Mat 2170
WEEK 1

Dr. Van Cleave

Spring 2014

Mat2170 Course Goals

- **Develop Algorithm Design Skills**:
  writing step-by-step instructions to solve problems
- **Develop Facility with the Object Oriented Paradigm**:
  using, extending, and developing Classes and Objects
- **Learn a Subset of the Syntax of the Java language**:
  be capable of writing significant Java programs
- **Develop Critical Thinking Skills**:
  the processes of discernment, analysis and evaluation of information

General Course Guidelines

- Syllabus
- Schedule (note **evening** exams)
- Academic Integrity
- Labs – weekly
- Quizzes, Worksheets – weekly
- Course Web Site (`www.eiu.edu/~mathcs`)

Lab Guidelines

- Focus on lab work when in lab.
- Come to lab prepared, with **written drafts of programs**.
- **Cheating is not allowed**. **Do your own work**.
- Unexcused late lab submissions will **not** be accepted.
- Not all labs are worth the same number of points.
- Finish incomplete labs on your own time when neccessary.

Evaluation

In this course there will be:

- Weekly — labs, worksheets, and quizzes
- Three written evening exams, and
- A comprehensive final exam

The relative weights of these components are:

Exams (3)	15% (each)
Quizzes, Worksheets	10% (total)
Laboratories & Projects	15% (total)
Final	30%

Your Responsibilities for the Semester

- Attendance — all lectures, labs, and exams
- Investing enough time on the course to succeed – about 15 hours per week outside of class. That's $>$ 2 hrs per day!
- Get help when you need it. Ask me questions. Come to my office. Send me email.
- Do your own work.
- Read the text & study the lecture slides.

More Responsibilities

- **Keep up** with the work. Turn assignments in on time.
- Turn off your cell phone and all other electronic devices, put them away, and keep them out of my sight during lectures and labs.
- Make-up exams are available only if agreed upon before the regular exam is given.
- No make-up quizzes will be given.

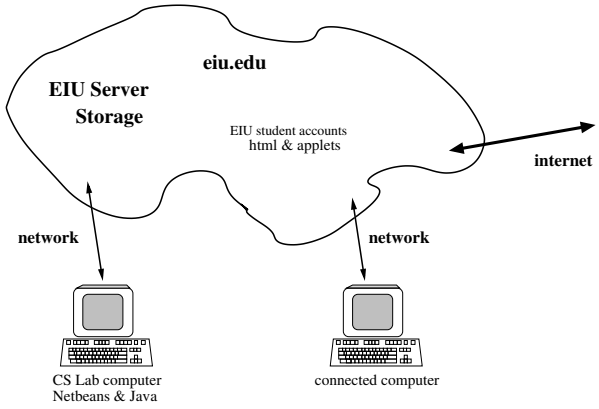
Week 1 Student Responsibilities

- **Reading**: Textbook, Chapters 1 and 2.1
- **Worksheet**: Worksheet 1
- **Lab**: Lab 1
- **Web publishing** of individual projects
- **Electronic submission** of entire Lab 1 folder
- **Attendance**: lecture & lab
- **Login** to your account in OM3041 Mac Lab before class Wednesday and report any problems to me asap

Week 1 Topics

- Getting ready for Lab 1:
  - Handouts — follow the directions; use the check–off boxes.
  - The Mathematics and Computer Science lab, OM 3041
  - Requirements
  - Netbeans, Java
  - `Hello World` Program
- Algorithms
- The Programming Process
- What is Computer Science?
- Computer Hardware

The CS Lab and EIU servers



An Overview of What You'll Need In Lab

1. An `EIU` student account, web page (automatically created), and your (email) **password**
2. The **Lab 1** and **Creating Java Programs...** Handouts
3. **acmLibrary.jar** — a file containing the ACM graphics library, which we have extended. It is available on our web site.

4. **netbeans** — an IDE used to create java programs and applets
5. **JDK** — the java interpreter
6. **safari** — a web browser
7. **WEB** — A way to transfer files from the lab to your web page
8. **SUB-MIT** — A way to electronically submit files for grading
9. Access to a printer

The `Hello World` Program (Java)

```
// Header comments go here
import acm.graphics.*;
import acm.program.*;

public class HelloProgram extends GraphicsProgram{
   public void run(){
       // Create and display a phrase to the user
       add(new GLabel("hello, world", 100, 75));
   } // end of run()

   public static void main(String[] args){
       new HelloProgram().start(args);
   } // end of main()
} // end of class HelloProgram
```


2170 programming and `acmLibrary.jar`

- The **Association for Computing Machinery** provides free java libraries (contained in **acmLibrary.jar**) which we will be using this semester.
- This library supports graphics, graphical user interfaces, and event–driven programming.
- We have extended this library to include even more helpful files.
- Programs can be more interesting and fun if we extend what others have written.
- Much more information is available at `jtf.acm.org`

The Integrated Development Environment (IDE)

- An **IDE** provides an organized way to:
  - view and select files from a project
  - edit files, and
  - compile and run programs
- There are multiple IDE choices — we will use `netbeans`
- `netbeans` is freely available from *S*un Microsystems
- To start up `netbeans`, click the bluish–grey cube on your dock
- `netbeans` itself has several windows and menus — follow the lab handout carefully.

The Java Developers Kit (JDK)

- Provides the Java compiler, which `netbeans` accesses.
- `JDK` is freely available from *S*un Microsystems
- We are currently using version JDK 1.6
- Programs are compiled into the "machine language" of the Java Virtual Machine (JVM).
- Java then interprets those programs by simulating the JVM.

Your Web Page

- When we compile and run a Java program from `netbeans`, we call that version an **application**.
- It is also possible to write **applets**, small interactive programs that run under the control of a web browser.
- EIU provides students with email and a web page.
- Part of your responsibility for labs will be to update your web page to include applets for each program and verify they have published correctly.
- Follow the instructions given in this week's lab and handout. Refer back to them as needed in subsequent weeks.

Algorithms

- An **Algorithm** is a set of instructions for solving a problem — much like a recipe for a particular dish, or the instructions for putting together a model airplane.
- An Algorithm is the underlying **logic** behind any program.
- **Algorithmic Properties**
  - A **Step–by–step method** for solving a problem
  - All steps must be **unambiguous** and **executable**
  - Must **terminate** with the **correct outcome**

The Programming Process

- **Algorithmic Design**
  - **Specifications** – types and restrictions of all required input and output for the program
  - **Test Suite** – well-selected inputs with **expected** outputs
  - **Logic** which solves problem (human readable)
    - General (Outline)
    - Detailed
- **Software**
  - **Coding** – translating Detailed algorithm into computer language (JAVA)
  - **Debugging** – locating and eliminating errors
  - **Maintenance** – evolution of program over time

Programming Errors and Debugging

- **Syntax error**
  - Violation of the grammatical rules of a language
  - Compiler displays error message(s)
  - Corrected by tracking error down and editing the program file
- **Logic/semantic error**:
  - Sometimes called a **bug**; the process of eliminating such errors is called **debugging**
  - Logic errors are much harder to find and eliminate than syntax errors
  - Good design and testing is essential to writing robust software
  - Time spent on design is well worth it

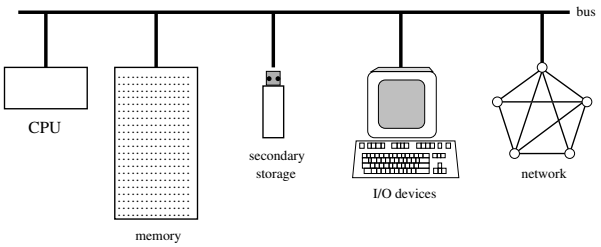
Software Maintenance

- Between 80% and 90% of total software cost is for maintenance **after** it has been released
- Reasons software requires maintenance:
  - Continued debugging over time
  - feature enhancement - updates requested by users or to compete in the marketplace
- Two Philosophies of Writing Programs
  - **Quick and Dirty** — get the program working and move on to next project
  - **Software Engineering** — the discipline of writing programs so they can be understood and maintained by others
- Programming is an art and skill – **learned by practice**, not rote memorization, much like playing the piano

What is Computer Science?

- **Hardware** - tangibles; the computer parts we can hold and feel
- **Software** - abstract, intangible
- **Problem Solving** - a **skill** one needs to **practice** in order to develop

Components of a Typical Computer



Computer Hardware

- **CPU** (**Central Processing Unit**) - an integrated circuit on a silicon chip; computations, coordinates computer activities
- **Memory** (**Primary Storage**) - usually a special integrated-circuit chip called a **RAM**, or *random-access memory*; information lost when machine turned off
- **Secondary Storage** - hard disk, thumb drive, CDs, diskettes, etc.; permanent data storage
- **Input/Output Devices** (**I/O devices**) - keyboard, mouse, monitor, printer
- **Network** - connection to other computers, **Internet**

Java & the Object Oriented Paradigm

- **Paradigm**: an existing theoretical framework or set of rules
- **Paradigm Shift**: a new idea/framework replaces an older one
- Old programming paradigm: **procedural** — programs were a series of statements, procedures and functions which operated on openly available data
- New programming paradigm: **object oriented** — data and operations are grouped together into integrated units called **objects**, providing some security for data integrity
- Each **object** is an instance of a particular **class**; a single class can serve as a pattern for many different objects.

Why Java?

- Used on the **AP exam** (of concern to HS teachers)
- **Simple**, efficient object oriented language
- Capacity to access and expand **libraries** of code, such as the **acmLibrary.jar**
- **Robust** and **Secure**: Designed for creating highly reliable software, with security features designed into the language and run–time system
- **Architecture Neutral** and **Portable**: Java was designed to work well over a network, regardless of machine type or operating system (multi-platform)

Why Java?

- **High Performance**: runs fast, responds quickly, cleans up after itself
- **Interpreted**, **Threaded**, and **Dynamic**: faster program development, multiple activities at the same time, and constantly evolving
- **Cost–effective**: open–source freeware is available on the Internet