CS606: Fundamental Computer Science II with Java (4 credits)

Fall 2007
Prof. Bergin

Dr. Bergin's Information:
jbergin@pace.edu
http://csis.pace.edu/~bergin
AOL IM: jb605pace

Dr. Bergin's Office Hours:
Office hours: Tuesdays 12:45- 1:15 and 3:30 pm – 5 pm, Thursdays 12:45- 1:15 and 3:30 pm - 5:45 pm, by appointment & online by AOL IM

Prerequisite:

Ability to specify classes in Java, including constructor overloading, private fields, public instance methods, and class variables and methods; ability to manage one and two-dimensional arrays (CS 602 or permission of the chair)

Description:

Derivation including abstract classes, and polymorphism. Interfaces. Exception handling. Building linked data structures including lists and trees. Recursive techniques for managing binary trees and for backtracking. Sorting algorithms. Information storage and retrieval systems and time complexity analysis. Coverage of Java collections framework. Applicable design patterns.

Specificities:

This Fall 2007 version of the course will emphasize the use of the Eclipse environment and test-driven development.

- Eclipse http://www.eclipse.org is an open extensible universal tool platform that provides a set of infrastructures for creating Integrated Development Environments (IDEs), graphical editors, etc. It provides a plug-in facility via which one can add more functionality.
- Test-driven development (TDD) http://www.testdriven.com is emerging as one of the most successful developer productivity enhancing techniques to be recently discovered. The three steps of TDD consist of write test, write code and refactor. We will use junit as the test framework: http://junit.org. It is integrated into Eclipse, by the way, so you don't need a separate download.

Objectives and Outcomes: (There is some flexibility here, depending on student needs)

Objective 1: Learn the principles associated with the construction of class derivation hierarchies. This includes the inheritance of public members from ancestor classes, augmenting an extending class with incremental fields and methods, method overriding, the difference between overriding and overloading, the principles of constructor chaining, polymorphism, and abstract classes. This also includes the uses of super ,final methods and final classes, as well as overriding equals() and toString().

> Outcome: Students will be able to explain and illustrate how polymorphism enables "old code" to manage new objects as well as to describe and exemplify the other programming constructs used in assembling classes into object-oriented software systems.

Objective 2: Learn about the interface construct, and the use of an interface type for implementing callbacks as used by listeners in constructing GUI's.

Outcome: Students will be able to specify an interface, declare classes that implement that interface, write code in accordance with the rules for interface usage, and describe and exemplify the value of this construct.

Objective 3: Acquire the ability for exception handling. This includes the try/catch/finally statement, the throws clause, extending the Exception class for checked exceptions and the RuntimeException class for unchecked exceptions, the throw clause, and the propagation of exceptions.

Outcome: Students will be able to write programs that recover from exceptions thrown by Java as well as throw and recover from exceptions of their own declaration.

Objective 4: Learn about memory management.

Outcome: Students will learn about parameter passing and garbage collection algorithms.

Objective 5: Learn about recursion.

Outcome: Students will be able to desk-check recursive methods by sketching the stack of activations and returns, and students will be able to build recursive methods on natural numbers and other data structures.

Objective 6: Acquire the ability to build and manage linked lists of objects.

Outcome: Students will be able to construct classes for list processing: appending nodes onto the front or the rear of a linked list, deleting nodes from a list, and traversing the list for any action that involves node-by-node activity (e.g. counting the nodes on the list, printing something from each node, or searching).

Objective 7: Extend the understanding of big-O through the study of binary search trees in contrast to linear searches. And the study of n squared versus n logn sorting algorithms.

Outcome: Students will be able to contrast the linear search algorithm with the binary search tree algorithm relative to performance, and their appropriateness for different applications.

Outcome: Students will be able to explain how an n logn sort acquires its advantageous performance.

Objective 8: Acquire the ability to create (parameterized) data structures using the classes provided by the Java Foundations Framework. Students will also learn about the pertinent patterns.

Outcome: Students will be able to build a priority queue with a Vector, LinkedList or an ArrayList.

Outcome: Students will be able to implement hashing.

Outcome: Students will be able to sort arrays with Arrays.sort( ) and where needed the Comparable interface and comparator objects. Students will also be able to sort objects in collection classes.

Objective 9: Acquire familiarity with Test Driven Development. All programming assignments will require appropriate junit tests.

Textbook (required):

Data Structures and the Java Collections Framework, 2ed
William J. Collins
McGraw-Hill, 2005

OPTIONAL:
Java Generics and Collections
Maurice Naftalin, Philip Wadler
O'Reilly, 2006

Your work must be turned in in hard copy at the beginning of the class at which it is due. By turning it in you certify that it is your own work except as noted by you on the work itself. Programming assignments (Java code) must be correctly formatted according to the course style guide.
See: http://csis.pace.edu/~bergin/patterns/codingpatterns.html

Course material:

Course material (syllabus, guidelines, lecture notes, assignments, announcements, readings, etc...) will be available online, in class, by email, and on the wiki. We are unlikely to use Blackboard for anything, but I will set up a wiki site for our use. There will also be a course mailing list so that we may all stay in contact through the semester.

Grading policy:

The primary determinant of your grade will be based on frequent assignments. There will also be a final exam on the required end-of-semester date. The exam is valued at approximately twice an individual assignment. Your grade is the simple average of all work required, including the exam.

Letter grades are assigned in the following way:

A	90-100
B	80-90
C	70-80
F	0-70

Any assignment that receives a grade that the student doesn't like, may be redone. It will be regraded and the new grade substituted. This policy may need to be limited later in the semester and will end two weeks prior to the end.

Participation, Attendance, Late Policy and Academic Dishonesty:

- Participation concerns individual participation in the different class activities.
- You will be expected to participate fully in any team assignments.
- Deadlines are very important and have to be respected. Missed deadlines will results in a decreased grade for the assignment/project. 10% immediate reduction.
- Attendance is required.
- The Seidenberg CSIS Academic Integrity policy will be followed. It will be added that if two similar programming assignments are submitted they will both get a grade of 0. All forms ho dishonesty are actively discouraged. It is one of the few ways to fail the course.

Suggested Course Topics: (May change during the semester)

- Abstract data types
- Exception handling
- Inheritance
- Polymorphism
- Encapsulation
- Interfaces
- Abstract classes
- Recursion
- Linked lists
- Queues
- Trees
- Hashing
- Memory management
- Searching
- Sorting
- Complexity
- Swing and AWT (Abstract Windowing Toolkit)

- Testing and Test Directed Development (TDD)
- Generic collections and the Java Collections Framework