Lecture 1: Overview & Intro to Java

CS 62
Spring 2018
Alexandra Papoutsaki & William Devanny

http://www.cs.pomona.edu/classes/cs062
Welcome!
Who we are:

Alexandra Papoutsaki  
David Ahia  
Kayla Cummings

William Devanny  
Alia Buckner  
Gloria Liou  
Matthew Paik

Arianna Chen  
Emily Chen  
Sarp Misoglu
Index Cards

• Write down the questions you have as you go
• Question answered? Strikethrough
• I will collect the feedback at the end of class
Why take CS62?

• How to implement algorithms and data structures in Java.

• 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.
Sample Problems

• Find the shortest path from Claremont to Chicago on interstate system (and do it efficiently).
Sample Problems

• Schedule final exams so there are no conflicts.
Sample Problems

• Design and implement a scientific calculator.
Sample Problems

• Design and implement a simulator that lets you study traffic flow in a city or airport.
Your responsibilities

• Skim reading in advance of lecture.
• After lectures, review notes and study examples carefully until you understand them.
• Come to labs prepared.
• Don’t remain confused. Faculty and TAs are here to help.
• Follow academic integrity guidelines
Assignments

• Lab work:
  • Learn tools and prepare work for weekly assignments.
  • Lab attendance is mandatory! *No lab today!!!*

• Weekly assignment is separate
  • Programs generally are due on Sunday nights.
  • See late policy on syllabus. $3^n\%$ penalty per day late.

• Daily homework
  • Not collected, but often on **regular Friday quizzes**.
  • *No quiz this Friday!*
Text

• Java Structures, $\sqrt{7}$ edition, by Duane Bailey
  • available online for free
    • http://www.cs.williams.edu/~bailey/JavaStructures/Book.html

• Various online resources
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 you have a disability affecting this, come see me.
Prerequisite

• Officially, CS 52 at Pomona

• Knowledge of Java equivalent to CS 51 at Pomona or CMC or the AP Test with 4 or 5.
  • *not CS 5 from HMC or CS 30 from Pomona!*

• Come see one of faculty if having any questions

• Assume comfortable with classes & objects, recursion, multi-dimensional arrays, etc. in Java
Heavy Workload

• students spend average of 8+ hours outside of class.
• … but not “weeder”
• Must both learn practical (programming) skills and more theoretical analysis skills
  • Learn about tools to become better programmer
  • Be ready to answer “interview questions”
Grading Policy

• We drop the two quizzes with the lowest grade
  • Keep this option for real emergencies and unpredictable events

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly Programming Assignments</td>
<td></td>
<td>35%</td>
</tr>
<tr>
<td>Exams: Total:</td>
<td></td>
<td>55%</td>
</tr>
<tr>
<td>Midterms: 15% each</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Exam: 25%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In-lab exercises and quizzes</td>
<td></td>
<td>10%</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>100%</strong></td>
<td></td>
</tr>
</tbody>
</table>
See online syllabus for other important information!

Using Github does not mean you can make your assignments publicly available

http://www.cs.pomona.edu/classes/cs062
Object-Oriented Design

• Objects are building blocks.

• Programs are collections of interacting objects.

• Objects cooperate to compute solutions or complete tasks.

• Objects communicate via sending messages.
The ticketing system in a movie theatre

Objects

• Objects can model objects from world:
  • Physical things
    • e.g., car, student, card, deck of cards
  • Concepts
    • e.g., meeting, date
  • Processes
    • e.g., sorting, simulations
More objects

• Objects have:
  • Properties, e.g., color, model, manufacturer
  • Capabilities, e.g., drive, stop, admit passenger

• Objects are responsible for knowing how to perform actions.
  • Commands: change object’s properties, (e.g., set speed)
  • Queries: respond based on object’s properties (e.g., how fast?)
Even more objects

• Properties typically implemented as “fields” or “instance variables”
  • Affect how objects reacts to messages
  • Can be:
    • Attributes, e.g., color
    • Components, e.g., door
    • Associations, e.g., driver

• Capabilities as “methods”
  • Invoked by sending messages
Quick Java Review
Primitive Data Types

- `char`, `int`, `byte`, `short`, `long`, `double`, `float`, `boolean`

- Use a small amount of memory to represent a single item of data

- All data of same primitive data type use the same amount of memory

- Cannot be used to instantiate type variables, that is no `new`

- Have corresponding object “wrapper” types:
  - `Integer`, `Double`, `Float`, `Boolean`, etc.
Objects

• Any data type that is not a primitive
• You already know **String**
  • Thousands more coming with Java by default
• You can create your own with the **new** keyword
• Contain data and methods
• Respond to messages
Classes

• Classes are templates for objects
  • The data type of that kind of object

• Constructor
  • Have the same name with the class
  • 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
    • `class Car extends Vehicle`
## Access modifiers

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>protected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Default (nothing!)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>private</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
Instance Variables

- or member variables or fields
- Declared in a class, but outside any method, constructor or block
- Each object has its own copy of the variable!
- Invoked as: `myObject.variableName`
Static Variables

• or class variables
• static means constant, i.e. it will be constant for all instances of the class
• cannot be defined in method body
• Invoked as: `myClass.variableName`
Local Variables

• Declared in method, constructor or block
• Destroyed after the execution of the method
• No access modifier

• What about these variables?

```java
public class Student {
    private String name;
    private int id;

    public static int numberOfStudents = 0;
}
```