Welcome!

° What is this class about?
  • Java programming ← somewhat
  • “software” ← somewhat
  • Solving engineering problems with programming ← our main focus!

° What will I get out of this class
  • A better understanding of specifying engineering problems
  • Techniques for specifying engineering problems in Java
  • A good understanding of the fundamentals of Java
**Course resources**

- The course web site contains important information associated with the course.
  - Check the site frequently for homework info, assignments, tips, lectures slides, etc.
  - Course web site: http://www.ecs.umass.edu/ece/ece122/

- Course text: Java Software Solutions, fourth edition, Lewis and Loftus, Addison-Wesley
  - You will only be responsible for material discussed in lecture and in discussion
  - When in doubt refer to the lecture slides.

- Course software:
  - DrJava – Its free from drjava.org!

---

**Course staff**

- Prof. Russell Tessier - lecturer
- Graduate TAs
  - Deepak Unnikrishnan
  - Ramakrishna Vadlamani
  - Ibis Benito
- Undergraduate TAs
  - Lukas Root
  - Michael Haines
  - TBD
Grade Composition

- 3 exams. 20%, 20%, 24%
- 6 homeworks 2% each
- 6 labs. 4% each

Assignments must be turned in on time!
- Excuse 1: The server was down – not likely
- Excuse 2: I “accidently” uploaded the wrong files – check programs immediately before submission
- Excuse 3: My network connection went down – make sure assignment is uploaded early

ENGIN112/ECE122 Comparison

<table>
<thead>
<tr>
<th></th>
<th>ENGIN112</th>
<th>ECE122</th>
</tr>
</thead>
<tbody>
<tr>
<td>No late assignments</td>
<td>![ENGIN112]</td>
<td>![ECE122]</td>
</tr>
<tr>
<td>Arrive to lecture on time</td>
<td>![ENGIN112]</td>
<td>![ECE122]</td>
</tr>
<tr>
<td>Most information from lecture slides</td>
<td>![ENGIN112]</td>
<td>![ECE122]</td>
</tr>
<tr>
<td>Well dressed</td>
<td>![ENGIN112]</td>
<td>![ECE122]</td>
</tr>
<tr>
<td>Taller</td>
<td>![ENGIN112]</td>
<td>![ECE122]</td>
</tr>
</tbody>
</table>
Brief Rehash of ENGIN112: Digital Systems

- Information can be represented by binary digits
  - 0 and 1
- We can build digital systems with logic gates

\[
\begin{array}{c}
A \\
B \\
\hline
Y
\end{array}
\]

- Groups of gates can perform computation

\[
\text{ALU} \quad \text{Function} \quad \text{Conditions} \quad \text{DataA} \quad \text{DataB} \quad \text{DataOut}
\]

How do we often solve task-based problems?

- Example: A trip to the store
  1. Get out of bed
  2. Get dressed
  3. Make list of items
  4. Get into car
  5. Drive to store
  6. Pick items off shelf
  7. Go to register
  8. Drive home

- Observations
  - Activities take place sequentially, one at a time
  - The order of the operations matter
  - If the “data” is the same. I should get the same result every time.

- Question: How does a computer solve problems?
Datapath I/O

- A wire can be driven by only one tri-state at a time
  - If InPass is active, AluPass must be inactive
  - If AluPass is active, InPass must be inactive

<table>
<thead>
<tr>
<th>Function</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadY</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LoadX</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InPass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutPass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AluPass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two values enter from the left (A and B)

- Need to perform (A+B)+A
- In -> X (Load A)
- In -> Y (Load B)
- A+B -> Y
- (A+B)+A -> Out

Four steps and then repeat
### Implementing the Control ROM

- Two values enter from the left (A and B)
  - Need to perform \((A+B)+A\)
  - \(X \rightarrow X\) (Load A) - State 00
  - \(Y \rightarrow Y\) (Load B) - State 01
  - \(A+B \rightarrow Y\) - State 10
  - \((A+B)+A \rightarrow\) Out - State 11

![Control ROM Chart](image)

---

### Datapath and Control ROM

- Control of data path with ROM
  - ROM contains “program”

![Datapath Diagram](image)
What is Programming Language?

- A programming language allows programmers to specify tasks for the computer **one task at a time**.
  - Just like most things in life
- The computer then “executes” this sequence of tasks on hardware to generate a result
  - For the same data, the computer should get the same result every time.
- To solve engineering problems we need to:
  - Specify the problem as a series of simple tasks
  - Know how to specify these tasks (syntax, rules, format, tools, etc).
- Programming language for this semester: Java!

Java Program Structure

```java
// comments about the class
public class MyProgram
{
    class header

    class body

    Comments can be placed almost anywhere
}
```
Java Program Structure

// comments about the class
public class MyProgram {
    // comments about the method
    public static void main (String[] args) {
        // method body
    }
}

In the Java programming language:
- A program is made up of one or more classes
- A class contains one or more methods
- A method contains program statements

These terms will be explored in detail throughout the course

A Java application always contains a method called main
A Java Program

/* Lewis/Loftus  6/11/2003
   Lincoln.java demonstrates the basic structure of a Java Application.
*/

public class Lincoln
{
    //  starts the class definition
    // Prints a presidential quote.
    public static void main (String[] args) // this is a method name
    {
        System.out.println ("A quote by Abraham Lincoln:";)
            // invokes another method
        System.out.println ("Whatever you are, be a good one.");
            // invokes another method
    }
    // ends the class definition
}

What to Notice in Lincoln.java

° The first few lines between /* and */ are comments.
  • Comments don't affect what the program does but are included to make the program easier to understand.
  • Comments included in a program are called inline documentation.
  • The comment symbols /* and */ can surround a multiline comment.
  • The comment symbol // can be used at the beginning of a single-line comment.
Comments

- Comments are notes in the text of a program to assist that/any programmer who must later modify or simply understand the code.
  - The original programmer often will not remember later the details of the program he or she has written.
- Comments within the text of a program are referred to as internal documentation.
  - Good internal and external documentation are essential.

Two Types of Comments

- A single-line comment in Java takes the form:
  - `//` This is a comment.
- A multi-line comment takes the form:
  ```
  /*
   * This is one comment
   * that spans several lines.
  */
  ```
  - It is also permissible to use the above notation for a single-line comment.
    ```
    /* This is a comment */
    ```
Good Programming Practices

- Appendix F of our text presents guidelines for good programming practices
  - Specific techniques for documenting programs.
  - Be Verbose
- You can assume the reader is computer literate and familiar with Java.
  - Assume the reader knows almost nothing about what the program is supposed to do.
  - You won’t be able to recognize what your program does after (at most) one week.

Identifiers and Reserved Words

- The words used when writing programs are called identifiers.
  - There are three categories of identifiers:
    1. Words that we make up (Lincoln and args)
    2. Words that another programmer chose (String, System, out, println, and main).
      - Often these are words that were chosen by a programmer for code that became part of a Java library.
      - Now available for use by all Java programmers.
    3. Words that are reserved for special purposes in the language (class, public, static, and void).
      - The designers of the Java language chose these words.
The Reserved Words

Reserved words are identifiers that have special meaning in a programming language.

- In the Lincoln program, they are class, public, static, and void.

```java
public class Lincoln {
    public static void main (String[] args) {
        System.out.println ("A quote by Abraham Lincoln:");
        System.out.println ("Whatever you are, be a good one.");
    }
}
```

Reserved Words

- The Java reserved words – these words cannot be used for any other purpose, such as naming a class or method.

```java
abstract else interface super
boolean extends long switch
break false native synchronized
byte final new this
case finally null throw
catch float package throws
char for private transient
class goto protected true
const if public try
continue implements return void
default import short volatile
do instanceof static while
double int strictfp
```
White Space

- Spaces, blank lines, and tabs are called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation

Summary

- Programs are a series of tasks for a computer to perform.
- A programming language defines how we specify the tasks
- Important to use many comments and to understand identifiers

- Reading: L+L: 1.1 – 1.4, Download and play with DrJava