Lecture 1: Overview & Intro to Java

CS 62
Fall 2018
Alexandra Papoutsaki & William Devanny

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

Alexandra Papoutsaki
Sam Gearou
James Marshall
Matthew Paik
Cecelia Sanborn
William Devanny
Berto Gonzalez
Sean MacPherson
Who are you?

• Name
• College
• Year

• Sign up sheet for the first two lectures
Why take CS62?

• It’s required to major/minor in Computer Science

You will learn:

• 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 and maintain them.
• How to analyze the complexity of alternative implementations of problems.

• It can help you with interview questions
Sample Problems

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

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

• Design and implement a scientific calculator.

web2.0calc
Your responsibilities

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

• Wednesday 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 midnight.
  • 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!*
• Java Structures, $\sqrt{7}$ edition, by Duane Bailey
  • available online for free
  • Hard copies in lab (in the departmental library)
  • http://www.cs.williams.edu/~bailey/JavaStructures/Book.html

• Various online resources that will be linked in website
Slides

• Will generally be available the morning of 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 an accommodation for this, come see me
Prerequisites

• Officially, CS 52 or 54 at Pomona
• Knowledge of Java equivalent to CS 51 at Pomona or the AP Test with 4 or 5.
  • not CS 5 from HMC or CS 30 from Pomona!
• We assume you are comfortable with classes & objects, recursion, multi-dimensional arrays, etc. in Java
• Come see me if you have any questions
Heavy Workload

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

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly Programming Assignments</td>
<td>35%</td>
</tr>
<tr>
<td>Exams:</td>
<td>Total: 55%</td>
</tr>
<tr>
<td>Midterms: 15% each</td>
<td></td>
</tr>
<tr>
<td>Final Exam: 25%</td>
<td></td>
</tr>
<tr>
<td>In-lab exercises</td>
<td>5%</td>
</tr>
<tr>
<td>Quizzes</td>
<td>5%</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

- We drop the quiz with the lowest grade
  - Keep this option for *real* emergencies and unpredictable events
Communication channels

• **Piazza**
  • I will send an invitation to your email after class
  • Anonymous posts that can be seen by your classmates: Very useful for questions that everyone might have
  • Preferred over email. The whole staff will be monitoring it so you’re more likely to get a quick response

• **Github**
  • Will be used for submitting assignments
  • Instructions will be posted on piazza
  • *Using Github does not mean you can make your assignments publicly available*
Index Cards

• Write down the questions you have as you go
• Question answered? Strikethrough
• I will collect the feedback at the end of class and will provide answers on piazza or at the beginning of next class
Office Hours/Mentor Sessions

• Papoutsaki: M 11am-12pm, F 1-4pm (Edmunds 222)

• Devanny: W 4-6pm, Th 2:30-4pm (Edmunds 230)

• Mentor sessions starting next week (Edmunds 2nd floor labs):
  • Th 7-9pm
  • F 3-5pm
  • Sa 8-11am
  • Sun 8-10pm

• Any changes will be announced on Piazza
See online syllabus for other important information!

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

• Objects are building blocks of a program.
• 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:
  • States or Properties, e.g., color, model, manufacturer
  • Behaviors or 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
Java Primer

Types, Classes, Objects

Some slides based on Goodrich and Tamassia, 2014
Java

• Java applications are compiled to byte-code

• Can run on any Java Virtual Machine (JVM)
  • Architecture independent

• Source code is saved in a file with .java suffix

• Byte-code is stored in a file named with .class suffix by the compiler
An Example: `Universe.java`

```java
public class Universe {
    public static void main(String[] args) {
        System.out.println("Hello Universe!");
    }
}
```
An Example: Universe.java

```java
public class Universe {
    public static void main(String[] args) {
        System.out.println("Hello Universe!");
    }
}
```
Components of a Java program

• Executable statements are placed in methods

• The static method named main is the first method to be executed

• A program block is defined within curly brackets (i.e. {})
The name of a class, method, or variable in Java

Can be any string of characters as long as it begins with a letter and consists of letters

<table>
<thead>
<tr>
<th>Reserved Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
</tr>
<tr>
<td>assert</td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>break</td>
</tr>
<tr>
<td>byte</td>
</tr>
<tr>
<td>case</td>
</tr>
<tr>
<td>catch</td>
</tr>
<tr>
<td>char</td>
</tr>
<tr>
<td>class</td>
</tr>
<tr>
<td>const</td>
</tr>
<tr>
<td>continue</td>
</tr>
</tbody>
</table>
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.
# Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean</code></td>
<td>a boolean value: true or false</td>
</tr>
<tr>
<td><code>char</code></td>
<td>16-bit Unicode character</td>
</tr>
<tr>
<td><code>byte</code></td>
<td>8-bit signed two’s complement integer</td>
</tr>
<tr>
<td><code>short</code></td>
<td>16-bit signed two’s complement integer</td>
</tr>
<tr>
<td><code>int</code></td>
<td>32-bit signed two’s complement integer</td>
</tr>
<tr>
<td><code>long</code></td>
<td>64-bit signed two’s complement integer</td>
</tr>
<tr>
<td><code>float</code></td>
<td>32-bit floating-point number (IEEE 754-1985)</td>
</tr>
<tr>
<td><code>double</code></td>
<td>64-bit floating-point number (IEEE 754-1985)</td>
</tr>
</tbody>
</table>

Example code:

```java
boolean flag = true;
boolean verbose, debug;
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257;
long l = 890L;
float pi = 3.1416F;
double e = 2.71828, a = 6.022e23;
```
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 fields (can be a primitive or class type) and methods

• Respond to messages
Classes

• Classes are templates/blueprints 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 (friendly)</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>
Data encapsulation

Hiding the internal state of an object and requiring all interaction to be performed through its methods.

• Can be achieved by declaring all the fields in the class as private and writing public methods to set and get the values of variables.
Questions

• Real-world objects contain ___ and ___.
• A software object's state is stored in ___.
• A software object's behavior is exposed through ___.
• Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data ___.
• A blueprint for a software object is called a ___.