

CSC 222: Object-Oriented Programming

Spring 2017

See online syllabus at: dave-reed.com/csc222

Course goals:

- To know and use basic Java programming constructs 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, lists, files, searching and sorting).
- To be able to design and implement a Java 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

Assumed background

technically, CSC 221 is a prerequisite for this course

- what is really needed is basic programming & problem-solving experience

- ✓ variables: data types, assignments, expressions
- ✓ control structures: if, if-else, while, for
- ✓ functions: parameters, return, libraries
- ✓ data structures: strings, lists, files

- early on, I will map Java constructs back to their corresponding Python
- if you learned a different language, will need to make your own connection

as an intro, 221 focused on programming-in-the-small

- simple problems; could be solved in 1-3 functions; few design choices

this class extends to programming-in-the-medium

- and lays the groundwork for programming-in-the-large by emphasizing the *object-oriented approach* to software design

2

When problems start to get complex...

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

- e.g., phone book lookup, checkerboard puzzle, word ladders
- must develop problem-solving approaches (e.g., iteration, recursion)
- be able to identify appropriate data structures (e.g., array, ArrayList, 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 stresses 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

Getting started

recall: you got a sneak peek at OO at the end of 221

- e.g., Die, DeckOfCards, RowOfCards

we will start next week with the philosophy of OO

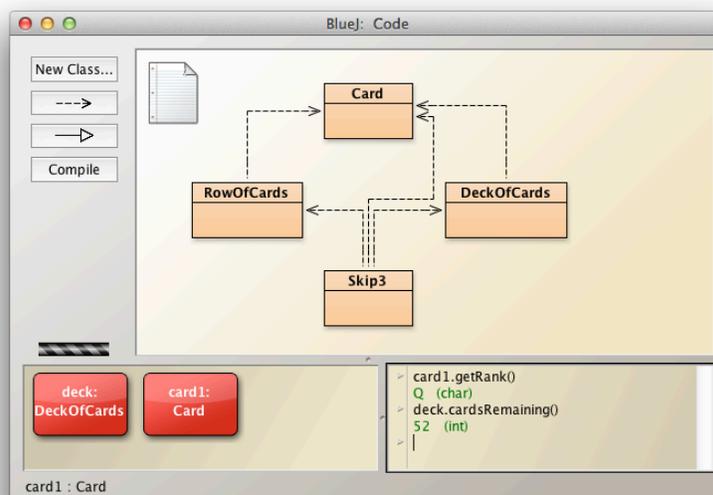
- concepts of class & object
- how designing classes and instantiating objects leads to modular, reusable code

we will be using Java (compiler/interpreter) & BlueJ (IDE)

- both are on the CD that comes with the book
- can also be downloaded for free from the Web
 - BlueJ (3.1) from www.bluej.org
- BlueJ is a simple, visual environment; designed for beginners to OO approach

5

BlueJ screenshot



6