

Accelerated Introduction to CS
Using Java 5

Class List

Records: 1-3

| Record | Student Information | Status | Credits |
|--------|---------------------------|----------|---------|
| 1 | Chaisuparasmikul, Pongsak | Enrolled | 4.00 |
| 2 | D'Souza, Leeroy Rinaldus | Enrolled | 4.00 |
| 3 | Kovacs, Julianna | Enrolled | 4.00 |

Schedule

- CS -201-071
M 06:25-09:05PM
Rice Campus - Wheaton Room RI 148
LAB W 06:25-09:05PM
- CS -201-392
MW 06:25-09:05PM IITV
LAB W 06:25-09:05PM

Java Must Have

- To write your first program, you need:
 - The Java™ 5 Platform, Standard Edition.**
 - Book Companion CD
 - A text editor.** Notepad, the simple editor included with the Windows platforms. To find Notepad, from the Start menu select **Programs > Accessories > Notepad.**
- You MUST use an IDE
 - Eclipse**
 - Jbuilder
 - Many available with the book Companion CD

Class Goals

- Students should be able to:
 - Analyze and explain the behavior of simple programs involving the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, functions, pointers
 - Write a program that uses each of the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, functions, pointers
- Break a problem into logical pieces that can be solved (programmed) independently

Class Goals

- Develop, and analyze, algorithms for solving simple problems.
- Use a suitable programming language, and development environment, to implement, test, and debug algorithms for solving simple problems.
- Write programs that use each of the following data structures (and describe how they are represented in memory): strings, arrays, structures, class libraries including strings and vectors
- Explain the basics of the concept of recursion.
 - Write, test, and debug simple recursive functions and procedures.

Class Goals

- Explain and apply object-oriented design and testing involving the following concepts:
 - data abstraction, encapsulation, information hiding, subclassing, inheritance, templates
- Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
- Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
 - Determine the time complexity of simple algorithms.
- Students will also learn some problem-solving strategies

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Text Book

- **TEXTBOOK (REQUIRED)**
 - Java 5 illuminated, Julie Anderson and Herve Franceschi, Jones and Bartlett ISBN 0-7637-1667-7
- **LAB MANUAL**
 - NA

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Contact Information

- Class Home Page,
<http://www.csam.iit.edu/~oaldawud/CS201/>
- Omar Aldawud
aldaoma@iit.edu
- Shulan Liu
liushul@iit.edu

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

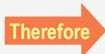
Introduction

- Chapter 1

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

What is a Computer

- Computer is a machine
 - Has parts
 - Runs on electricity
 - Breaks down sometimes
- What makes it different from other machines?
 - Can follow a list of instructions
 - Instructions process symbols



*A computer is
an information-processing
machine*

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Instructions

- Computer follows a list of instructions one instruction at a time
 - Start with first instruction
 - Do each instruction in turn
 - Stop when last instruction reached
- Similar to how you follow a list of instructions...

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Instructions

- Kinds of instructions a computer can follow (in one step)
 - "Add the numbers 16 and 23"
- Kinds of instructions a computer can't follow (in one step)
 - "Balance my checkbook"
- What's the difference?
 - Computer instructions are simple (add 2 numbers, subtract one number from another)
 - Must combine thousands of computer instructions in a **program** to do more complex tasks

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

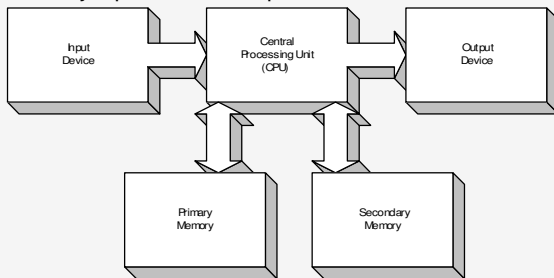
Instructions Types

- Kinds of computer instructions
 - **Arithmetic**: add, subtract, multiply, divide
 - **Comparison**: compare two numbers to see which is greater, or whether the two numbers are equal or not
 - **Branching**: jump to some other instruction in the program (list of instructions) and continue processing from there
 - **Looping**
- A computer can execute millions of these simple instructions each second

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

What is a Computer?

- Major parts of a computer



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Java 5 Illuminated

An Active Learning Approach

Julie Anderson • Hervé Franceschi

Chapter 1

- Introduction to Programming and the Java Language

Topics

- **Basic Computer Concepts**
- Data Representation
- Introduction to Programming Languages
- Introduction to Programming
 - Programming Basics
 - Program Design with Pseudocode
 - Developing a Java Application

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Basic Computer Concepts

- Hardware
 - Central Processing Unit
 - Memory and Storage Devices
- Operating Systems
- Application Software
- Computer Networks and the Internet

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Hardware

- A typical computer consists of:
 - **CPU**: executes the instructions of the program
 - **Hard disk**: stores instructions and data so program can be loaded into memory and executed
 - **Memory unit**: stores the program instructions and data while executing
 - **Keyboard** and mouse: used for data input
 - **Monitor**: used to display output from a program
 - Other accessories

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Central Processing Unit (CPU)

- **Arithmetic Logic Unit**: performs integer arithmetic and logical operations
- **Floating-point Unit**: performs floating-point operations
- **Hardware registers**: store data and memory addresses
 - A small number of **registers**. each is a high-speed storage area for temporary results
- **Instruction pointer**: keeps track of current instruction to execute
- **control unit**: determines which instruction to execute next
- Examples: Intel Pentium 4, Sun Microsystems SPARC, IBM PowerPC

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Processor Instruction Set

- Move data from one location to another
- Perform a calculation
- Change the sequence of instructions to execute (the flow of control)
- Instruction set
 - The set of machine language instructions the CPU understands
 - Examples: Intel Pentium, Motorola PowerPC, IBM AS/400
 - Some CPUs understand machine language instructions of others -- AMD and Cyrix chips understand Intel Pentium instructions

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

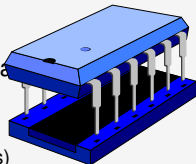
CPU Speed

- Rated in MHz or GHz
- In one clock cycle, a CPU
 - fetches an instruction from memory,
 - decodes the instruction, or
 - executes the instruction
- Pipelining allows overlap of operations to improve performance
 - In computers, a pipeline is the continuous and somewhat overlapped movement of instruction to the processor or in the arithmetic steps taken by the processor to perform an instruction. Without a pipeline, a computer processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle
- 2-Ghz CPU can execute 2 billion instructions per second

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Memory or Storage Devices

- Memory consists of cells that hold one bit.
- A **bit**'s value can be 0 or 1
- A **byte** is 8 binary digits (bits)
- Storage capacity is expressed as
 - **Kilobytes** (1,024 bytes)
 - **Megabytes** (1,048,576 bytes)
 - **Gigabytes** (1,073,741,824 bytes)



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Operating System Software

- **boots** when computer is turned on and runs continuously
- Controls the peripheral devices (disks, keyboard, mouse, etc.)
- Supports multitasking (multiple programs executing simultaneously)
- Allocates memory to each program
- Prevents one program from damaging another program
- Example: Microsoft Windows, Linux

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Application Software

- Written to perform specific tasks
- Runs "on top of" operating system
- Examples: word processor, spreadsheet, database management system, games, Internet browser, etc.

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Computer Networks

- Networks connect two or more computers so they can share files or devices
- **Local Area Network (LAN)** : computers located geographically close to one another
- **Servers** provide services, such as:
 - access to database, downloading of files, e-mail delivery
- **Clients** use these services. Most desktop computers are clients.

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

The Internet

- Evolved from ARPANET, a military research project
- **Web servers** deliver Internet content (**Web pages**) to clients via a browser.
- Web pages identified using a **URL (Uniform Resource Locator)**
- **Domain Name Servers (DNS)** translate URL to **Internet Protocol (IP)** address, which identifies specific computer (servers) on the Internet

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Data Representation

- Binary Numbers
 - Expressed in base 2 system (two values are 0 and 1)
- Hexadecimal Numbers
 - Base-16 system used as shorthand for binary numbers
- ASCII Character Set
 - Uses 7 bits to encode each character, we can represent 128 characters.
- Unicode Character Set
 - Each Unicode character is represented as 16 bits, we can generate 65,536 characters

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Binary Equivalents of Decimal Numbers

| <u>Decimal</u> | <u>Binary Equivalent</u> |
|----------------|--------------------------|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0100 |
| 7 | 0111 |
| 8 | 1000 |

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Powers of 2

| | <u>Decimal</u> | | <u>Decimal</u> |
|-------|----------------|----------|----------------|
| 2^0 | 1 | 2^8 | 256 |
| 2^1 | 2 | 2^9 | 512 |
| 2^2 | 4 | 2^{10} | 1024 |
| 2^3 | 8 | 2^{11} | 2048 |
| 2^4 | 16 | 2^{12} | 4096 |
| 2^5 | 32 | 2^{13} | 8192 |
| 2^6 | 64 | 2^{14} | 16384 |
| 2^7 | 128 | 2^{15} | 32768 |

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Decimal (base 10) numbers

- A decimal number can be represented as the sum of powers of 10 (the base) with coefficients in the base 10 alphabet (0 - 9)

For example:

$$2485 = 2000 + 400 + 80 + 5$$

$$2485 = 2 * 1000 + 4 * 100 + 8 * 10 + 5 * 1$$

$$2485 = 2 * 10^3 + 4 * 10^2 + 8 * 10^1 + 5 * 10^0$$

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Converting From Decimal to Binary

- Just as a decimal number can be represented as a sum of powers of 10 (the base) with coefficients in the base 10 alphabet (0 to 9),
- A decimal number also can be represented as the sum of powers of 2 (the base of the binary system) with coefficients in the base 2 alphabet (0 and 1)
- So we need an algorithm to do that

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Converting From Decimal to Binary

1. Find the largest power of 2 that is smaller than or equal to the decimal number
2. Subtract that number from the decimal number
3. Insert 1 in binary number for position equivalent to that power of 2
4. Repeat 1 - 3, until you reach 0

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Example: convert 359 to binary

1. Largest power of 2 that is smaller than 359 is 256 (2^8)
2. $359 - 256 = 103$
so $359 = 2^8 * 1 + 103$
1. Largest power of 2 that is smaller than 103 is 64 (2^6)
2. $103 - 64 = 39$
so $359 = 2^8 * 1 + 2^6 * 1 + 39$
(continued on next slide)

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Example: convert 359 to binary

1. Largest power of 2 that is smaller than 39 is 32 (2^5)
2. $39 - 32 = 7$
so $359 = 2^8 * 1 + 2^6 * 1 + 2^5 * 1 + 7$
1. Largest power of 2 that is smaller than 7 is 4 (2^2)
2. $7 - 4 = 3$
so $359 = 2^8 * 1 + 2^6 * 1 + 2^5 * 1 + 2^2 * 1 + 3$
(continued on next slide)

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Example: convert 359 to binary

1. Largest power of 2 that is smaller than 3 is 2 (2^1)
2. $3 - 2 = 1$
so $359 = 2^8 * 1 + 2^6 * 1 + 2^5 * 1 + 2^2 * 1 + 2^1 * 1 + 1$
1. Largest power of 2 that is smaller than or equal to 1 is 1 (2^0)
2. $1 - 1 = 0$, so we are finished

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Our results

Finally, insert missing powers of 2 with coefficient 0.
Thus, 359 =

$$2^8 \cdot 1 + 2^7 \cdot 0 + 2^6 \cdot 1 + 2^5 \cdot 1 + 2^4 \cdot 0 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 1$$

Removing powers of 2, we get:

1 0 1 1 0 0 1 1 1

Or

1 0110 0111

CS201http://www.csam.iit.edu/~oaldawud/CS201

Method 2

- Divide 359 by 2 → 179 and the remainder is 1
- Divide 179 by 2 → 89 and the remainder is 1
- Divide 89 by 2 → 44 and the remainder is 1
- Divide 44 by 2 → 22 and the remainder is 0
- Divide 22 by 2 → 11 and the remainder is 0
- Divide 11 by 2 → 5 and the remainder is 1
- Divide 5 by 2 → 2 and the remainder is 1
- Divide 2 by 2 → 1 and the remainder is 0
- Divide 1 by 2 → 0 and the remainder is 1

Answer: 101100111



CS201http://www.csam.iit.edu/~oaldawud/CS201

Hexadecimal Numbers

- Base-16 number system
- Uses digits 0 - 9 and letters A - F
- One hexadecimal digit can express values from 0 to 15
 - For example: C represents 12
- Thus, one hexadecimal digit can represent 4 bits

CS201http://www.csam.iit.edu/~oaldawud/CS201

Hexadecimal - Binary Equivalents

| Hex | Binary | Hex | Binary |
|-----|--------|-----|--------|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | A | 1010 |
| 3 | 0011 | B | 1011 |
| 4 | 0100 | C | 1100 |
| 5 | 0101 | D | 1101 |
| 6 | 0110 | E | 1110 |
| 7 | 0111 | F | 1111 |

CS201http://www.csam.iit.edu/~oaldawud/CS201

Examples

Binary number: 0001 1010 1111 1001

Hex equivalent: 1 A F 9

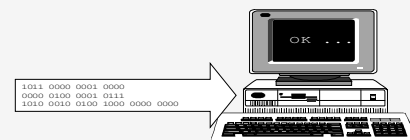
Binary number: 1011 0011 1011 1110

Hex equivalent: B 3 B E

CS201http://www.csam.iit.edu/~oaldawud/CS201

Programming a Computer

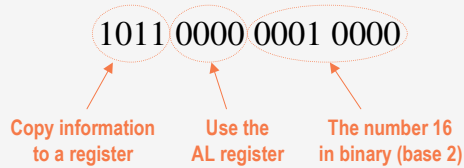
- Computers understand machine language directly
 - Example: add 16 and 23 in Intel 8086 machine language
 - 1011 0000 0001 0000
 - 0000 0100 0001 0111
 - 1010 0010 0100 1000 0000 0000



CS201http://www.csam.iit.edu/~oaldawud/CS201

Programming a Computer

- Parts of machine language program step
 - First statement from example



– Instruction says: Copy the number 16 into the AL register

- Difficult to read and write machine language!

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

The Unicode Character Set

- Each character stored as 16-bits
- Maximum number of characters that can be represented: 65,536 (2^{16})
- ASCII character set (used by many programming languages) stores each character as 7 bits (maximum number of characters is 128).
- For compatibility, first 128 characters of Unicode set represent the ASCII characters

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Some Unicode Characters

| <u>Unicode Character</u> | <u>Decimal Value</u> |
|--------------------------|----------------------|
| * | 42 |
| 1 | 49 |
| 2 | 50 |
| A | 65 |
| B | 66 |
| a | 97 |
| b | 98 |
| } | 125 |

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Programming Languages

- Machine language
- Assembly language
- High-level languages

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Machine & Assembly Languages

- Machine language
 - Written using CPU instruction set
 - Difficult to write, and not portable
- Assembly language
 - Written using mnemonics for instructions and symbolic names for variables
 - Assembler converts code to machine language
 - Easier to write, but still not portable

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Machine Language

- **Is not portable**
- **Runs only on specific type of computer**
- **Is made up of binary-coded instructions (strings of 0s and 1s)**
- **Is the language that can be directly used by the computer**

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Low Level Languages

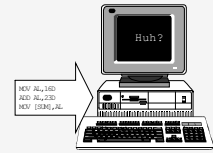
- Assembly language
 - One step up from machine language
 - Each assembly language instruction corresponds to one machine language instruction

```
MOV AL, 16D    → 1011 0000 0001 0000
ADD AL, 23D    → 0000 0100 0001 0111
MOV [SUM], AL  → 1010 0010 0100 1000 0000 0000
```

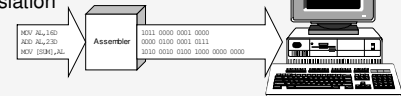
CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Low Level Languages

- Assembly language
 - Computer can't understand assembly language directly



- Must translate from assembly to machine language
- Assembler: program that does this translation



CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Assembly Languages

- Are **machine dependent** and run on only one specific type of computer
- Are translated into machine code by **assemblers**
- Are made up of English-like abbreviations such as LOAD, STORE, or ADD

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

High-Level Languages

- Examples: Fortran, Perl, COBOL, C++, Java
- Highly symbolic
- Portable among CPU architectures
- Languages can be designed for specific uses:
 - Perl: Internet applications
 - Fortran: scientific applications
 - COBOL: business applications

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

High-level languages

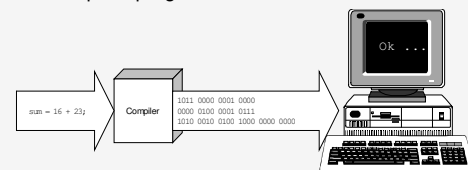
- Each high-level language instruction may correspond to several machine language instructions
- Easier for people to read and write than assembly
- C++ example:

```
sum = 16 + 23; → 1011 0000 0001 0000
                → 0000 0100 0001 0111
                → 1010 0010 0100 1000 0000 0000
```

CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

Programming a Computer

- High-level languages
 - Must translate high-level language program to machine language before computer can execute it
 - Compiler: program that does this translation



CS201 <http://www.csam.iit.edu/~oaldawud/CS201>

High-Level Languages

- **Compiled**
 - Compiler converts source code (instructions and data) into machine language, then program is executed
- **Interpreted**
 - Interpreter converts instructions into machine language at run time as instructions are executed
 - Usually executes more slowly than compiled program

CS201http://www.csam.iit.edu/~oaldawud/CS201

High-Level Languages

- **Are portable**
- **Are translated into machine code by compilers**
- **Instructions are written in language similar to natural language**
- **Examples -- FORTRAN, COBOL, Pascal, C, C++**
- **Many are standardized by ISO/ANSI to provide an official description of the language**

CS201http://www.csam.iit.edu/~oaldawud/CS201

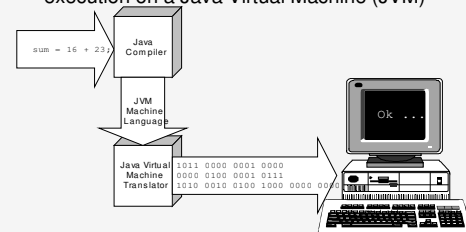
Java

- Combination of **compiler** and **interpreter**
- Compiler converts source code into **byte codes** (an instruction set for a **virtual**, machine-independent processor)
- At run time, the **Java Virtual Machine (JVM)** interprets the byte codes and converts them into the machine language on which the program is running.

CS201http://www.csam.iit.edu/~oaldawud/CS201

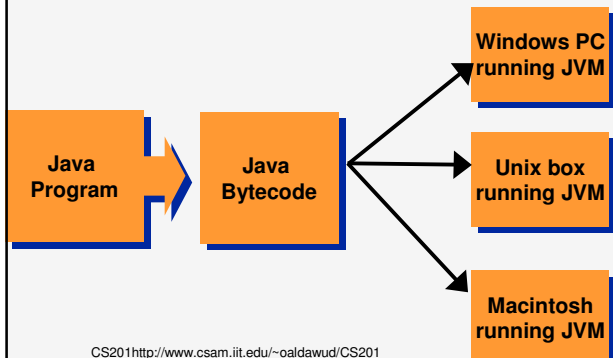
Programming a Computer: Java

- **The Java Programming Language**
 - A high-level language
 - Java compiler translates to Java bytecodes for execution on a Java Virtual Machine (JVM)



CS201http://www.csam.iit.edu/~oaldawud/CS201

Java Portability



CS201http://www.csam.iit.edu/~oaldawud/CS201

Java

- **Achieves portability by using both a compiler and an interpreter**
- **Java compiler translates a Java program into an intermediate Bytecode--not machine language**
- **An interpreter program called the Java Virtual Machine (JVM) translates each successive instruction in the Bytecode program to machine language and immediately runs it**

CS201http://www.csam.iit.edu/~oaldawud/CS201

The Java Language

- Created by Sun Microsystems in 1995
- Syntax based on C++
- Object-oriented
- Support for Internet applications
- Extensive library of prewritten classes
- Portability among platforms
- Built-in networking

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Java Programs

- Applets
 - Small programs designed to add interactivity to Web sites
 - Downloaded with the Web page and launched by the Internet browser
- Servlets
 - Run by Web server on the server
 - Typically generate Web content
- Applications
 - Programs that run standalone on a client

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

An Introduction to Programming

- Programming Basics
- Program Design with Pseudocode
- Developing a Java Application

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Programming Basics

- Programming is translating a problem into ordered steps consisting of operations a computer can perform:
 - Input
 - Calculations
 - Comparisons of values
 - Moving data
 - Output
- The order of execution of instructions is called **flow of control**

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Program Design with Pseudocode

- Pronounced *sue-dough-code*
- English-like language for specifying the design of a program
- Programmer can concentrate on design of program without worrying about Java language rules (**syntax**)
- Then convert pseudocode into Java code

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Four Types of Flow of Control

- Sequential Processing
 - Execute instructions in order
- Method Call
 - Jump to code in method, then return
- Selection
 - Choose code to execute based on data value
- Looping or Iteration
 - Repeat operations for multiple data values

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Sequential Processing

- The pseudocode for calculating the sum of two numbers would look like this:

```
read first number
read second number
set total to (first number +
              second number)
output total
```



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

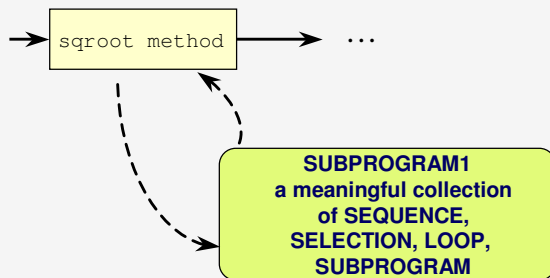
Method Call

- Calling the method** executes the method
- Methods can take **arguments** (data to use) and return values
- Here is pseudocode for calculating the square root of an integer:

```
read an integer
call the square root method,
  with integer as argument
output the square root
```

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Methods



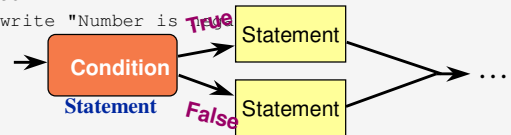
CS201<http://www.csam.iit.edu/~oaldawud/CS201>

IF Condition THEN Statement1 ELSE Statement2

Selection

- The pseudocode for determining if a number is positive or negative is:

```
read a number
if the number is greater than
  or equal to 0
  write "Number is positive."
else
  write "Number is negative."
```



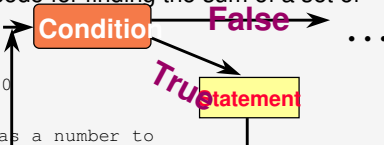
CS201<http://www.csam.iit.edu/~oaldawud/CS201>

WHILE Condition DO Statement1

Looping

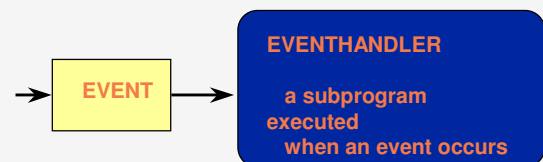
- The pseudocode for finding the sum of a set of numbers is:

```
set total to 0
read a number
while there was a number to
  read,
  add number to total
  read the next number
write total
```



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

ASYNCHRONOUS CONTROL



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Object-oriented Programming (OOP)

- **Class**
 - tool for encapsulating data and operations (**methods**) into one package
 - defines a template or model for creating and manipulating **objects**
- **Objects**
 - data created using the class and its methods
 - an object is an *instance* of the class
 - creating an object is **instantiation**

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

- **Date class**
 - **data: month, day, year**
 - **operations to set and return month, day, year**
- **a Date object**
 - **June**
 - **23**
 - **2004**

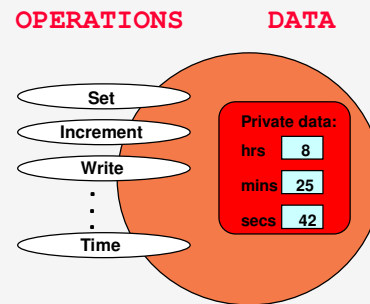
CS201<http://www.csam.iit.edu/~oaldawud/CS201>

OOP Advantage: Reuse

- Well-written classes can be reused in new applications
- Shortens development time because programmers don't need to write new code
- Programs are more robust because the class code is already tested

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

An Object of class Time



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Developing a Java Application

1. Write the source code
 - Using an Integrated Development Environment (IDE) or text editor
 - Save in a **.java** file
2. Compile the source code:
 - **`javac ClassName.java`**
 - Creates **.class** file
3. Execute the application:
 - **`java ClassName`**
 - Run by the Java Virtual Machine (JVM)

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

A First Application

```
1 // First program in Java
2 // FirstProgram.java
3
4 public class FirstProgram
5 {
6     public static void main( String [] args )
7     {
8         System.out.println( "Programming is not "
9                             + " a spectator sport!" );
10        System.exit( 0 );
11    }
12 }
```

CS201<http://www.csam.iit.edu/~oaldawud/CS201>



- Java is case-sensitive.
- The class name and the source filename must match exactly, including capitalization.

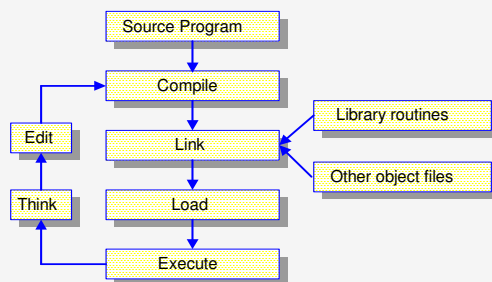
CS201http://www.csam.iit.edu/~oaldawud/CS201

Program Errors

- Compiler errors
 - Found by the compiler.
 - Usually caused by incorrect syntax or spelling
- Run-time errors
 - Reported by the JVM
 - Usually caused by incorrect use of prewritten classes or invalid data
- Logic errors
 - Found by testing the program
 - Incorrect program design or incorrect execution of the design

CS201http://www.csam.iit.edu/~oaldawud/CS201

Program Life Cycle



CS201http://www.csam.iit.edu/~oaldawud/CS201

Introduction: Java Platform

- Java™ platform is based on the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices.
- Portability
 - Java platform allows you to run the same Java application on lots of different kinds of computers.

CS201http://www.csam.iit.edu/~oaldawud/CS201

JVM

- Java virtual machine or "JVM™"
 - a **translator** that turns general Java platform instructions into **tailored** commands that make the devices do their work

CS201http://www.csam.iit.edu/~oaldawud/CS201

Java Program Phases

1. Edit
 - IDE, vi, emacs, notepad
2. Compile
 - **javac**
3. Load
 - Class Loader
4. Verify
 - Verifier (Security and Validation)
5. Execute
 - **Java** Interpreter (java or appletviewer)

CS201http://www.csam.iit.edu/~oaldawud/CS201

The java Compiler

- javac
 - Creates bytecodes
 - Stored on disk as **.class**
 - Each file contains bytecode for one and only one class

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

The java Class Loader

- Loads bytecodes in memory
 - Loads **.class** files from disk into memory

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

The Java Bytecode Verifier

- Confirms that all bytecodes are:
 - **Valid** and
 - Don't violate Java's **Security** restrictions
- Works on bytecodes from memory

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

The java Interpreter

- Reads bytecodes and translate them into a language that is relevant to the targeted computer architecture
 - PC
 - Machintosh
 - Cell Phone
- For applets execution its built into the browser or the appletviewer

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Your first program: HelloWorldApp.java

- **Displays the greeting "Hello world!". To create this program, you will:**
- **Create a source file**
 - A source file contains text, written in Java.
- **Compile the source file into a bytecode file.**
 - The *compiler*, **javac**, converts these instructions into a bytecode file

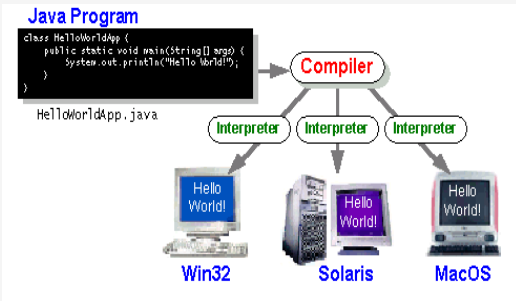
CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Your first program: HelloWorldApp

- **Run the program contained in the bytecode file**
 - The Java interpreter implements the Java VM
 - This interpreter takes your bytecode file and carries out the instructions by translating them into instructions that your computer can understand.

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

JRE



CS201<http://www.csam.iit.edu/~oaldawud/CS201>

JAVA Program

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

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Steps

1. Save the file as HelloWorldApp.java
2. Compile
 - Javac HelloWorldApp.java → HelloWorldApp.class
3. Run the program
 - Java HelloWorldApp

CS201<http://www.csam.iit.edu/~oaldawud/CS201>

Output

```
MS-DOS Prompt
8 x 12
C:\java>dir
Volume in drive C is DB02
Volume Serial Number is F3C4-E800
Directory of C:\java

.<DIR>          07-22-99 11:23p .
..<DIR>        07-22-99 11:23p ..
HELLOW~1 JAV  272 07-23-99 12:39a HelloWorldApp.java
HELLOW~1 CLA  478 07-23-99 12:40a HelloWorldApp.class
                2 file(s)          758 bytes
                2 dir(s)      218,734,592 bytes free

C:\java>
```

CS201<http://www.csam.iit.edu/~oaldawud/CS201>