

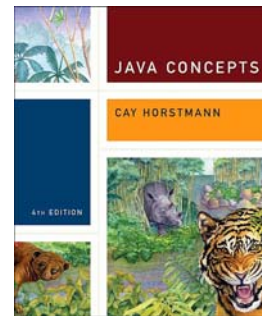
CSC 222: Computer Programming II

Spring 2005

See online syllabus at: www.creighton.edu/~davereed/csc222

Course goals:

- to know and use basic programming tools for object-oriented problem solving (e.g., classes, polymorphism, inheritance, interfaces)
- to appreciate the role of algorithms and data structures in problem solving and software design (e.g., object-oriented design, searching and sorting, recursion, stacks, queues, and linked lists)
- to be able to design and implement a program to model a real-world system, and subsequently analyze its behavior.
- to develop programming skills that can serve as a foundation for further study in computer science



1

221 vs. 222

221: programming in the small

- focused on the design and analysis of small programs
- introduced fundamental programming concepts
 - ✓ variables, assignments & expressions
 - ✓ control structures: if, if-else, while, for
 - ✓ classes: data fields, constructors, methods
 - ✓ parameter passing, local variables
 - ✓ Strings, ArrayLists, arrays
 - ✓ using existing classes: Circle, Die, Card, ...
 - ✓ designing/implementing new classes: DotRace, ESPTester, DeckOfCards

you should be familiar with these concepts (we will do some review in next 2 weeks, but you should review your own notes & text)

222: programming in the medium

- focuses on the design and analysis of more complex programs
- will introduce more advanced programming concepts & techniques
- greater emphasis on problem decomposition, code reuse & modifiability
 - ✓ problem solving: decomposition, testing & debugging strategies
 - ✓ OOD: class design, inheritance, interfaces
 - ✓ algorithms: searching & sorting, recursion, efficiency
 - ✓ data structures: arrays, ArrayLists, stacks, queues, linked lists

2

When problems start to get complex...

...choosing the right algorithm and data structures are important

- e.g., phone book lookup, checkerboard puzzle
- must develop problem-solving approaches (e.g., iteration, recursion)
- be able to identify appropriate data structures (e.g., array, linked list, stack, queue)

...code reuse is important

- designing, implementing, and testing large software projects is HARD
whenever possible, want to utilize existing, debugged code
- reusable code is:
 - clear and readable (well documented, uses meaningful names, no tricks)
 - modular (general, independent routines – test & debug once, then reuse)

3

Object-oriented programming

OOP is the standard approach to software engineering

philosophy: modularity and reuse apply to data as well as functions

- when solving a problem, must identify the objects involved
e.g., banking system: customer, checking account, savings account, ...
- develop a software model of the objects in the form of abstract data types (ADTs)
an ADT is a collection of data items and the associated operations on that data
in Java, ADTs are known as *classes*

OOP stressed ADTs in order to

- hide unnecessary details (programmer doesn't have to know the details of the class in order to use it)
- ensure the integrity of data (programmer can only access public operations)
- allow for reuse and easy modification (can plug classes into different applications)
- *inheritance* and *interfaces* can further facilitate the development of reusable code

4

Short term outlook

next 2 weeks, we will do a crash course review of Java and 221

- cover basic concepts, review program examples
- also, introduce new Java features, a new Java IDE – Eclipse

you should review 221-level material ON YOUR OWN

- review your book & notes from 221
or, better yet,
- read Ch. 1-4, 6-9 of the text for this class

note: Java 1.5 (a.k.a. Java 5.0) makes substantial changes over Java 1.4

- for our purposes, these changes are EXTREMELY GOOD!
- but, be prepared for some differences over last semester

- we will discuss how you can download all software tools (FOR FREE!) next week