Scope, Pass-by-Value, Static

Exam-Level 01

CS 61B Spring 2024

Announcements

- Welcome to CS 61B!
- Please read our Ed guidelines before you post to make sure everything follows the rules
- Pre-Semester Survey: due Monday 1/22 at 11:59pm PT
- Homework 0B: due Monday 1/22 at 11:59pm PT
- Project 0: due Monday 1/29 at 11:59pm PT

Meet Your TA!

Hi! I’m Aniruth.

- He series
- aniruth.n@berkeley.edu
- EECS and Business
- Senior
- Took a gap year!
- First taught 61B Summer 2021

Fun facts:

- Cook a lot (probably too much, food mad expensive)
- Used to live in the living room
- Former 61Baller

Logistics

This is an *exam-prep* level discussion. We talk about exam problems, always starting with:

What is the question asking?

I recommend trying either a bridge or regular discussion in combination if you find yourself wanting more review of concepts (if your schedule permits).

When applicable, I will be referencing real-world examples and topics, especially relevant for jobs/interviews in the software industry (a bit more career-oriented).

Everything (my work through the worksheet and these slides) will be posted up at aniruthn.com/teaching/sp24-cs61b. Resources from past semesters are also available (Summer 2021 had slides that I made personally).

Give me feedback on what works and what doesn't in the weekly surveys! There'll be a section where you can do so based on what discussion you attend.

Content Review

Review: Pointers, Types

On chalkboard

Quick Java Basics

```
public class Hello {
    public static void main(String[] args) {
        System.out.println(“Hello world!”);
    }
}
```

- In Java, pretty much everything is defined in a class
- Type declarations: Java is statically typed, so we have to tell the computer what type of value every variable holds and what every function returns (ie. `int, void`)
- Don’t forget the brackets and semicolons!

Structure of a Class

```
public class CS61BStudent { // Class Declaration
    public int idNumber; // Instance Variables
    public int grade;
    public static String instructor = “Hug”; // Class (Static) Variables
    public CS61BStudent (int id) { // Constructor
        this.idNumber = id; // this refers to the instance of the CS61BStudent we are in
        this.grade = 100;
    }

    public boolean watchLecture() { // Instance Method
        ...
    }
    public static String getInstructor() { // Class (Static) Method
        ...
    }
}
```

Instantiating Classes

```
public class CS61BLauncher {
    public static void main(String[] args) {
        CS61BStudent studentOne; // Declare a new variable of class CS61BStudent
        studentOne = new CS61BStudent(32259); // Instantiate and assign to our new instance
        CS61BStudent studentTwo = new CS61BStudent(19234); // Both at once

        studentOne.watchLecture(); // Instance methods are called on instance

        CS61BStudent.getInstructor(); // Static methods can be called on the class OR the
                                        instance

        CS61BStudent.watchLecture(); // Fails. Which student is watching lecture?
        studentOne.getInstructor(); // Works, though is seen as bad practice.
    }
}
```

Overview: Static vs. Instance

Static variables and functions belong to the whole class.
Example: Every 61B Student shares the same instructor, and if the instructor were to change it would change for everyone.

Instance variables and functions belong to each individual instance.
Example: Each 61B Student has their own ID number, and changing a student's ID number doesn't change anything for any other student.

<u>Check for understanding: can you reference instance variables in static methods? Can you reference static variables in instance methods?</u>

*Don't worry if you don't fully understand the difference right now! We'll talk more about this in future discussions

Aside: Analogies

Analogies are a powerful tool to deepen your understanding and to demonstrate it.

They also build intuition.

These concepts (and many others we will discuss) are language-agnostic; even higher-level languages like Python have classes, static/instance, and many of the data structures we cover.

You should ideally have a good analogy for static/instance that is clear and makes sense; one I like is the blueprint/instance one.

Worksheet

CS 61B
Fall 2023

Scope, Pass-by-Value, Static

Exam-Level 01: January 22, 2024

1 Quik Maths

(a) Fill in the blanks in the main method below. (Fall '16, MT1)

```
public class QuikMaths {
    public static void multiplyBy3(int[] A) {
        for (int i = 0; i < A.length; i += 1) {
            int x = A[i];
            x = x * 3;
        }       ↑ x is a local variable; this method does not change A
    }

    public static void multiplyBy2(int[] A) {
        int[] B = A;
        for (int i = 0; i < B.length; i+= 1) {
            B[i] *= 2;                             arr → 2 3 3 4
        }
    }
                                      6      7
    public static void swap(int A, int B) {
        int temp = B;     temp → 7
        B = A;             B → 6 ← change local copies of parameters
        A = temp;          A → 7 ↙ int is a primitive type
    }

    public static void main(String[] args) {
        int[] arr = new int[]{2, 3, 3, 4};
        multiplyBy3(arr); // Value of arr: {_2, 3, 3, 4_______________}

        arr = new int[]{2, 3, 3, 4};
        multiplyBy2(arr); // Value of arr: {_4, 6, 6, 8_______________}

        int a = 6;
        int b = 7;
        swap(a, b); // Value of a: ___6___  Value of b: ___7___
    }
}
```

(b) Now take a look at the code below. How could we write 'swap' to perform swapping primitive variables in a function? Be sure to use the `IntWrapper` class below.

Order of Completion:
6
1 2
3 4 5
7 8 9

```
class IntWrapper {
    int x;
    public IntWrapper(int value) {
        x = value;
    }
}

public class SwapPrimitives {
    public static void main(String[] args) {
        int a = 6;
        int b = 7;

 1      IntWrapper aWrapper = new IntWrapper(a)_____;

 2      IntWrapper bWrapper = new IntWrapper(b)_____;

 3      swap(_aWrapper___, _bWrapper__);

 4      a = _aWrapper.x_______________________;

 5      b = _bWrapper.x_______________________;

    }

6   public static void swap(IntWrapper A_____, IntWrapper B_____) {

 7      int temp = B.x__________________________;  Can't just swap the underlying pointers

 8      B.x = A.x_______________________________;

 9      A.x = temp.x____________________________;
    }

}
```

2 Static Books

Suppose we have the following `Book` and `Library` classes.

```
class Book {
    public String title;
    public Library library;
    public static Book last = null;

    public Book(String name) {
        title = name;
        last = this;
        library = null;
    }

    public static String lastBookTitle() {
        return last.title;← 4
    }
    public String getTitle() {
        return title;
    }
}
```

```
class Library {
    public Book[] books;
    public int index;
    public static int totalBooks = 0;

    public Library(int size) {
        books = new Book[size];
        index = 0;
    }

    public void addBook(Book book) {
        books[index] = book;
        index++;
        totalBooks++;
        book.library = this;← 3
    }
}
```

(a) For each modification below, determine whether the code of the `Library` and `Book` classes will compile or error if we **only** made that modification, i.e. treat each modification independently.

1. Change the `totalBooks` variable to **non** `static` Compiles
2. Change the `lastBookTitle` method to **non** `static` Compiles
3. Change the `addBook` method to `static` Error: can't reference "this"
4. Change the `last` variable to **non** `static` Error: can't reference last in static method if it's instance variable
5. Change the `library` variable to `static` Compiles