CSE 115 / 503
Introduction to Computer Science I

Carl Alphonce

cse-115-alphonce@cse.buffalo.edu

219 Bell Hall
Agenda

• Syllabus: the basics [15]
  – who, what, when, where, why & how
• Announcements [05]
• Schedule of topics [05]
• Advice [10]
• Activity [10]
• Questions [05]
Syllabus

• who: instructors, UTAs, grading TAs
• what: topic list on next slide
• when: lecture, lab (recitation), outside class
• where: lecture room, Baldy 21, up to you
• why:
  – understand cs problems & problem solving techniques
  – learn some cs history
  – appreciate breadth and depth of field
  – give societal context and impact of field
• how: lecture, labs, writing assignments, & your questions

READ SYLLABUS, BRING QUESTIONS TO NEXT CLASS, SIGN AND TURN IN SIGNATURE FORM
Lab announcement

• Labs for all sections EXCEPT Thursday lab sections begin next week.
• Labs for THURSDAY sections begin this week.

• Make sure you do your preparatory work BEFORE coming to lab.
Office hour / Help session announcement

• Regular office hours for the TAs begin next week.

• Help sessions begin next week.

• My office hours begin this week.
Schedule of topics

- Object-oriented (OO) problem solving (analysis, modeling, decomposition)
- OO programs as systems of interacting objects
- Objects (as having properties & behaviors, as service providers)
- Classes (as defining types, as defining objects – objects are class instances)
- UML class and object diagrams
- Types, variables, operators, expressions
- Messages (defined using methods), parameters & arguments
- Class relationships: has-a (aka composition or aggregation), knows-a (aka association)
- Class relationship: is-a (aka inheritance or generalization)
- Interfaces, Polymorphism
- Class relationship: implements-a (aka implementation or realization)
- NGP
- Class relationship: uses-a (aka dependency)
- Abstract Classes
- Arithmetic, Boolean algebra
- Selection and repetition (looping)
- Collections (e.g. HashMap, ArrayList) and generic collections
- Design patterns (e.g. Iterator, State, Singleton, Strategy, Composite, Decorator, Null Object)
Advice (1)

• This is NOT high school
  – freedom + responsibility
  – peer group is different: most of you had As in HS
  – HS average vs. university average:
    • C is “expected” grade, B is “above average”, A is “exceptional”
• 1 credit hour =
  – 1 (lecture) or 2 (lab) hours of contact time
  – 3 hours of non-contact time (i.e. outside class)
• This course: 4 credits = 17 hours average per week
  – 5 hours contact + 12 hours non-contact
Advice (2)

• Learn how to study (web site has resources)
• Studying must happen EVERY DAY, not just the night before an exam
• Ask questions whenever you have them (and you are expected to have lots of questions)
• Office hour attendance is the norm, not the exception
Activity

- Goal: demonstrate object behavior
- Needed: 3 volunteers
Questions?
Lecture #2

Agenda

• Announcements
• Questions?
• Computing environment
  – accounts
  – e-mail accounts/forwarding
• Web site tour (schedule + readings)
• Activity/Review
• Problem solving, starting with analysis, decomposition and modeling
Announcements

• Put out your name signs!
• Did you not get a handout last class?
  – Syllabus
  – Lab 1 preparatory work
• Cancelled labs: 115B3 [F 4-6], 503A2 [T 4-6] (not 115A2)
  • Register for another section. Check course web site for available sections; try to switch to B1, B2 or B4.
  • If you switch to any other section but want to keep this lecture, clear it with the TA for the section first (send them e-mail); they will notify me.
• Reminder: Thursday recitations start this week!
Questions?

• Syllabus questions?

• Other questions?
Computing environment

• UB computing account (UBUnix)
  
  - workstations
  - compute servers
  - file server
  - e-mail server
    @buffalo.edu

• CSE computing account
  
  - Baldy 19
  - compute server
  - file server
  - e-mail server
    @cse.buffalo.edu
CSE computing infrastructure
(partial picture)

- **file server**
  (armstrong.cse.buffalo.edu)

- **compute server**
  (everest.cse.buffalo.edu)
  - thin clients in Baldy 19

- **compute server**
  (yeager.cse.buffalo.edu)
  - thin clients in Baldy 21
  - remote sessions
E-mail forwarding: 4 choices

From @cse to @buffalo  
From @buffalo to @cse

From off-campus to on-campus  
From on-campus to off-campus

However you forward your mail, ALWAYS send e-mail destined for professors from a UB account: @cse or @buffalo
Web site tour

www.cse.buffalo.edu/faculty/adrienne/FA2004/cse115

Visit web site regularly (several times a week)

• announcements
• labs and other assignments
• schedule with readings
• resources
Activity / Review

• Why did we do this activity?

• What is an object?
OO systems: interacting objects

"Teaching object-orientation must start with a sufficiently complex example"
I stole this picture from...

Kristen Nygaard (8/27/1926-8/10/2002)
Developer of Simula and Simula67, the first OO language, together with Ole-Johan Dahl.

Also winner, together with Ole-Johan Dahl, of:

IEEE 2002 John von Neumann Medal
ACM 2001 Turing Award
Ole-Johan Dahl

Passed away in June 2002, about two months before Kristen Nygaard.
Simula

The 1965 SIMULA manual
The first object-oriented report
Problem solving

• How do we start to solve a problem?
  – analysis
  – decomposition
  – modeling

• What makes for a good solution?
  – Next time, but think about it!
Lecture #3

Agenda

• Prelude
  – Announcements
  – Review
  – Questions?

• New material
  – Connecting to CSE systems using XWin32 and SSH
  – Problem solving (analysis, decomposition, modeling)

• Postlude
  – Reminders
  – Questions?
Announcements

• Bell 101 (bad news and good news)
  – new this semester: need eng account to use
  – anyone can get an eng account (do it!)
• E-mail (and other CSE-specific info)
  www-local.cse.buffalo.edu/Consulting
Review (1)

- Objects have:
  - **properties**, which together define the internal state of the object, and
  - **behaviors**, which determine
    - what services the object is prepared to provide to other objects, or to put it another way
    - what messages the object will respond to.
Review (2)

• Different objects can have different sets of properties and behaviors, as demonstrated in our activity:
  – dog
  – robot
  – statue

• Object-oriented (OO) programs are viewed as systems of interacting entities.
Review (3)

- OO has its roots in the language Simula
- Simula was designed in early 1960’s by
  - Kristen Nygaard
  - Ole-Johan Dahl
- They jointly received computing’s highest awards for their work
- OO is about forty years old!
Questions?
XWin32 and SSH

Emacs editor
yeager.cse.buffalo.edu

encrypted tunnel with “X11 forwarding”

window for Emacs editor
your home/dorm machine

XClient runs on remote machine

SSH connects securely

XServer runs on local machine
Problem solving

• Problem solving is a methodical process
• Basic processes of software construction:
  – analysis to build a specification
  – decomposition
  – modeling
  – implementation
• Somewhat simplified: some processes omitted (for now)
Example (1)

Problem:
I am tired of walking everywhere. I want a device which will transport me from place to place.
Analysis

• Given a problem (expressed in the customer’s terms), how do you figure out what is needed to solve it?
• Communicate with client to uncover requirements.
• Result is a specification, expressed more formally, that spells out in detail what the final product must do.
Specification

“It should list each component of the system, including its capabilities and properties, and its interactions with other components.”

– Brown notes
Designing a Model

- The specification must be translated into a form that is understandable by a computer.
- To do this we first design a model of the system (a “simplified representation”)
- Our models will use the building blocks of OO programming to represent components of the system.
Example (2)

Specification?
Example (3)

Specification?

• How far does it need to go?
• How quickly?
• Indoor/outdoor
• Across water?
• Human powered or not: what fuel source?
• Cost?
• Size?
• Controls?
Example (4)

- Once specification is available, build a model.
- Once model is OK, build system.
- This can be done incrementally (design power source independently of entertainment system) and iteratively (specification/model/product can be refined through several iterations).
Next step?

• So we how a model – now what?

• We need to implement it!

• We will implement our models using a computer programming language called Java.
Reminders

- No class Monday: Labor day
- Read chapter 1 of the Brown notes BEFORE coming to class on Wednesday
Questions?
Lecture #4

Agenda

• Prelude
  – Announcements
  – Review
  – Questions?

• New material
  – Classes and objects
  – UML class diagrams
  – Creating objects (instantiating classes)
  – has-a relationship (lifetime)

• Postlude
  – Reminders
  – Questions?
Where do objects come from? (The “birds and bees” lecture)

Programmer writes a program in a high-level language like Java: MyApplet.java

Computers don’t understand programs expressed in high-level languages 😞
Compilation

Programmer writes a program in a high-level language like Java: MyApplet.java

A compiler translates program to an equivalent low-level form that a computer can understand 😊

MyApplet.class
Runtime

Compiler translates

Runtime refers to the time during which a program is executing, or running.
Objects exist only at runtime

Objects do not exist while the programmer writes the program, except in their minds.

Objects exist only at runtime
Huh?

- If objects are the basic building blocks of object-oriented programs, and programmers don’t directly manipulate objects, what do programmers write?

  - They write class definitions.
  - Objects are instances of classes.
  - Classes are instantiated only at runtime.
So what are class definitions?

• A class definition is a description of the properties and behaviors that instances of the class will have.
• Recall that we said a running OO program is a system of interacting objects.
• Possible relationships between objects are determined by relationships between classes.
UML class diagrams

(see lecture 5 slides for this material)
Lecture #5
Agenda

• Prelude
  – Announcements: lab open hours coming on-line
  – Review
    • Classes and objects
    • UML class diagrams (noon class)
  – Questions?

• New material
  – UML class diagrams (morning class)
  – Creating objects (instantiating classes)
  – has-a relationship (lifetime)

• Postlude
  – Reminders – keep up with readings, start work early
  – Questions?
UML class diagrams (1)

• UML = Unified Modeling Language
• We use only class diagrams, not other UML diagrams
• Purpose:
  – keep OO concepts separate from implementation language
  – operate at a more abstract level than programming language – avoid making implementation decisions when designing code
UML class diagrams (2)

- **Class box**
  - Class name
  - Properties
  - Behaviors

- **Relationships**
  - inheritance
  - implements
  - composition
  - dependency
  - association

- **Only as much detail as is needed**
Where do objects come from?

- Objects are instances of classes;
- We instantiate classes:
  - e.g. new BouncingBall()
  - There are three parts to this expression:
    - new
    - BouncingBall
    - ()
'new' is a "reserved word" in Java. This means that the word 'new' has a special meaning in the Java language.

'new' is the name of an operator whose job it is to create an instance of a given class.

'new BouncingBall()'
Lecture #6

- Slides to come
- We showed an example of a running Applet
Lecture #7

• Prelude
  – Announcements
    • No recitation on Thursday
    • All sections start lab 3 next week
  – Review
    • Class definitions, class instantiation/object creation, naming of objects, variable declarations, sending a message
  – Questions

• “New” material
  – Composition relationship
    • semantics (typically part/whole, lifetime issues)
    • syntax (expressing the relationship in Java)
  – Sending messages with arguments

• Postlude: Questions?
Lecture #8

• New material
  – Association relationship
    • semantics, lifetime
    • syntax (expressing the relationship in Java)
  – mutator methods
  – accessor methods
Lecture #9

• New material
  – nuts and bolts of writing (non-constructor) methods in Java
    • method header
      – access control modifier (acm)
      – return type specification (rts)
      – method name
      – parameter list (may be empty!)
    • method body
      – return statement required if rts is not void
      – flow of control returns to caller at a return; implicit return for void methods
Lecture #11
Lecture #12

• Quiz #1
Lecture #13

• Inheritance (is-a) relationship: fundamental relationship in OO
  – categorization (shapes, animals)
  – captures shared properties/behaviors
  – expressed with “extends” clause in Java
  – type inclusion
  – terminology: superclass – subclass, base class – derived class,
    parent class – child class
  – Java is a single-inheritance language
  – java.lang.Object is at the root of the class hierarchy
  – special status of Object (no parent)
  – special status of java.lang (classes always accessible with short name)
  – IMPORTANT: POLYMORPHISM (the “P” word)
Lecture #14

• practical effects of inheritance
  – non-private members are inherited into subclasses
  – subclass instances are superclass instances, and can be used in place of superclass instances
  – method overriding & super.method()
  – constructors and super()
  – method overloading
Lecture #15

• Exam review
Lecture #16

• Exam
• BRING YOUR UB CARD!!
Lecture #17

• Exam return?
Lecture #18

• POLYMORPHISM!