2f 53 79 73 74 65 6d 01 00 03 6f 75 74 01
00000e0 00 15 4c 6a 61 76 61 2f 69 6f 2f 50 72 69 6e 74
00000f0 53 74 72 65 61 6d 3b 01 00 13 6a 61 76 61 2f 69
0000100 6f 2f 50 72 69 6e 74 53 74 72 65 61 6d 01 00 07
0000110 70 72 69 6e 74 6c 6e 01 00 15 28 4c 6a 61 76 61
```

VARIABLES

- **A variable is a named location where a program stores values**
  - Each class defines areas to store information
- **Each variable has three pieces of information**
  - *Name:* What this variable is called
  - *Type:* The kind of data that is stored (integers, floats, characters, etc.)
  - *Value:* The value at any period in time

VARIABLE NAMING

- **May start with any letter of the alphabet, an underscore, or a dollar sign**
  - valid: **boxCount3** invalid: **23box**
- **Other characters may be a number, letter, an underscore or a dollar sign**
  - Valid: **a$** or **_counter** or **box_count_3**
- **Upper and lower case characters are different**
  - MyVal is different than myVal
- **None of the fifty keywords (defined in Table 1-1) may be used in a variable name**
- **Programming style guides may be used to promote code consistency among developers.**
  - For example, most programmers start variables with a lowercase letter and capitalize each word:
    capitalizeEachWordButTheFirst

EXAMPLE 3

Example3.java UNREGISTERED

Example3.java

```java
/*
 This program illustrates the differences
 between int and double.
 Filename: Example3.java
 */

class Example3 {
    public static void main(String args[]) {
        int var; // this declares an int variable
        double x; // this declares a floating point value

        var = 10; // assign var the value 10
        x = 10; // assign x the value of 10.0

        System.out.println("Original value of var: " + var);
        System.out.println("Original value of x: " + x);

        // Now divide both by 4
        var = var / 4;
        x = x / 4;

        System.out.println("var after division: " + var);
        System.out.println("x after division: " + x);

    }
}
```

Line 23, Column 45 Tab Size: 4 Java

CONTROL STATEMENTS

- **The if Statement**
  - Allows code to selectively execute parts of a program

```
if (condition) statement;
or
if (condition) { statements }
```

Examples

```
if (c < 10) System.out.println("c is less than ten");
if (c == 5) {
     System.out.println("c is equal to five");
     c = c+1;
     System.out.println("c plus one is : " + c);
}
```

CONTROL STATEMENTS (CONT.)

- **The for Loop**
  - Repeatedly execute a sequence of code

```
for (initialization; condition; iteration) statement;
   or
for (initialization; condition; iteration ){statements}
```

  - Example

```
int count;
for (count = 0; count < 5; count = count + 1)
   System.out.println("This is count " + count);
```

EXAMPLE FORDEMO

ForDemo.java UNREGISTERED

ForDemo.java

```
1   /*
2   Demonstration of the for loop
3   Filename ForDemo.java
4   */
5
6   class ForDemo {
7       public static void main(String args[]) {
8       int count;
9       for (count = 0; count < 5; count = count + 1)
10          System.out.println("This is count: " + count);
11
12      System.out.println("Done!");
13      }
14  }
```

Line 13, Column 4 Tab Size: 4 Java

USE CASES

USE CASES

- A Use Case is a modeling technique that is used to describe either an existing or proposed system.

- The primary components within a use case are a description of how a system is used, and the identification of actors in the system.

USE CASES

- **Use Case provide the following benefits**
  1. Provide a clear description of what the system will do
  2. Help to describe the functional requirements of the system
  3. Help to establish testing by identifying verification and validation opportunities
  4. Provide a path for defining classes and objects

USE CASE NAME/DESCRIPTION

Use Case: *<Use case name>*
Id:
Level: *<Low, Medium, High>*

Description
<A description of what is happening in this use case.>

Actor(s)
<Who are the actors in this use case?>

Stakeholders and Interests
<Who would be affected or interested in this use case?>

USE CASE
MAIN SCENARIO

Pre-Conditions

- *Condition 1*
- *Condition 2*

Trigger

<What happened to start this use case?>

Post-Conditions

Success end condition

Failure end condition

Minimal Guarantee

Main Scenario

1. *Step 1*
2. *Step 2*

Alternate Scenarios

NEXT WEEK

- **JABG: Read Ch. 2 and 3 (and 1 if you haven't already)**
- **Bring your laptop to class**
  - Java installed
  - Eclipse (or Netbeans) installed
  - Git client installed
  - Windows users
    - Cygwin
    - TortiseGit (optional)
- **Helpful guides are available on the Blackboard site**