

CSI333 Lecture 1

- ▶ A CLASS is a datatype.
A CLASS is a master pattern (like a rubber stamp) for creating one or more OBJECTS, together with code and interface definitions for OPERATIONS (called METHODS) on those objects.
- ▶ An OBJECT or CLASS INSTANCE is some ACTUAL DATA plus a way to run its methods on that data.

Object oriented programming and software design: Most or all pieces of data are CLOSELY ASSOCIATED WITH operations on that data.

But, within computer hardware, that data and those operations are quite different.

Time used for Q/A

- ▶ Stamp, method cloud, instance, Java reference/C++ pointer variable, instantiation, method invocation, pointing, sketched on blackboard. reference/pointer likened to a remote control, from “Head First Java”.
- ▶ Distinction between declaring a name to signify a variable, and the allocation of the variable.
- ▶ Distinction between object instance and a pointer/reference to the object.

Abstraction—One recurring theme from PP Chap. 1

I propose this definition: The act or the result of judiciously ignoring details by inventing and using a concept and a name instead of the well-defined body of details the abstraction replaces.

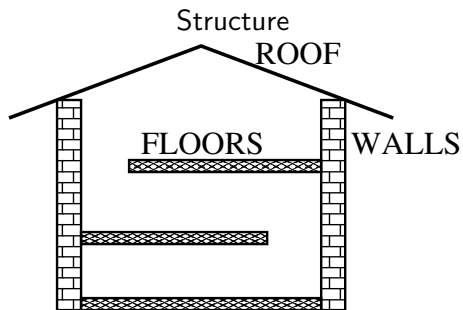
Examples:

1. Each name (and signature) of a method is an abstraction for the code you wrote in the method body.
2. A class is often an abstraction for some entities in the real world, like dates, shapes, personnel records, etc.
3. An abstract data type (like a STACK) is an abstraction for multiple choices of ways to implement it (like by a linked list versus an array).
4. Read the discussion and examples in PP Chap. 1.

Programming-shown quickly-please repeat

- ▶ Programming is basically translating an ALGORITHM into a practical PROGRAM to make a computer carry out the algorithm.
- ▶ Algorithm usually means a fairly short composition of directions, where each step is simple to carry out, that is presented to solve a usually simple and well-defined problem.
- ▶ “Algorithm” also indicates that the directions are guaranteed (by whom?) to terminate no matter what is the input. (Otherwise, the directions should be called a “computational procedure”.)
- ▶ Advanced programming and software development goes beyond this because what big programs and systems do is too complicated and often faulty and ill-defined to be expressed by a single problem to be solved by an algorithm.

House Architecture One

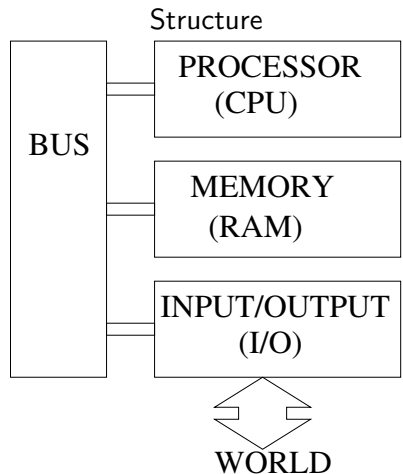


Function

Provide sheltered living spaces for people.

Let's see what we mean by architecture. Here's a very much oversimplified idea of architecture of buildings. Architects talk about form and function.

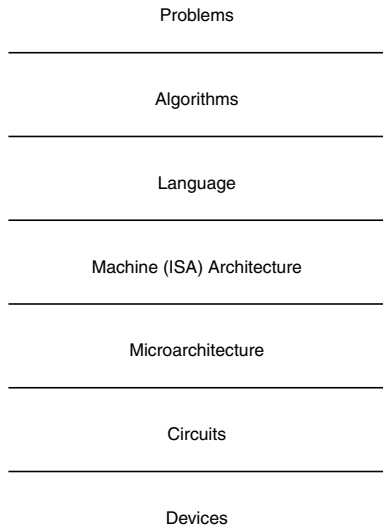
Computer Architecture One



Function

Communicate, store and process **digitally represented** data under **digitally stored** program control

Computer System Architecture—Layered View



Progression from Problem to Program: Expressions

An *expression* expresses a calculation on numbers or other kinds of values (like logic or *Boolean* values). People learn:

$$39.99 + 119.95 * (100\% - 50\%) - 5.00 + 950.00 * (100\% - 10\%)$$

Do the innermost *subexpressions* first:

$$39.99 + 119.95 * 50\% - 5.00 + 950.00 * 90\%$$

$$39.99 + 119.95 * (100\% - 50\%) - 5.00 + 950.00 * (100\% - 10\%)$$

$$39.99 + 119.95 * 50\% - 5.00 + 950.00 * 90\%$$

Now the 2 multiplications are innermost, because, when add and multiply compete, multiply wins:

$$39.99 + 59.975 - 5.00 + 855.00$$

(we say “* has greater *precedence* than +”)

The left addition can be done first:

$$99.965 - 5.00 + 855.00$$

The above means subtract 5.00 from 99.965 then add 855.00.

$$94.965 + 855.00$$

Finish:

$$954.965$$

What is the Computer Science Problem?

- ▶ Computers ain't that smart.
- ▶ Is it a good idea to attempt an algorithm the emulates *exactly* what a person does?
- ▶ What algorithm should be programmed into an electronic *calculator*?
- ▶ Can your algorithm *scale up* to handle large problem instances efficiently?
- ▶ Can ideas about expressions and algorithms to process them be applied to other situations? **YES!** Most programming languages, and the famous SGML/HTML/XML family of languages have the *Nested Expression* kind of syntax.

The Calculator Problem

Let's start with SIMPLE CASES.

3

The Calculator Problem

Let's start with SIMPLE CASES.

3

3

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

3

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

3 That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one:

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: 3

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 +$

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 +$

The Value box stores the previously inputted values while new values and operators are read.

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4$

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed.

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4$

4
3

+

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed.

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4 =$

4
3

+

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed.

The Calculator Problem

Let's start with SIMPLE CASES.

3 =

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4 =$

3

4

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed.

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4 =$

3 + 4

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed.

Operators and values are REMOVED after they are used.

The Calculator Problem

Let's start with SIMPLE CASES.

$$3 =$$

That's the answer. Easy. Let's try $3 + 4 =$
Key it in, one by one: $3 + 4 =$

3 + 4

The Value box stores the previously inputted values while new values and operators are read. The Operator box stores previously inputted operators while new input is processed. Operators and values are REMOVED after they are used.

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

39.99

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

39.99 +

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95 *$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95 * ($$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95 * (100\%$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95 * (100\% -$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

$$39.99 + 119.95 * (100\% - 50\%)$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

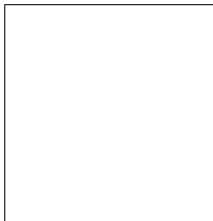
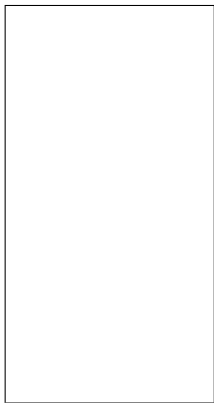
$$39.99 + 119.95 * (100\% - 50\%)$$

The Calculator Problem

Some data is entered *before* the rest of the data. The calculator cannot tell what the user will enter next, but it must remember data already entered.

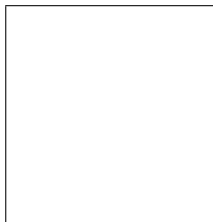
$$39.99 + 119.95 * (100\% - 50\%) - 5.00 + 950.00 * (100\% - 10\%)$$

39.99



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

39.99



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

39.99 +



39.99

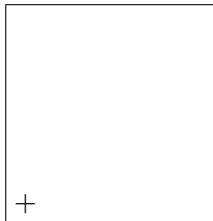


- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

39.99 +



39.99



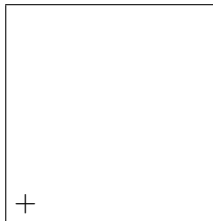
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95$$



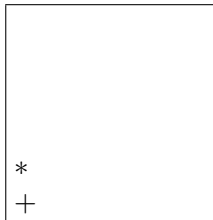
39.99



+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 *$$



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\%$$

100%
119.95
39.99

(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% -$$

100%
119.95
39.99

(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% -$$

100%
119.95
39.99

-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\%)$$

100%
119.95
39.99

-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\%)$$

50%
100%
119.95
39.99

-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% *$$

50%
100%
119.95
39.99

-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% *$$

50%
100%
119.95
39.99

*
-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$

50%
100%
119.95
39.99

*
-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$

0.2
50%
100%
119.95
39.99

*
-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$

0.2
50%
100%
119.95
39.99

*
-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box;

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$

10%

100%
119.95
39.99

-
(
*
+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result)

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$

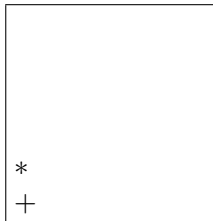
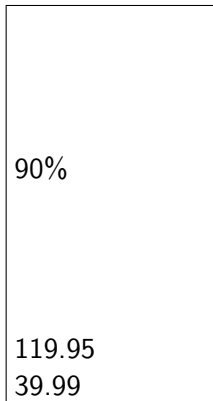
90%

119.95
39.99

(
*
+

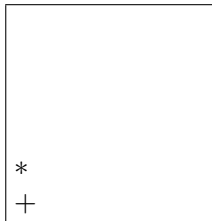
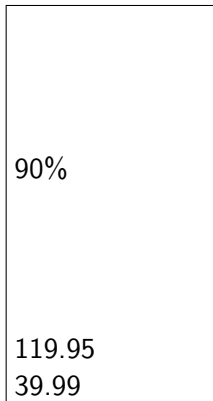
- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2)$$



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched "("

$$39.99 + 119.95 * (100\% - 50\% * 0.2) -$$



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched "("

$$39.99 + 119.95 * (100\% - 50\% * 0.2) -$$

107.95

39.99

+

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2) -$$

130.945

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2) -$$

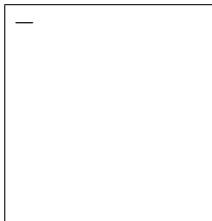
130.945

-

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2) - 5.00$$

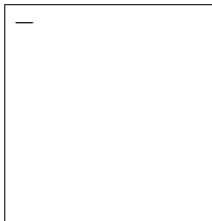
130.945



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2) - 5.00$$

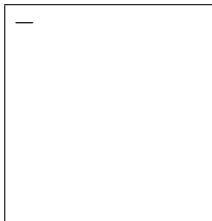
5.0
130.945



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

$$39.99 + 119.95 * (100\% - 50\% * 0.2) - 5.00\text{end}$$

5.0
130.945



- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched "("

$$39.99 + 119.95 * (100\% - 50\% * 0.2) - 5.00\text{end}$$

131.945

- ▶ Use a Value Box and an Operator Box to keep input (and results) until they are needed.
- ▶ Each step: Choose either (1) PUSH current input into relevant box; OR (2) REDUCE (replace values and operator with their result) OR (3) remove a matched “(”

Project 1

Your job is to understand the algorithm based on these ideas and then implement it (for Boolean, not numeric expressions) by a C++ or Java program. HOMEWORK: practice it on paper.

Observations:

1. The value or operator accessed or removed from the corresponding “box” is ALWAYS the LAST ONE put in the box but not yet removed. Those boxes can be *stacks*, i.e., pushdown or Last-In-First-Out (LIFO) stores.
2. The input is processed left-to-right.
3. The decision to PUSH, or REDUCE, or REMOVE a “(” is based only on the current input token and the top of the Operator stack.
4. When two operators affect the PUSH or REDUCE decision, the decision is based on their relative *precedence* (and associativity).