Announcements

- Final exam is on Wed, 12/14, 9am-11:30am, Room C103
  - Comprehensive although the topics since the second midterm will be treated more heavily
  - You may use a page of notes as we did in the midterm exams
  - JavaFX (GUI) will not be tested in the final

- Between now and the final, my office is open always!
  - Come get help if you have any questions

Review

- This list in this slide set is compiled not to go through in any detail in class, but to suggest to you what to review as you prepare for the final.

- We will quickly skim together in class while emphasizing the important ones, particularly for the final.

Goals (from the first lecture)

- Get an introduction to computer science
- Learn how to solve a problem by:
  - defining the problem
  - developing a solution (develop an algorithm)
  - implementing the solution by writing a computer program
  - testing and fixing the programming solution
- Learn to program in Java
- Provide a healthy mix of the practical and theory
You learned Java, what about other languages?

- What you learned with Java will easily translate to (almost) any programming language.
- That is, the concepts that you learned this semester will be the basis of many, if not all, existing programming languages as well as the ones that you will see in the future.

Using classes built by someone else

- System
- String
- Math
- Scanner
- Random
- BufferedReader
- FileReader
- ...

Building classes of our own . . .

- Point
- Circle
- Student
- Account
- Ipad
- Message
- Mailbox
- Player
- Team
- Car, PassengerCar, SportsCar
- Person, Employee, Manager, Faculty, Staff, etc.
- ...

Class

- Fields
- Constructors
- Methods
- A way to encapsulate the state and behavior of an entity (abstract or concrete)
Method
- Models behavior of objects
- By invoking methods in an object, you are accessing (viewing/modifying) the state of the object
- Signature of a method
- Accessor methods (getters/readers)
- Mutator methods (setters/writers)
- Parameter passing in Java
  - passing primitive values by value
  - passing object references by value

Constructor
- Difference between a constructor and a method
- Can be overloaded much like a method

Names in programming
- Or, more specifically in Java:
  - Class names
  - Method names
  - Variable names
    - Fields (members) in a class
      - Static variables (class variables, static fields)
      - Non-static variables (instance variables, non-static fields)
    - Local variables in a method
      - Parameters and local variables are treated the same

Visibility control
- How do we control the visibility of the state information in an object?
- How does this relate to the concept of information hiding?
- How does this relate to the concept of encapsulation?
- How does the concept of a class relate to the concept of encapsulation?
- private
- protected
- public
Scope and lifetime of a variable

- Scope and lifetime of a static field
- Scope and lifetime of a non-static (dynamic) field
- Scope and lifetime of a local variable
- Scope and lifetime of a parameter

What is a primitive (or built-in) data type?

- What are those that we have seen?
- What are those that Java supports?

What is a non-primitive data type, i.e., user-defined data type?

- They are object types such as
  - `Point` type
  - `Student` type
  - `Account` type
  - etc.

Is `String` a primitive data type or an object data type?

- Some operations on `Strings`
  - +
  - substring
  - `charAt`
  - `toLowerCase`
  - length
  - split
  - . . .
Can an object reference be a field (static or non-static) of an object?

- Of course, that is how "object composition" is implemented in Java
- Give an example
- Can an object reference be a parameter to a method? Give an example
- Can an object reference be a local variable? Give an example
- How big is an object reference (in number of bits)?
- How big is an int value (in number of bits)?

What is an expression in a programming language like Java?

- Is 45 an expression?
- 45 + 23
- x (an int variable)
- b (a boolean variable)
- x/y (where x and y are int's)
- (x < y) && (x > z)
- (a || b) (where a and b are boolean's)
- !(a && b) (where a and b are boolean's)
- s1 + s2 (where s1 and s2 are two String variables)
- ...

Some basic operators

- that are used to build more complex expressions:
  - +
  - -
  - *
  - /
  - %
  - ...
  - with ( and )

What is an expression in a programming language like Java?

- An expression produces a value when evaluated
- A value is stored in a location in memory
- Computation is really storing values in memory locations in some meaningful ways (storing and updating as computation continues)
- Assignment statement (the '=' operation) is how you store/update a value in a memory location
Operator precedence

- What is the value of the following expression?
  \[3 + 4 \times x - y / 4 + 4\]

What is a statement?

- What is the difference between an expression and a statement?
- Give a few examples of each

Statements in Java we are familiar with

- `System.out.print`
- `System.out.println`
- `if...then...else`
- `break`
- `return`
- `=` (assignment statement)
- `while` loop
- `for` loop (two different kinds)
- `method` calls
- `try...catch...`
- `compound` statements
- ...

Conditionals in Java

- `if...then`
- `if...then...else...`
Looping constructs

- while
- for
- for (‘for-each’)
- Using recursion to loop

Data structures

- Simple primitive ones
  - int, double, char, boolean, ...
- Compound data structures
  - Objects containing fields of various types
  - Arrays containing data/objects of a certain type
    - linear arrays
    - 2D arrays
    - arrays are objects!
    - array of arrays

Loops and arrays

- Use of while or for or recursion to iterate over elements in an array (1D or 2D arrays)

Exception handling

- try . . . catch . . . finally
I/O

- Standard I/O
  - Keyboard input
    - System.in
    - java.util.Scanner
  - Console/Screen output
    - System.out.print
    - System.out.println
- File I/O
  - Open a file
  - Read from or write to it
  - Close it

Memory model for:

- Primitive data types in an object
- Primitive data types in a method as a parameter or a local variable
- Object reference in any context: a variable of any object type is really just an object reference of 32 bits and the actual object is sitting in another location in memory (free store, aka heap) referenced by those 32 bits.
- Fields in an object (static vs. non-static)
- Arrays (Arrays are objects too)

Complexity of data that you can deal with in your program

- An object containing an array in it.
- That array containing a set of objects
- Those objects containing other objects including arrays
- Those objects and arrays in turn including others,
  - etc. etc. etc.

Method overloading

- foo(int x);
- foo(boolean x);
- foo(int x, double y);
Method calls (both static and non-static)

- The dot (.) notation
- Internal calls without the '.' notation
- External calls with the '.' notation
- A dot used with a dynamic object
- A dot used with a static object (a class name followed by a dot followed by a method name with actual arguments)

The object references **this** and **super**

- **this**
- **this(. . .)**
- **super**
- **super(. . .)**

Software development activities

1. Understand requirements
2. Do the design
3. Implement the design
4. Test the implementation

- Often iterating over the steps as you debug your implementation and/or design

Some concepts in software design (object-oriented design)

- Object composition vs. inheritance
- Composition models a 'has-a' relationship between objects
- Inheritance models an 'is-a' relationship between objects
Software development methods

- Incremental development
- Test-first development

Testing

- Use of `main` in `UseX` for class `X`
- Test-first development

Debugging a Java program

- Using Eclipse
  - Break points
  - Stepping
- Using `System.out.println`

Bits and bytes

- A bit as an abstraction of an electrical wire
- Base 2 numbers and base 10 numbers
- (You will not be tested on this in the final exam)
Character codes

- ASCII code (7~8 bits)
- Unicode (16~21 bits)

Libraries in programming languages

- Packages in Java
  - import ...
  - java.lang.*
  - java.util.*
  - java.io.*
  - javafx.*
  - ...

Constants in Java

- Use final

Equality tests

- ==
- equals
Parsing with **split** in **String**

- Use of regular expressions

**Interfaces**

- Design of a class
- Subtyping

- Examples
  - Comparable with `compareTo`
  - Shape

**Inheritance**

- The Object class
  - `toString` and `equals`
- Models an 'is-a' relationship with inheritance
  - cf. object composition ('has-a' relationship)
- See Person, Employee, Manager example
- protected visibility control

**Inheritance (cont.)**

- Overriding a method definition in a subclass
- Polymorphism
  - Polymorphic object references
  - Method name polymorphism
- Dynamic binding
  - Binding of what to what?
  - Dynamic method name lookup, dynamic method dispatching, late binding, dynamic binding, run time binding (these all mean the same thing)
  - cf. static binding, early-binding, compile-time binding
Types

- Declared type vs. actual type
  - dynamic type (aka, actual type)
    - Used by method dispatching to type check a **value** at run time
  - static type (aka, declared type)
    - Used by compiler as it type check a piece of **code** at compile time
- Type casting and type downcasting
- What do we gain by using inheritance?

Recursion

- Think recursively!
- Recursion as a control mechanism - can use it for looping
- Learn different patterns
- Functional programming
- Need more exercises? See Exercises 18.2, 18.4, 18.5, 18.8, 18.10, 18.11, 18.13, 18.14, etc.

Sorting

- Selection sort: \( O(n^2) \)
- Insertion sort: \( O(n^2) \)
- Quick sort (only saw a demo, so you are not responsible for understanding how it works, but it was interesting to see the difference, particularly in performance): \( O(n \log_2 n) \)
- Sorting an array of primitive type data
- Sorting an array of objects
  - using **equals**
  - using **compareTo** in **Comparable**

Searching

- Linear search: \( O(n) \)
- Binary search: \( O(\log_2 n) \)
  - both iteratively and recursively
  - both using primitive data and objects
- Performance concerns in search
  - How can we improve the searching time?
    - with sorting
    - with binary search
  - Much more on this next semester
Graphical user interface (GUI)

- JavaFX basics
- Event-driven programming
- Animations
- JavaFX UI controls
- Multimedia

For these topics, see the lecture notes and try the examples we went over in class

Programs/Applications

- Histograms
- Sorting
- Searching
- Tic Tac Toe
- Analyzing text features with files
- Modeling baseball teams
- Modeling geometric shapes
- Modeling banks
- Modeling iPhones
- Modeling mailboxes with messages
- Processing student records using file I/O
- Modeling geometric shapes
- Modeling various cars
- ...

That is quite a lot... 

- But yes, we have done all of those...

- If you want to take more CS courses, next ones are
  - CSE 214 (CS II, Data Structures and Advanced Programming)
  - CSE 215 (Foundations of CS)
  - CSE 220 (System Fundamentals I)

What computer scientists do...

- How do we design/build computers
- How do we design/build languages
- Compilers
- Operating systems
- Algorithms
- Database systems
- Artificial intelligence (AI)
- Software engineering
- Distributed systems
- Web systems
- Mobile apps
- Many many more...
What you learned . . .

- Now that the initial difficult stage of learning CS and programming is done, you are ready to have some **fun** with more CS and interesting programming - that is what you will do in CSE 214!
- Still what you learned so far is a very good introduction to computer science!
- After CSE 220 and CSE 214 followed by CSE 219, you will be able to take many upper level courses
- If you want to discuss any of these with me, see me.

Suggestion

- If you are planning on taking CSE 214 or CSE 220 next semester or in the future, . . .
- Review this course material during the break
  - Start with the first lecture and go through them all
  - As you read the lecture notes, do each lab exercise followed by the problem sets, without looking at my sample solutions (Do compare yours with my solutions though)
  - Doing it the second time will be much easier and you will learn a lot!!!
- If you need help as you review it, let me know. I will help.

Finally . . .

- I had lots of fun teaching the course.
- I hope you had some fun, too.
- I am sorry you had to work that hard for this class throughout the semester, but unfortunately that's what it takes to study CS -:(
- Good luck with your finals and have a great break!
- Quoting Joseph Campbell, . . .
  "Follow your bliss!"
- One more thing: please do the course evaluation.