CMPS 12B

Introduction to Data Structures
Winter 2004

Brian F Hanks

Instructor and TAs

- Instructor
  - Brian Hanks
  - BE 189A
  - Office Hours: Thursday 1 – 3 PM
- Teaching Assistants
  - Christopher Harrison
  - Jay Kreps

Texts

- Data Abstraction and Problem Solving with Java: Walls and Mirrors, by Carrano and Prichard
  - Available at Bay Tree and Amazon.com
- C for Java Programmers, by Tomasz Muldner
- On to C, by Charlie McDowell

Course Web Site

- Course web site is at http://www.soe.ucsc.edu/classes/cmps012b/Winter04/
- Can get there from SOE home page, under classes link.

Course Content

- In this course, we will learn about basic data structures and abstract data types
  - Lists, Stacks, Queues, Trees
- We will learn more about Java
  - Exceptions, I/O, interfaces
- We will learn about the C language
- We will learn about Unix and software tools
  - AFS, gmake, RCS, debuggers

What you should know

- Material in Chapters 1 through 6 of Java by Dissection
  - Arrays
  - Classes
    - Instance and class methods, variables
    - Scope and this
    - Class constants
    - Calling methods
  - Passing array and object references to methods
Assignments and Grading

- Programming assignments and written homework
- Final Exam (30%)
  - 40% of course grade
- 2 quizzes (30%)
  - Approximately 1 hour each
- Final Exam (30%)

Assignments and Grading

- You must get at least 50% in each area (programming/homework, quizzes, final) to pass this class
- Getting 50% in each area is not sufficient
  - For example, if you get 51% in each area, you will probably NOT pass

Working Together

- All written homework, the quizzes and the final must be done individually.
- I would like everyone to pair program on the programming assignments
  - You don't have to but...
  - There are lots of benefits associated with pairing
  - You will still have to do the same assignments if you work by yourself

Pairing

- You can tell me who you want to partner with
- You can let me assign you a partner
- You can tell me that you want to work by yourself.

Distributed Pair Programming

- You can use my DPP tool if you want
- I will help you set it up for this class – make an appointment to see me
  - Don't use the setup from last quarter
- You will have to provide your own headset

Lab

- 5 lab sections
  - Monday 11 AM – 1 PM
  - Monday 1 – 3 PM
  - Tuesday Noon – 2 PM
  - Tuesday 8 – 10 PM
  - Wednesday 1 – 3 PM
- TA will be there to assist you
- No lab exercises – time for you to work on your programming assignments
- Start next week
CATS accounts
- You must have a CATS account for this class
- Your programs must run on the CATS machine
  unix.ic.ucsc.edu
- You must read email sent to your CATS address
  – If you don't read it, you may miss something important
  – If you use hotmail or yahoo or whatever, get your CATS email forwarded.

Academic Dishonesty
- On the programming assignments, you may work only with your partner
- On everything else, you must work alone
- Copying other people's programs is strictly forbidden – we have a program that looks for copying, and it is very good.
- Downloading code from the Internet is also forbidden. I can find it just as easily as you!

Academic Dishonesty
- Academic Dishonesty will not be tolerated.
- Results of Academic Dishonesty
  - Failure in this class
  - Possible expulsion from the major
  - Read the academic dishonesty policy on course web site.
  - See me if you have any questions about permissible behavior

Java Class
- A class defines
  - some data
  - some operations (methods) on that data
- A class defines a new type
- Objects or instances of the class are variables of the type.

Abstraction
- Abstraction
  - Hides or ignores details at certain times
  - Easier to deal with an abstraction than to deal with many details at the same time
- Data abstraction
  - focus on what data is, not how it is stored or how operations are performed on it
- Functional abstraction
  - focus on purpose of method without knowing details of how it does it

Abstract Data Type
- What's an abstract data type?
  - Data
  - Operations (methods) that operate on the data
- In Java, we define an ADT with a class
- A class defines a type
- An object or an instance of the class is a variable of the type
Data Structure

- A construct used in a programming language to store a collection of data
  - e.g., array of ints
- A Data Structure does not define operations on the data
- A Data Structure is the underlying programming construct used to implement an ADT
- That’s what this class is about – specifying ADTs and using data structures to implement them.

Information Hiding

- Details of module are hidden
- Details of module are inaccessible from outside the module
  - details are private
  - no other module can tamper with these hidden details
- User of a module does not need to worry about details
- Developer of module does not need to worry about how details will be used

Reading

- Chapter 1:
  - Problem solving and software engineering, up through phase 2: pages 4 – 9
  - Achieving a modular design: pages 16 – 22
- Chapter 3:
  - Abstract Data Types, Specifying an ADT upto but not including Axioms: pages 106 – 122
- Appendix A:
  - String Tokenizer class, Java Exceptions, Text I/O: pages 713 – 724 (or 711 - 722)

Programming Assignment 1

- On web site
- 2 parts
  - Part 1: ADT design – due Tuesday Jan 13
  - Part 2: The program – due Sunday, Jan 18

Abstract Data Type Example

- 24 hour clock
  - What's the data?
  - What are the operations?
- See Clock.java
- Note that the data can be represented in different ways without affecting the users of the ADT.
  - hour, minute as separate fields
  - minutes from midnight as a single field

Preconditions & Postconditions

- Precondition: statement of condition that must exist before invoking the method
- Postcondition: statement of conditions that hold after method executes (assuming preconditions were satisfied)
- When you design your method, specify the pre and post conditions
- Part of the specification of an ADT
Preconditions & Postconditions

- What are the preconditions and postconditions?
  - `double Math.sqrt(double x)`
  - `public Card drawCard()`
  - `public Clock(int hour, int minute)`
- What if preconditions are violated?
  - Method may return a special value, throw an exception, or just not work correctly!
  - We'll talk about this more during the quarter

Lists

- We use lists frequently
  - What might we put on a list?

Lists

- Lists have
  - A first item – called the head
  - A last item – called the tail
- All items except the tail have a unique next item, or successor
- All items except the head have a unique previous item, or predecessor

Lists

- Lists contain items of the same type
  - grocery items
  - phone numbers
  - What can you do with a list?

What can you do with a List?

- Count the number of items
- Add an item
- Remove an item
- Look at an item (retrieve it)
- The items on the list with these operations form an ADT.
  - You must specify the behavior of an ADT's operations

ADT List Operations

- Create an empty list
- Determine whether a list is empty
- Count the number of items in the list
- Add an item to the list at a given position
- Remove the item at a given position from the list
- Remove all the items from the list
- Get the item at a given position in the list
  - See ADTList.txt for pseudocode
How do we use the List?

- Let's use our ADT operations to
  - Create a grocery List
  - Display a List

Documentation

- As your programs get more complex, it becomes more important to document them.
- Comments help you understand what you did last week
  - Good way to see if your design makes sense – long explanatory comments indicate opportunity for better design
- Comments help me understand what you were thinking
- We're going to use Javadoc

Documentation

- Extremely important to document your ADTs
- Need to provide public documentation so that someone else can understand what your ADT does and use it correctly
- Include preconditions and postconditions in your documentation

Javadoc

- Java Tool that generates web page documentation of your programs
- Special form of comment
  - /** comment */
- Special Tags
  - @author
  - @version
  - @see
  - @param
  - @return

Javadoc

- javadoc [arguments] file1.java file2.java ...
- Useful arguments
  - -private
  - -author
  - -windowtitle
- See CardGame example

gmake

- gmake is a program that will automate the building and submission of your programs
- Helps you make sure you submit the correct files
- Makes sure your .class files are up to date
Makefile
- `gmake` uses a file named `Makefile` to tell it what to do
- `Makefile` contains rules
- Rules tell `gmake` how to create a 'target' file from its prerequisites
  - e.g., how to make a class file from a Java sourcefile
- Other rules can be defined to submit your files, clean up your directory, etc.
- See the `Makefile` for the Card Game

No More tio!
- We will not be using the tio package in this class
- Instead, we will use `java.io.*`
- Java provides 3 default stream variables
  - `System.in` of type `InputStream`
  - `System.out` of type `PrintStream`
  - `System.err` of type `PrintStream`
- You should be familiar with `System.out`

Getting user input
- Use an `InputStreamReader` to read from the standard input stream
- Wrap the `InputStreamReader` with a `BufferedReader`
- Use the `readLine` method of `BufferedReader` to get a `String`
- See `InputExample.java`

Getting user input
```java
System.out.print("Please enter your name: ");
BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
String name = stdin.readLine();
System.out.print("Please enter your age: ");
int age = Integer.parseInt( stdin.readLine() );
```