

Lecture 1: Overview & Java

CS 62

Spring 2015

Kim Bruce & America Chambers

TAs: Christina Tong, Jennifer Nguyen, Laura Pandori,
Maria Martinez, Sean Zhu & Michael Diamond
Course web page: <http://www.cs.pomona.edu/classes/cso62>

1

Why CS 62?

- How to implement algorithms and data structures in Java & C++.
- How to design large programs (in object-oriented style) so that it is easy to modify them.
- How to analyze complexity of alternative implementations of problems.

2

Sample Problems

- Find the shortest path from Claremont to Chicago on interstate system
 - (and do it efficiently)
- Schedule final exams so there are no conflicts.
- Design and implement a scientific calculator.
- Design and implement a simulator that lets you study traffic flow in a city or airport.
- Design parallel algorithms to run on multicore computers

3

Your Responsibilities

- Skim reading in advance of lecture.
- After lecture, review lecture notes, sample code, and text until well understood.
- Come to labs prepared.
- Don't remain confused. Ask faculty or TAs.
- Follow academic integrity guidelines.

4

Assignments

- Lab work:
 - Learn tools & prep work for weekly assignments
 - Lab attendance mandatory!
- Weekly assignment is separate
 - Programs generally due on Sunday nights.
 - See late policy on syllabus: 3rd% per day late
- Daily homework
 - Not collected, but often on **regular Friday quizzes**
 - *No quiz this Friday.*

5

Texts

- Java Structures, $\sqrt{7}$ edition, by Duane Bailey
 - available on-line for free
- C++ for Java Programmers, by Mark Allen Weiss
 - highly recommended for last 1/3 of course
 - Not at book store: buy on-line

6

Slides

- Will generally be available before class
 - with code, where applicable
- Designed for class presentation, not for complete notes
- Will need to take notes (perhaps on slides)
- No laptops or other electronics open in class
 - If that is problem, come see me.

7

Prerequisite

- One of:
 - CS 51 at Pomona or CMC (*not CS 5 from HMC!*)
 - AP CS A exam with score of 4 or 5
 - Fluent in Java and object-oriented programming & permission of instructors
- Come see one of faculty if any questions
- Assume comfortable with classes & objects, recursion, multi-dimensional arrays, etc. in Java

8

See syllabus for other
important information!

9

Object-Oriented Design

- Objects are building blocks.
- Program is collection of interacting objects.
- Objects cooperate to compute solution.
- Objects communicate via sending messages.

10

Objects

- Model physical and conceptual world, as well as processes.
- Objects have:
 - Properties, e.g. color, size, manufacturer, ...
 - Capabilities, e.g. drive, stop, admit passenger
- Objects responsible for knowing how to perform actions.
 - Commands: change state
 - Queries: response based on properties

11

More Objects

- Properties typically implemented as “fields” or “instance variables”
 - Affect how object reacts to messages
 - Can be
 - Attributes, e.g., color
 - Components, e.g., doors
 - Associations, e.g., driver
- Capabilities as “methods”
 - Invoked by sending messages

12

Quick Java Review

13

Classes & Interfaces

- Interfaces
 - Provide info on publicly available methods of objects
- Classes are templates for objects
 - Constructors generate new distinct objects
 - `new Car("Toyota",...)`
 - Specify all fields and methods – public and non-public
 - May be used as basis for more refined classes via inheritance

14

All Classes Specialize “Object”

- Object class has methods:
 - `public boolean equals(Object other)`
 - Default behavior returns true only if same object
 - `public String toString()`
 - Returns string representation of object - default is hexadecimal
 - Typically want to override to be more useful
 - `public int hashCode()`
 - Unique identifier defined s.t. if `a.equals(b)` then `a`, `b` have same `hashCode`.
 - Cover in later chapter of text.

15

Enum Types

- Example:
 - `enum Suit {CLUBS, DIAMONDS, HEARTS, SPADES};`
- Operations:
 - `int compareTo(Suit other)`
 - `String toString()`
 - `int ordinal()` *starts with 0, not 1*
 - `static Suit valueOf(String name)`
 - `static Suit[] values()` *returns array of all values*

16

Card Deck Examples

- CardInterface -- interface
 - AbsCard
 - abstract class, implements CardInterface
 - Card extends AbsCard
 - OtherCard extends AbsCard
- } *alternate implementations*
- Deck
 - Class holding array of cards

17

Java Keywords

- *Abstract* class -- can't be instantiated
 - usually some methods missing
- Information hiding qualifiers:
 - *public*
 - *private*
 - *protected*
- *Static* -- copy associated with class, not objects
- *Final* -- only assigned to once
 - in its declaration or constructor

18

Interfaces & Inheritance

- Class implements interface if supports all methods defined in interface
- Interface can extend another by adding methods
 - If A extends B and x has type A, then also has type B
- One class can extend another
 - inherits fields and methods
 - can override existing methods, add new ones
- instanceof & casts
 - Ex: in Ratio class later

19