A Note for those Stumbling on these Slides (Hi!)

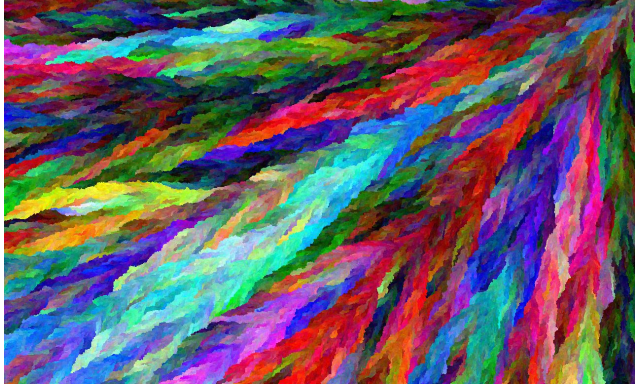
These lecture slides are not intended as written reference materials.

- Just reading them probably won't be very educational.

Best used in combination with webcasts and source code references.

- See (Video) and (Code) links under each lecture: http://datastructur.es

Note: I'm slowly modernizing my 61B slides. You'll see two different styles throughout the semester. This one is the new style, co-developed with Prof. Lisa Yan when we taught DS100.



CS61B, LECTURE 1, SPRING 2023

Introduction to 61B

61B Topical Overview

Lecture 1, CS61B, Spring 2023

- **Welcome!**
  - **61B Topical Overview**
  - 61B Logistics
- Our First Java Programs
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

61B Overview

What is 61B about?

- Writing code that runs efficiently.
  - Good algorithms.
  - Good data structures.
- Writing code efficiently.
  - Designing, building, testing, and debugging large programs.
  - Use of programming tools.
    - git, IntelliJ, JUnit, and various command line tools.
  - Java (not the focus of the course!)

Assumes solid foundation in programming fundamentals, including:

- Object oriented programming, recursion, lists, and trees.

Why 61B?

In 61B covers:

- How to build industrial strength code. Some highlights:
  - DIY software testing.
  - Using industrial strength tools (git, IntelliJ, the IntelliJ debugger).
  - Selection of appropriate data structures.
  - Software and class design.
  - Teamwork (at the end).
- The most popular topics for job interview questions in SWE.
  - Examples: Hash tables, binary search trees, quick sort, graphs, Dijkstra's algorithm.
- Some really cool math. Examples:
  - Asymptotic analysis.
  - Resizing arrays.
  - The isometry between self-balancing 2-3 trees and self-balancing red black trees.
  - Graph theory.
  - Compression theory.

Question for You

What do you hope / expect to learn from this class? Why are you taking it?

-

Who Are You?

(Note to Zoom Viewers, in the future I'll be using Google Forms when asking questions out loud, but for today, I'm just asking the in person folks for a show of hands).

Who Are We?

Instructors: Josh Hug hug@cs.berkeley.edu 779 Soda
Justin Yokota jyokota@berkeley.edu

GSIs:

- Full time: Allen Gu, Jedidiah Tsang, Shreyas Kompalli, SreeVidya Ganga, Alex Schedel, Angelina Songco, Anish Bajaj, Anton Zabreyko, Aram Kazorian, Eric Che, Crystal Wang, Dominic Conricode, Eddie Park, Ergun Acikoz, Lucy Lu, Meshan Khosla, Noah Adhikari, Sherry Fan
- Part time: Adit Shah, Alexander Lew, Ali Khani, Austin George, Circle Chen, Dhruti Pandya, Dylan Hamuy, Elisa Kim, Emily Su, Hailey Park, Jasmine Lin, Jennifer Prince, Kenneth Wang, Kyle Zhang, Laksith Prabu, Max Ye, Sahityasree Subramanian, cheese, Royce Ren, Shirley Chen, Shreyas Kallingal, Stella Kaval, Yaofu Zuo

Tutors:

- Allison Hong, Angel Aldaco, Ashley Kao, Ashley Ye, Ayati Sharma, Carl Ji, David Yang, Deepak Ragu, Dhruv Ahuja, Elana Ho, Erik Kizior, Kevin Sheng, Lawrence Shieh, Manke Luo, Michael Wu, Olivia Huang, Omar Yu, Pradyun Kumar, Ronald Wang, Surbhi Jain, Taylor Hodan, Thomas Lee, Vanessa Teo, Xifeng Li (plus approximately 20 more incoming)

We will also have a large number of academic interns (300 applications so far!)

61B Logistics

Lecture 1, CS61B, Spring 2023

- **Welcome!**
  - 61B Topical Overview
  - **61B Logistics**
- Our First Java Programs
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Course Components

Lectures provide you with an introduction and a foundation.

You'll learn most of what you learn in the class by:

- Programming (labs, hws, projects, discussion section).
- Solving interesting problems (study guides, HW3, HW4 old exam problems, discussion section).

The Manner in Which Learning Occurs (TMWLO)

Learning occurs by doing.

Evaluation

Four types of points in this class:

- Low effort, everyone should get them: **Weekly Surveys, Course Evaluations**
  - Median score is 100%
- High effort, everyone should get them: **HW, Project, Lab**
  - Median score is 100%
- High effort, not everyone gets them: **Exams**
  - Median score is 60%
  - Final exam score can replace midterms if you have a bad midterm (or two).
  - Dates: February 9, March 16, May 9th. Alternates possible if truly necessary.
- Pacing points: **Attending Discussion, Lab, and keeping up with Lecture**
  - Small amount of extra credit for keeping up with class.
  - Will not increase your score beyond 30,000 points.
    - Example: You have 29,772 points and earn 800 pacing points, you get 30,000 points.

Full details around point distributions, letter grade assignments, grade replacement, etc. are on the website.

Lateness Policies

TBD

Class Phase

This class is divided into three phases:

- Phase 1 (weeks 1 - 4): Intro to Java and Data Structures.
  - All coding work is solo.
  - Moves VERY fast.
  - **HW0 (intro to Java) due Friday (in two days!)**
- Phase 2 (weeks 5 - 10): Data Structures:
  - All coding work is solo except 2nd half of final project.
  - Moves moderately fast.
- Phase 3 (weeks 12 - 14): Algorithms.
  - Coding work is entirely dedicated to final project, done in pairs.
  - Slower pace.

Phase 1 Overview

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- **Our First Java Programs**
  - **Phase 1 Overview**
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Overview of Phase 1

We're now beginning Phase 1: Programming Intensive Introduction to Java.

- Weeks 1-4, Lectures 1 - 11.
- Two homeworks:
  - **HW0** (**Sep 2**): Browser-based **intro to Java** HW.
  - **HW1** (**Sep 9**): Being a good classmate.
- Four labs to introduce you to various tools (starting this week).
  - **Lab 1**: **IntelliJ** and Git
  - **Lab 2**: The IntelliJ Debugger
  - **Lab 3**: Unit Testing
  - **Lab 4**: More Advanced Git, Debugging
- Two projects, both must be done independently:
  - **Project 0** (due ??): Data Structure Utilization.
  - **Project 1A** (due ??): Data Structure Construction.
  - **Project 1B** (due ??): More Data Structure Construction.
- Midterm (covering lectures 1 - ?): September 14 at 7 PM.

Due dates in **gold**.

Hello World

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- **Our First Java Programs**
  - Phase 1 Overview
  - **Hello World**
  - Hello Numbers
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Intro to Java

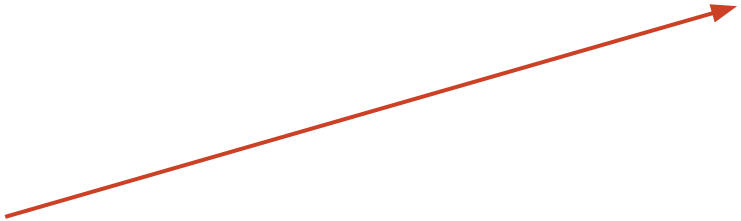
Let's try writing some simple Java programs.

- First I'll write them in Python (~99% of you have seen Python).
- Then I'll write the equivalent Java program.

If you've only ever written code in MATLAB, this will be a little harder for you, but still comprehensible.

1. Intro, Hello World Java

[vid1] [vid2] [slides] [guide]



(See video or code linked in the guide)

Hello Numbers

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- **Our First Java Programs**
  - Phase 1 Overview
  - Hello World
  - **Hello Numbers**
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Larger

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- **Our First Java Programs**
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - **Larger**
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Reflections on Java

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- **Our First Java Programs**
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - **Reflections on Java**
- Workflow
  - Compilation
  - IntelliJ
- HW0: Due Friday!

Java and Object Orientation

Java is an object oriented language with strict requirements:

- Every Java file must contain a class declaration*.
- **All code** lives inside a class*, even helper functions, global constants, etc.
- To run a Java program, you typically define a main method using `public static void main(String[] args)`

*: This is not completely true, e.g. we can also declare “interfaces” in .java files that may contain code. We’ll cover these soon.

Java and Static Typing

Java is statically typed!

- All variables, parameters, and methods must have a declared type.
- That type can never change.
- Expressions also have a type, e.g. "`larger(5, 10) + 3`" has type int.
- The compiler checks that all the types in your program are compatible **before the program ever runs**!
  - e.g. `String x = larger(5, 10) + 3` will fail to compile.
  - This is unlike a language like Python, where type checks are performed DURING execution.

Your Reflections on Static Typing

What are some good things about static typing?

What are some bad things about static typing?

My Reflections on Static Typing

The Good:

- Catches certain types of errors, making it easier on the programmer to debug their code.
- Type errors can (almost) never occur on end user's computer.
- Makes it easier to read and reason about code.
- Code can run more efficiently, e.g. no need to do expensive runtime type checks.

The Bad:

- Code is more verbose.
- Code is less general, e.g. had to have two different larger functions earlier.

Compilation

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- Our First Java Programs
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- **Workflow**
  - **Compilation**
  - IntelliJ
- HW0: Due Friday!

Compilation vs. Interpretation

In Java, compilation and interpretation are two separate steps.

Why make a class file at all?

- .class file has been type checked. Distributed code is safer.
- .class files are 'simpler' for machine to execute. Distributed code is faster.
- Minor benefit: Protects your intellectual property. No need to give out source.

You can learn more about all this in 61C and particularly 164.

Note: .class files are easily reversible into similar looking Java files.

IntelliJ

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- Our First Java Programs
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- **Workflow**
  - Compilation
  - **IntelliJ**
- HW0: Due Friday!

There are many different workflows for writing programs.

- **Text editor** + **command line**: (CS61A, CS88). We just did this.
  - **Text editor**: Writing your code.
  - **Command line**: Running your code.
- **Jupyter Notebooks**: (Data 8)
  - Write and run code in the same environment.
- **Integrated Development Environment (IDE)**: (E7, 61B)
  - Write code and run code in the same environment.
  - Tons of additional features like a debugger, code autocomplete, continuous syntax checking, decompilation (from .class to .java), etc.

Let's see what our programs look like in the IDE for our course.

(screenshot)

Admonition

Our expectation is that everyone in this class is using IntelliJ.

- It is not strictly required, but staff will provide **no support** for other tools or workflows.

IntelliJ

Lecture 1, CS61B, Spring 2023

- Welcome!
  - 61B Topical Overview
  - 61B Logistics
- Our First Java Programs
  - Phase 1 Overview
  - Hello World
  - Hello Numbers
  - Larger
  - Reflections on Java
- Workflow
  - Compilation
  - IntelliJ
- **HW0: Due Friday!**

Time to Go Learn Java Basics!

I am not going to spend time in this class covering for loops, while loops, etc. in Java!

- You've seen this all before in some other language.

HW0A is out, and is due this Friday!

- We show you how to translate various Python constructs into Java, you write some short programs.
  - If you're one of the 1.5% that haven't seen Python before, you'll be fine.
  - If you're one of the 0.3% who have only used MATLAB and find yourself stuck on this HW, post on Ed and we'll reach out and work with you one-on-one.
- Not required to use IntelliJ for HW0A since IntelliJ setup isn't until lab 1.

If you can, start lab 1 early! Most of it is just downloading and installing software.