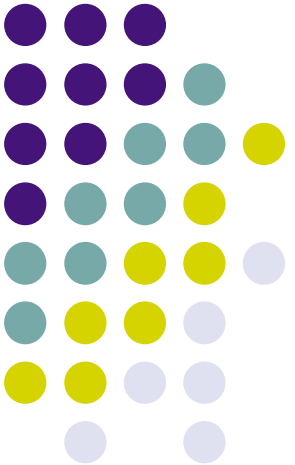
Principles of Computer Science I

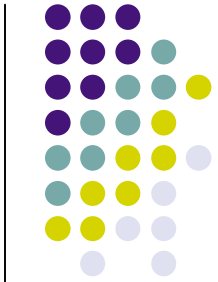
Prof. Nadeem Abdul Hamid

CSC 120 – Fall 2006

Lecture Unit 1 - Introduction



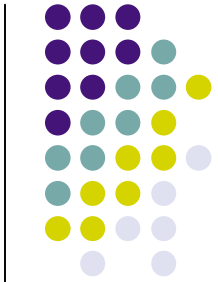
Lecture Outline



- What is a computer?
- What is computer science?
- Course mechanics
- Hardware & programming overview
- Compiling & running Java programs
- Binary numbers

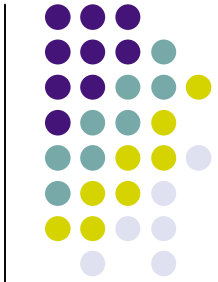
Portions based on *Python Programming: An Introduction to Computer Science* by John Zelle

What is a ‘computer’?



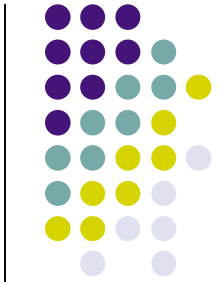
- What have you used a ‘computer’ for?
  - Writing a paper… balancing checkbook… playing games…
- Computers also used to…
  - Predict the weather… design airplanes… make movies… run businesses… perform financial transactions… control factories…
- How can one device perform so many tasks?
- What *exactly* is a ‘computer’?

Modern Computers



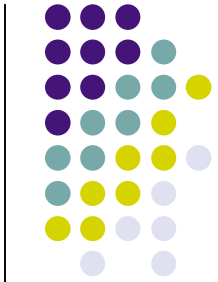
- “A machine (?) that stores and manipulates information under the control of a changeable program.”
- Two key elements to definition:
  - Device for manipulating information
    - Calculators, gas pumps also manipulate info… but these are built to only perform single, specific tasks
  - Operate under control of a changeable program
    - Can provide step-by-step instructions to a computer telling it what to do
    - By changing the *computer program*, can get the computer to perform different tasks

A Universal Machine



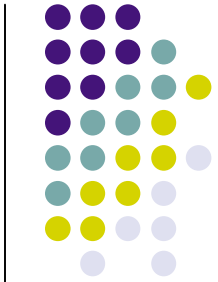
- Every computer is a machine for *executing* (carrying out) programs
- Many different types of computers
  - Macintoshes, PCs…
  - Thousands of other kinds of computers, real and theoretical
- Remarkable discovery of computer science:
  - All different types of computers have same power
  - With suitable programming, each computer can basically do all the things any other can

Programming



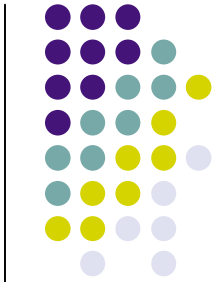
- *Software* (programs) control *hardware* (the physical machine)
- Building software = programming
  - Challenging
    - See the big picture while paying attention to small details, but…
    - Anyone can learn to program
  - Become a more intelligent user of computers
  - Fun
  - Career

Computer Science (CS)



- Is NOT “the study of computers”
  - Dijkstra: computers are to CS as telescopes are to astronomy
  - Aeronautics engineer vs. airplane pilot
- The study of *computation*
- Fundamental question of CS: *What can be computed?*
  - Computers can carry out any process we describe
  - So, what processes can be described in order to solve problems?
- *Algorithm*: Step-by-step process to solve a problem

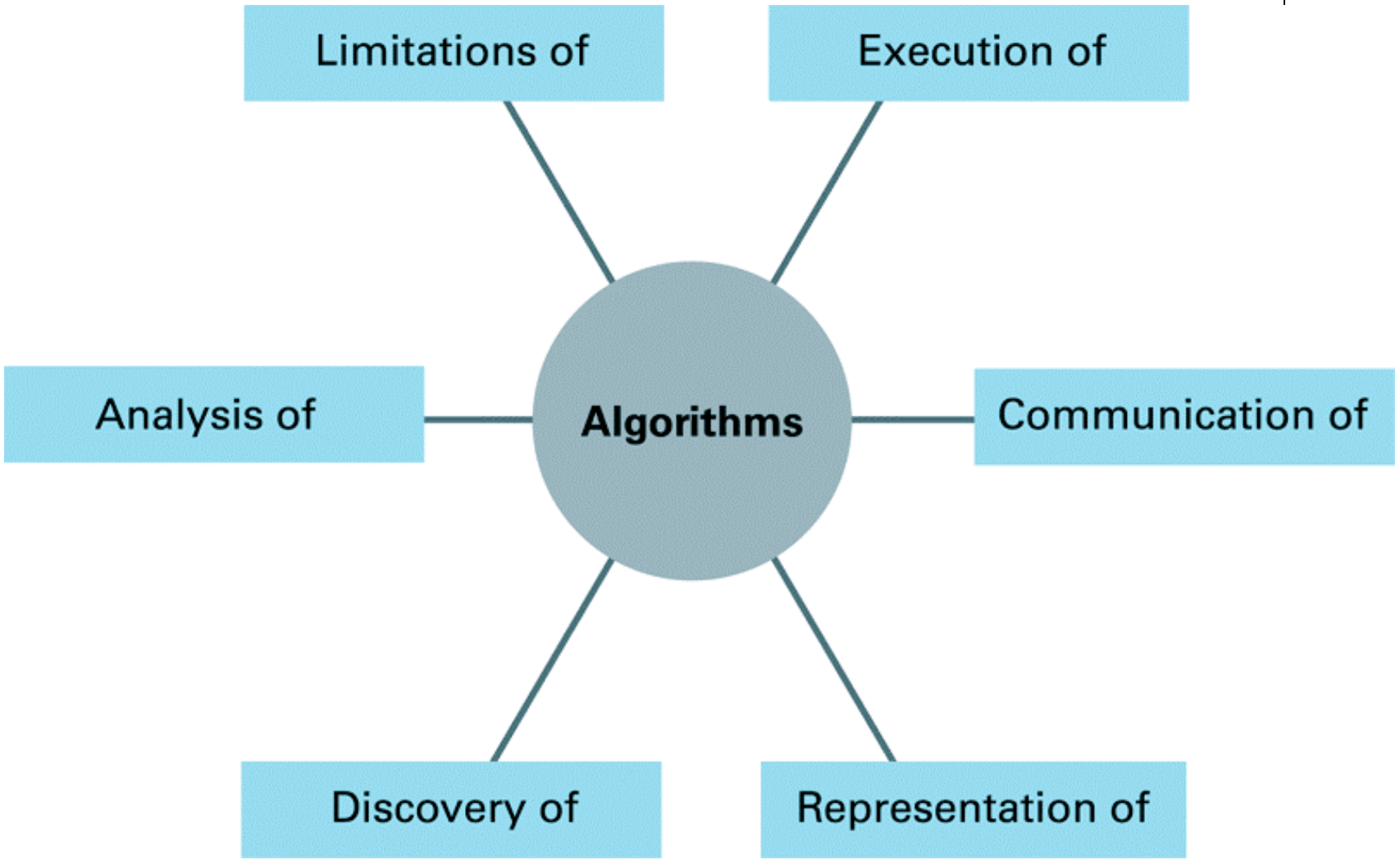
Algorithms in CS



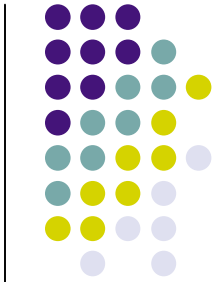
- Step-by-step process that solves a problem
  - More precise than a recipe
  - Eventually has to stop with an answer
  - General description of a process rather than specific to a computer or programming language

- Areas of CS discipline span
  - Theory (mathematics)
  - Experimentation (science)
  - Design (engineering)

Role of Algorithms in CS

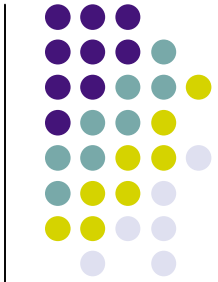


Subareas of CS



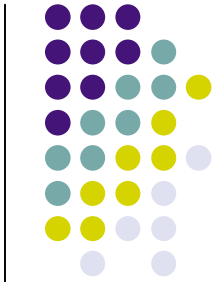
- Architecture — hardware-software interface
- Artificial Intelligence — thinking machines
- Computational Geometry — theory of animation, 3-D models
- Graphics — from Windows to Hollywood
- Operating Systems — run the machine
- Scientific Computing — weather, hearts
- Software Engineering — peopleware
- Theoretical CS — analyze algorithms, models
- Many other subdisciplines… *networking, numerical and symbolic computation, bioinformatics, databases and information retrieval, web and multimedia design, security, human-computer interaction…*

Course Mechanics



- Syllabus, lectures notes, assignments, etc. on web page
  - *http://cs.berry.edu/csc120*
- Class meetings
  - Lectures: Mon/Wed/Fri, 11-11:50AM, SCI 233
  - Labs: Thurs, 12:30–2:30PM, SCI 228
- Contact
  - Office: SCI 354B — Phone: 368-5632
  - Email: *nadeem@acm.org*
- Office Hours
  - Mon: 10-11am, 2:30-4pm
  - Tue: 9-11am
  - Wed: 10-11am, 2:30-4pm
  - Thu: 9-11am
  - Fri: 10-11am
  - (or by appointment…)

Assignments

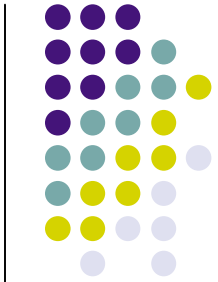


- Weekly lab/homeworks
  - Due on Wednesdays (usually)
- Programming Projects

- DON'T WAIT UNTIL DAY/NIGHT BEFORE TO START WORKING ON ASSIGNMENTS
  - No late work accepted, without formal excuse/prior arrangement
  - You will NOT be able to complete the programming assignments in one night

- Send email if you have a problem (attached relevant files and say where you're stuck)

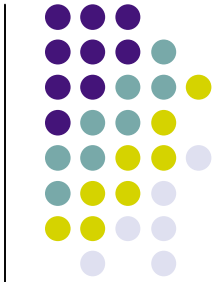
Programming Assignments



- Completed programs must ‘work’!!!
  - Compile and run (will learn what that means later)

- If you leave programming assignments to the last minute, you will run a major risk of having incomplete work

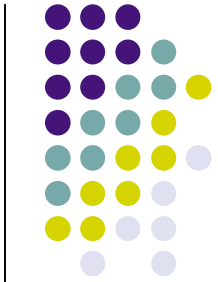
Materials and Resources



- Textbook:
  - *Java Concepts, 4th Edition*, Cay Horstmann
- Online course website: Check regularly

- Software (in computer lab SCI 228)
  - Java 5.0 (JDK): *http://java.sun.com/j2se/1.5.0/download.jsp*
    - Compiler; runtime system
  - DrJava: *http://www.drjava.org*
    - Editor; development environment

Assessment and Grading



- Class/Lab participation and attendance
- Chapter Quizzes
- Programming Assignments
- Exams -- *Tentative dates:*
  - Exam 1: Monday, September 25, 2006
  - Exam 2: Wednesday, November 1, 2006
  - Final exam: Wednesday, December 13, 2006 (8 - 10 am)
- Policies (see syllabus)
  - Attendance
  - Academic integrity (*Pair programming)
  - Late work
  - Disabilities

Hardware Basics

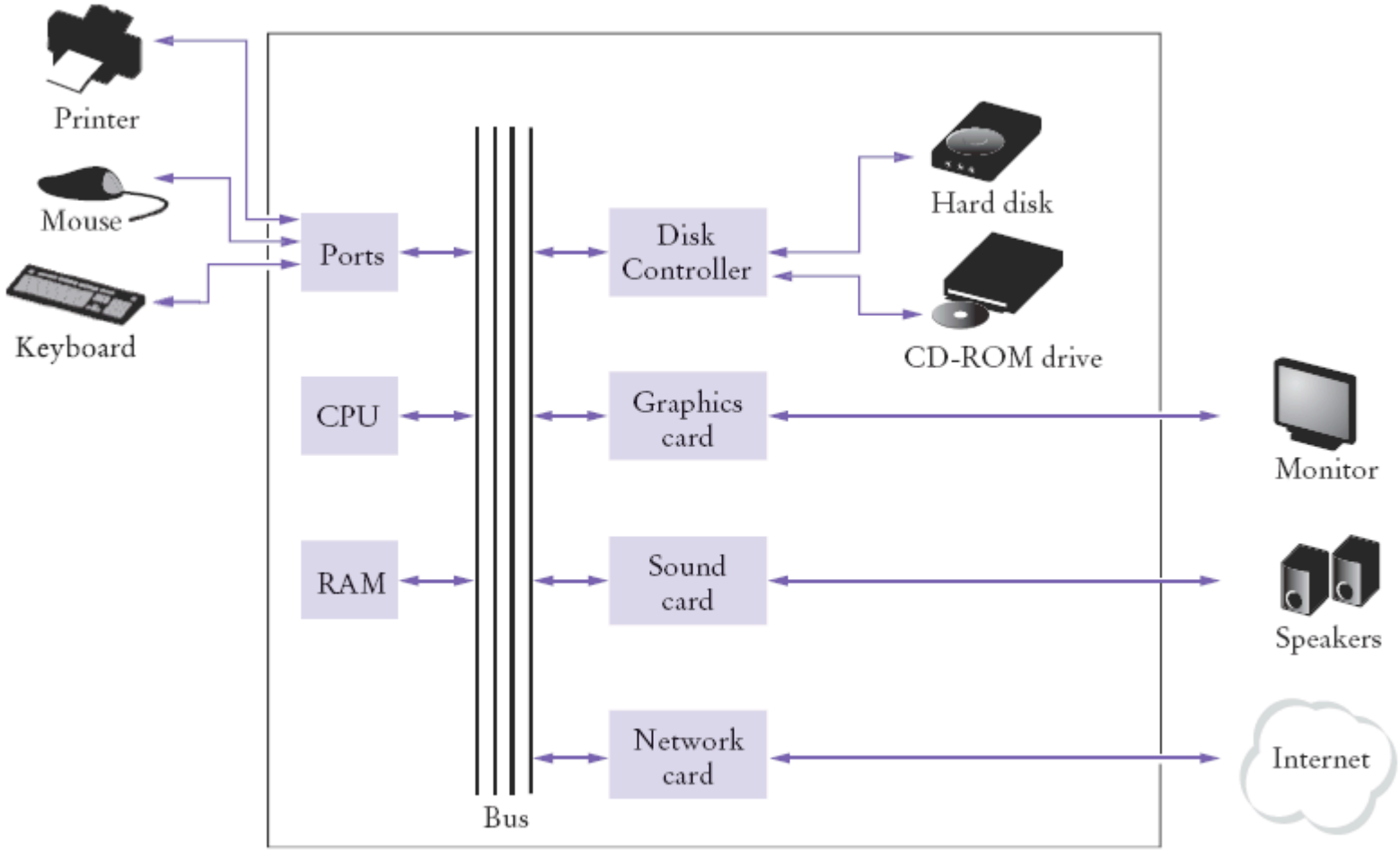
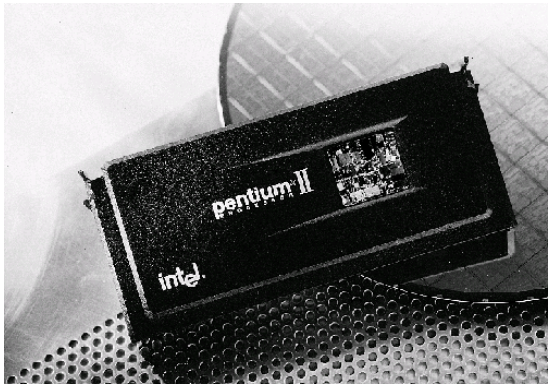


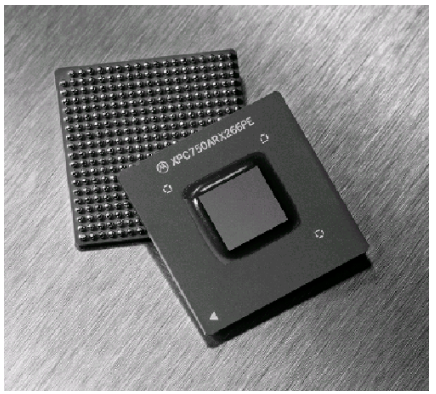
Figure 5 Schematic Diagram of a Computer

Central Processing Unit (CPU)

- Heart and brain of the machine
- Chip composed of *transistors*, wiring
- Two primary components
  - Arithmetic/Logic Unit (ALU) performs arithmetic and logical operations
  - Control Unit controls the order in which instructions in the program are executed

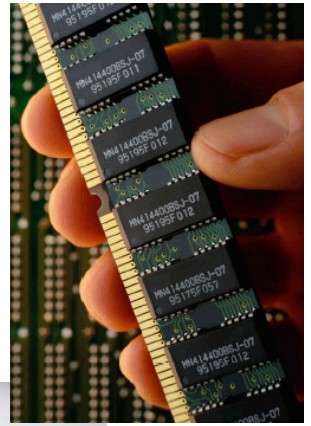
Pentium (left) and *PowerPC G3* chips



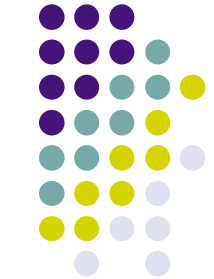


Memory

- Stores programs and data
- CPU can only directly access info. in *main memory* or *primary storage* (RAM- random access memory)
  - Relatively expensive
  - Volatile (loses data when no power)
- Secondary storage- more permanent
  - Hard disk
  - Floppy, CD, DVD, tape, …



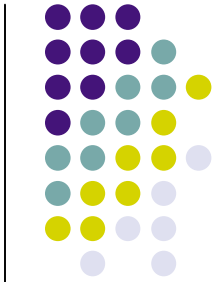
RAM - Main Memory



- Ordered sequence of storage cells
- Each holds one piece (a ‘word’) of data
- ‘Data’ is a sequence of bits (on/off - 0/1)
- 8 bits = 1 byte

- Each memory cell has a unique address (integer number)

Peripheral Devices

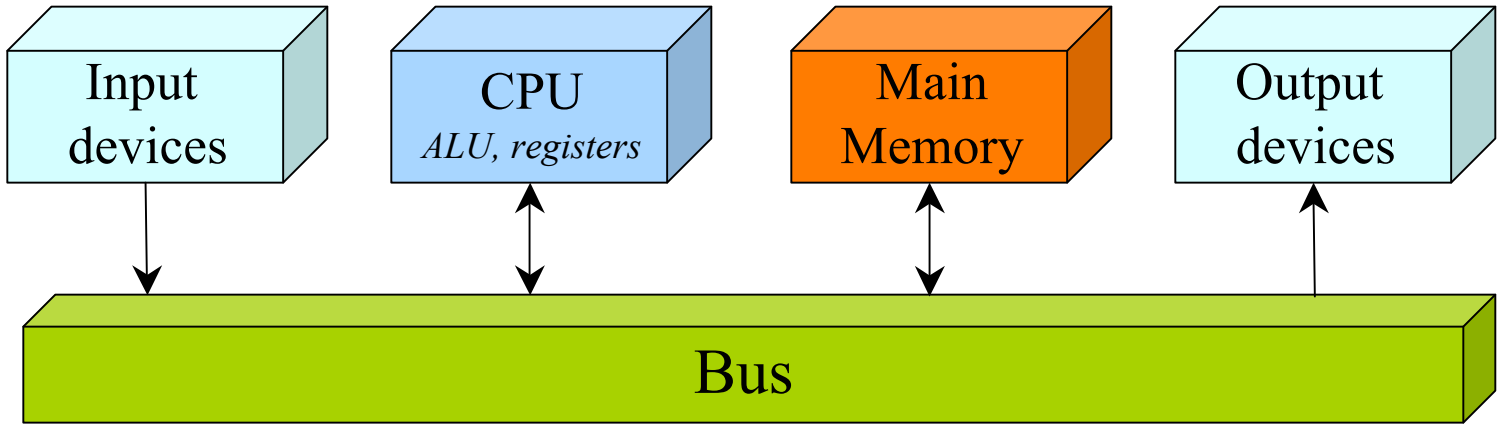


- Input Devices
  - keyboard
  - mouse
- Output Devices
  - printer
  - video display
  - LCD screen
- Auxiliary Storage
  - disk drive
  - CD-ROM drive
  - DVD-ROM drive

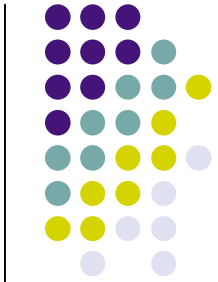
Fetch-Execute Cycle

- How the CPU operates

- Fetch the next instruction
- Decode the instruction into control signals
- Get data if needed (from memory)
- Execute the instruction



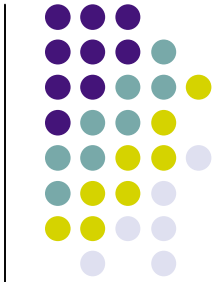
Machine Code



- Instructions/operations that CPU ‘understands’
- Different vendors (Intel, Sun, IBM) use different sets of machine instructions

- Extremely primitive
- Encoded as (binary) numbers

Programming Languages



- Could we use English to give instructions to a computer?
  - "I saw the man in the park with the telescope."
    - Who had the telescope? Who was in the park?
  - 'Natural languages' are full of ambiguity and imprecision
    - Made up for by lots of redundancy and shared human knowledge
- Computer scientists design precise notations for expressing instructions/statements: *programming languages*
- Programming languages have structures with
  - Precise form (*syntax*)
  - Precise meaning (*semantics*)

High-level Languages

- Designed to be used and understood by humans
  - C, C++, Java, Perl, Scheme, BASIC, …
- All have well-defined, unambiguous syntax and semantics

- Problem:
  - Humans write *code* (programs) in high-level languages
  - Computers only ‘understand’ machine language (0’s, 1’s)

- *Compiler*: Program that translates programs from high-level language to machine code

Java Programming Language

- Benefits
  - Simple (relatively)
  - Safe (security features prevent many ‘bad’ things)
  - Platform-independent (‘write once, run anywhere’)
  - Rich library (packages)
    - Lots of code already written for you to do lots of stuff
  - Designed for Internet (applets)

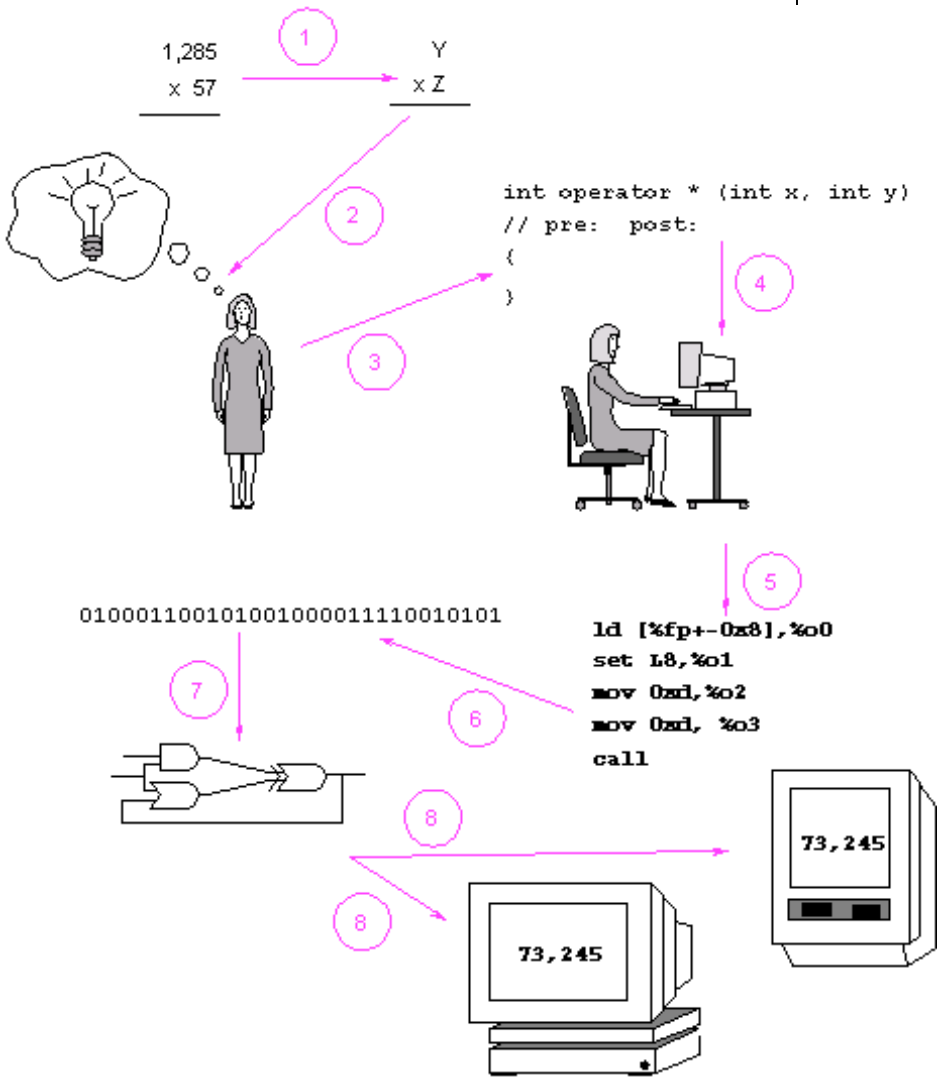
Caveats

- Programs we write in this course will not be fancy
  - Today's sophisticated programs/games built by teams of highly skilled programmers, artists, other professionals
- Java language
  - Was designed for professionals, not students
  - Evolving - features change with different versions (we'll be using 5.0)
  - Cannot learn all Java in one semester
    - In fact, (???) no one can hope to learn entire Java library in a lifetime…
- Goal of this course: Learn how to think about problem solving and expressing precise solutions using a programming language

Writing a Program

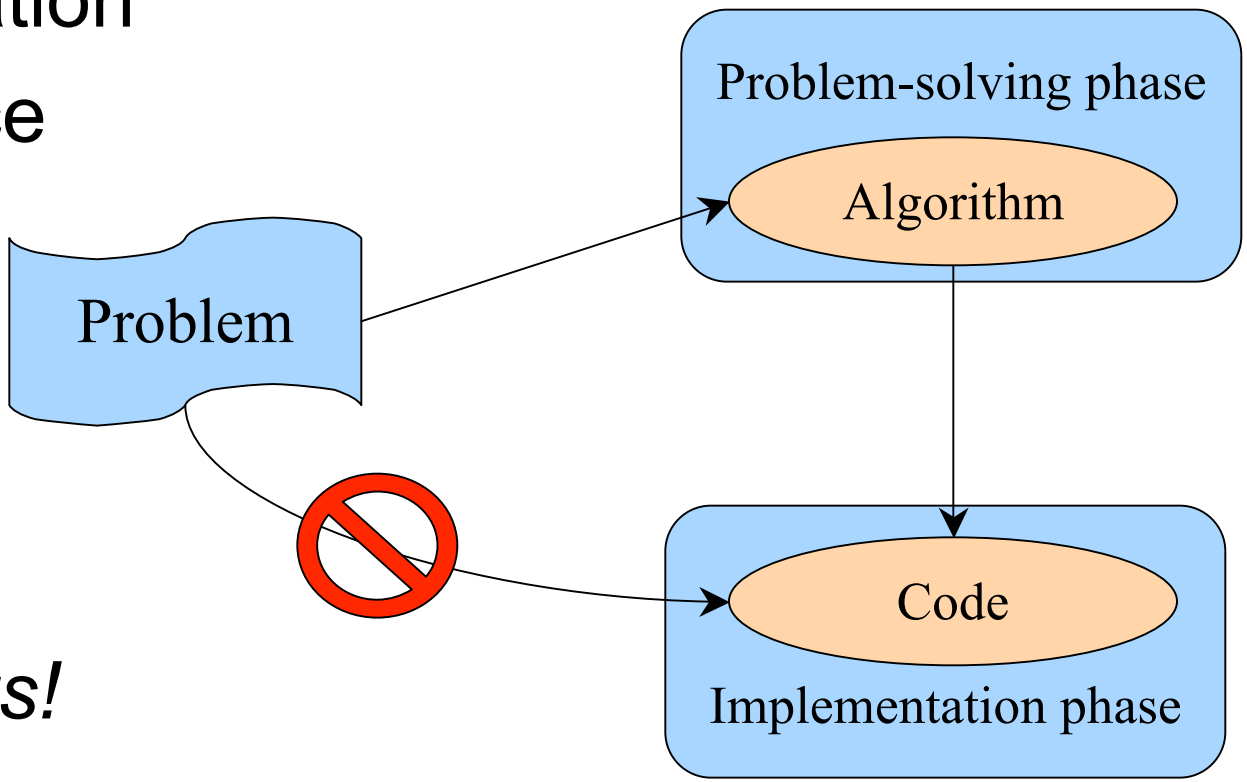
- Specify the problem
  - remove ambiguities
  - identify constraints
- Develop algorithms, design classes, design software architecture
- Implement program
  - revisit design
  - test, code, debug
  - revisit design
- Documentation, testing, maintenance of program

- From ideas to electrons



Software Life Cycle

- Problem-solving and design
- Implementation
- Maintenance



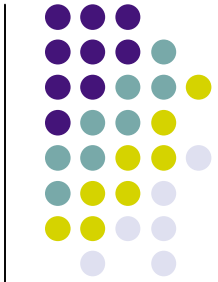
- *No shortcuts!*

Algorithms and Programs

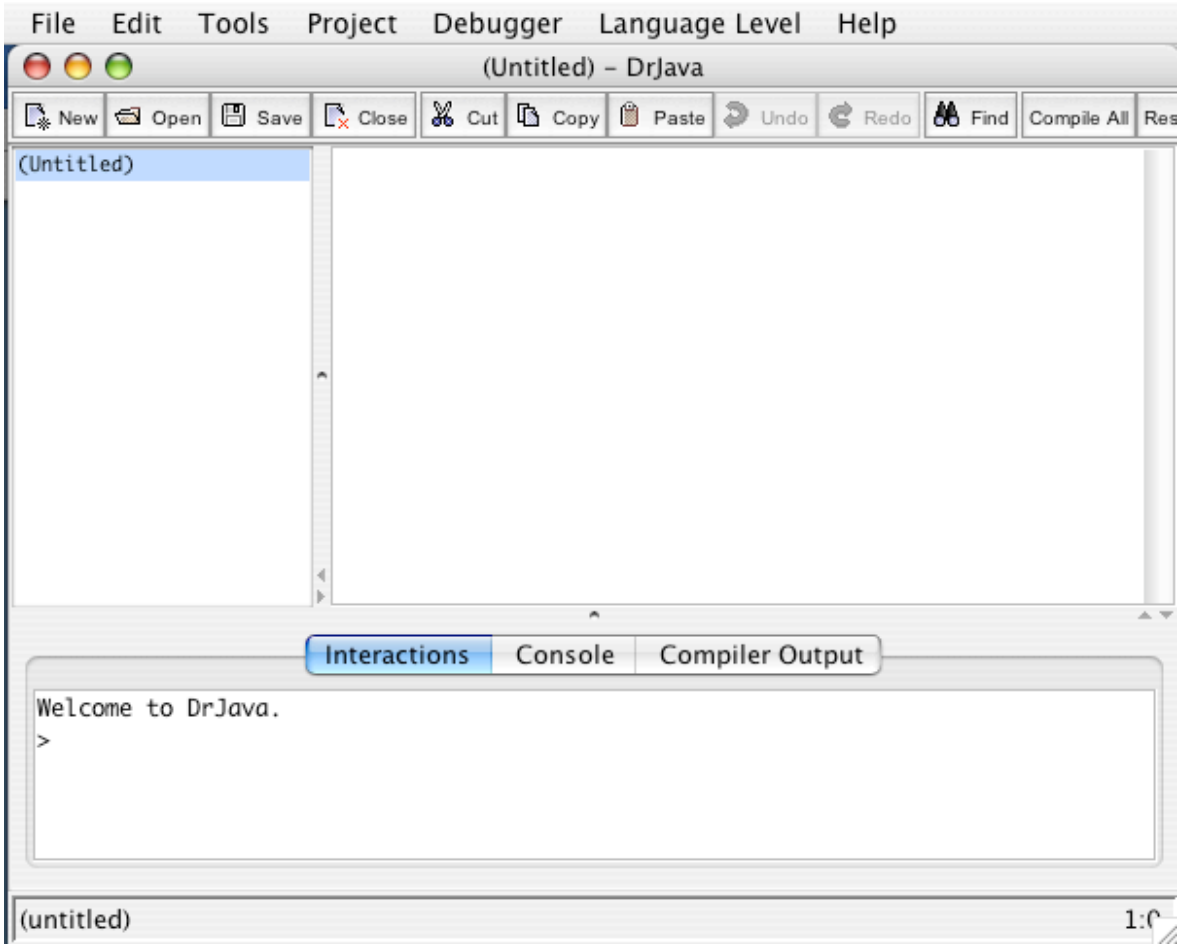
- Algorithm
  - instructions for solving a problem in a finite amount of time using a finite amount of data
  - expressed in a precise, but general, way (using English?), independent of type of computer

- Program
  - an algorithm written for a particular computer using a particular language

Writing and Compiling a Java Program Using DrJava



IDE = Integrated Development Environment



Java Compilation Process

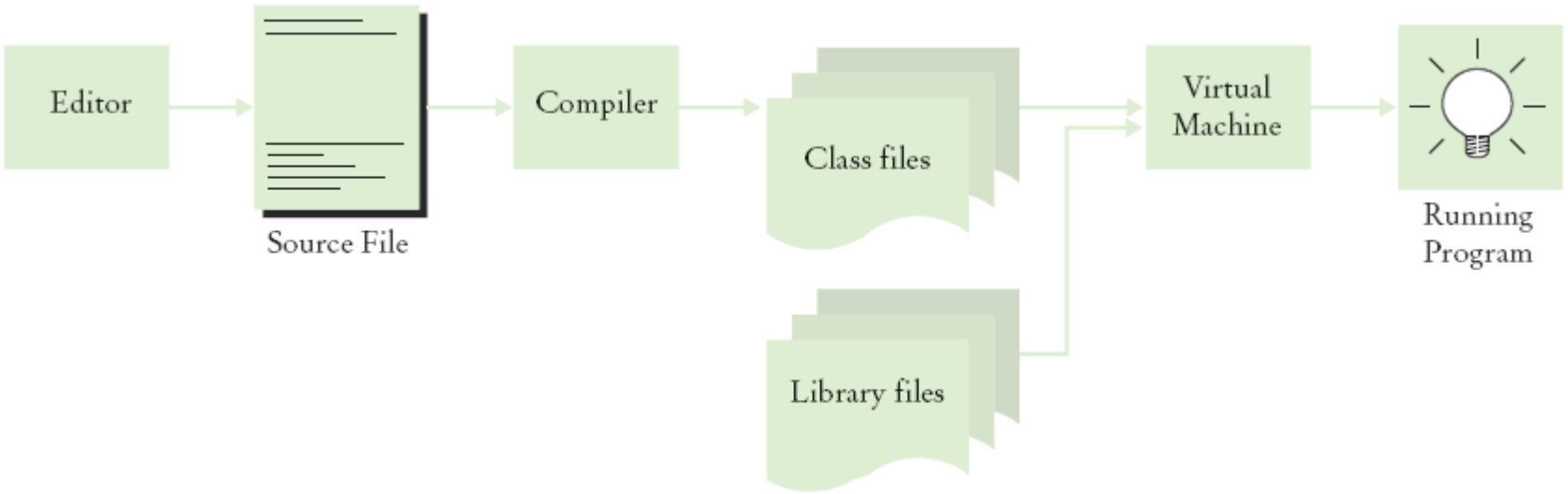
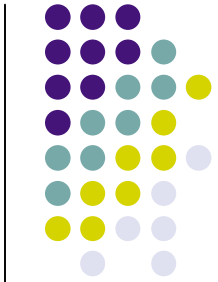


Figure 14 From Source Code to Running Program

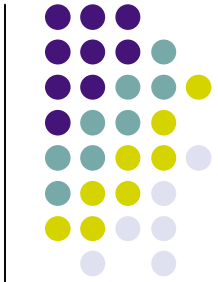
`HelloTester` Program



```
public class HelloTester {

  public static void main( String[] args ) {
    // Display a greeting in the console window

    System.out.println( "Hello, World!" );
  }

}
```

- Note:
  - Java is *case-sensitive* (upper/lowercase matters!)
  - Java doesn't care about layout (spaces/new lines)
    - But we (human beings) need proper layout in order to easily read and understand programs

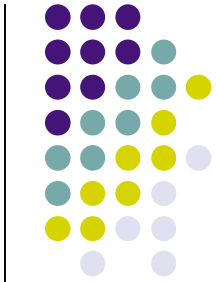
Basic Java Concepts



- Classes
  - Fundamental building blocks of Java programs
  - Every program made up of one or (usually) more classes

- Methods
  - Collection of programming instructions (statements) that describe how to carry out a particular task
  - Every method has a name
  - Every Java application needs at least a `main` method (i.e. a method named ‘main’)

Plumbing



- For now, just understand the following skeleton to be the basic ‘plumbing’ necessary for writing a Java program
  - Name the class according to what it is for
  - Fill in your instructions where the dots are

```
public class ClassName {

  public static void main( String[] args ) {
    ...
  }

}
```

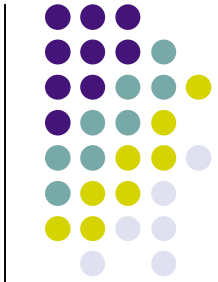
`HelloTester` Program

- Defines a new *class*
- 'public' means usable by everyone

```
public class HelloTester {

  public static void main( String[] args ) {
    // Display a greeting in the console window

    System.out.println( "Hello, World!" );
  }

}
```

- Every Java source code file can contain at most one public class, and name of the public class must match (spelling & capitalization) the name of the file containing the class (with .java extension)

HelloTester Program

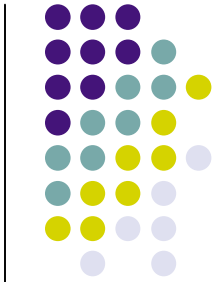


- Defines a *method* called 'main'
- 'static' means the method does not operate on an *object*

```
public class HelloTester {

  public static void main( String[] args ) {
    // Display a greeting in the console window

    System.out.println( "Hello, World!" );
  }

}
```

- The 'args' *parameter* is a required part of the main method - contains command-line arguments which we will not use for now

HelloTester Program



• This line is a *comment*

```
public class HelloTester {

  public static void main( String[] args ) {
    // Display a greeting in the console window

    System.out.println( "Hello, World!" );
  }

}
```

- Comments are purely for benefit of human readers to explain in more detail some part of the code
- All text between // and the end of the line is completely ignored by compiler

HelloTester Program

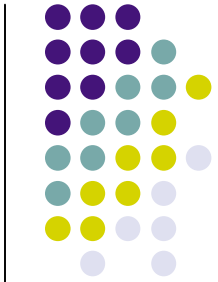
- This *statement* prints a message on the screen

```
public class HelloTester {

   public static void main( String[] args ) {
      // Display a greeting in the console window

      System.out.println( "Hello, World!" );
   }

}
```

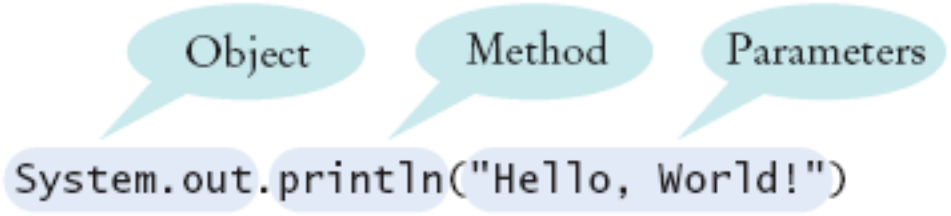
Statements

- Each statement ends with a semicolon
  - Forgetting ; is a very common error and will confuse the compiler
- Statements inside the *body* of a method (i.e. enclosed between braces { }) are executed one by one
- `println` is a method that prints a line of text
  - Destination of the output could be a file, window, networked computer, printer, …
  - We specify the *console* (terminal) *window* using the `out` object, contained in the `System` class

Invoking a Method

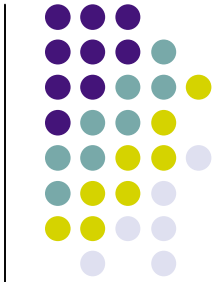


- To do something with an object, you *call* (or, *invoke*) a method by specifying
  - Object you want to use
  - Name of the method you want to use
  - Pair of parentheses, containing additional information the method needs to operate
    - This additional information is called the *parameter(s)*



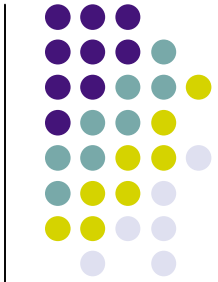
  - (Notice different meanings of the periods)

Strings



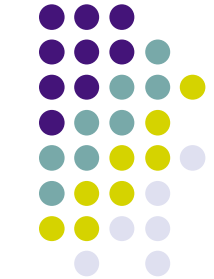
- Sequence of characters enclosed in quotation marks
  - “Hello, World!”
  - “main”
- Any text strings must be enclosed in quotation marks so compiler treats them as plain text and doesn’t try to interpret them as program instructions

Comments



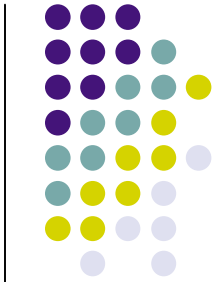
- Two forms of comments in Java
  - `// … till end of line`
  - `/*  … between … */`
    - This form of comments can span multiple lines
- Use comments as you are writing your code
  - Include header comment at the top of any source code files you work on
- Skeleton class file for all assignments
  - GenericClass.java
  - Example: HelloProblem.java

Errors



- Syntax errors (compile-time errors)
  - Violation of rules of form, detected by compiler
- Logic errors (semantic/run-time errors)
  - Program does something you did not intend when it runs
  - Harder to track down (compiler can't detect)
- Defensive programming
  - Structuring programs and development so that errors are isolated to small parts of program

Predictions that didn't make it



- "I think there is a world market for maybe five computers." – Thomas Watson, IBM chair, 1943
- "Where … the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons." – *Popular Mechanics*, 1949
- "Folks, the Mac platform is through—totally." – John C. Dvorak, *PC Magazine*, 1998
- "There is no reason anyone would want a computer in their home." – Ken Olsen, Digital Equipment Corp. president, chairman, and founder, 1977
- "I predict the Internet .. will go spectacularly supernova and in 1996 catastrophically collapse." – Bob Metcalfe, 3Com founder, 1995

L. Kappelman, "The Future is Ours," CACM 44:3 (2001), p46