CS 61B Fall 2023

Introduction to Java

Discussion 01: January 22, 2024

1 Welcome to CS 61B

The semester has just started and the CS 61B staff are adding some finishing touches to the course infrastructure, but they need your help! You'll need to use the `CS61BStudent` class as defined in the slides, copied below for your convenience.

```
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.grade = 100;
    }

    public void watchLecture() { // Instance Method
        ...
    }

    public static String getInstructor() { // Static Method
        ...
    }
}
```

All of the following parts involve filling in the `CS61B` class on page 3. You may use any of the types discussed in lecture. It's up to you to decide which types would work best for each variable and method signature!

(a) We need to declare (also possibly instantiate and assign!) a few important variables. Recall that variables in the body of the class are compiled even before the constructor fully creates an instance of the class; in other words, carefully consider what information we have access to. Define the following variables within the class:

1. `university`: the name of the university, which should be "UC Berkeley" **for all semesters of `CS61B`**
2. `semester`: the semester that the course is being taught
3. `students`: the `CS61BStudent`s enrolled in this semester's `CS61B`. Remember that the course has a fixed capacity!

(b) Each `CS61B` instance represents one semester of the course. Create a constructor that takes in a `capacity` for the maximum number of students enrollable, an array `signups` consisting of the students who signed up for the course (in order), and the `semester` (e.g. "Fall 2023").
In the constructor, we want to enroll `capacity` students from `signups` and initialize the `semester` instance variable.
Hint: We have both a constructor variable and instance variable named `semester`*. How can we distinguish them?*

(c) Let's now implement some highly-requested features as methods. Consider what the method should return, its argument types, whether it should be `static`, etc.

1. `makeStudentsWatchLecture`: makes every `CS61BStudent` *enrolled* in this semester of the course watch lecture (excluding waitlisted students).
2. `changeUniversity`: takes in a new university name `newUniversity`. Changes the `university` **for all semesters of `CS61B`** to `newUniversity`

(d) Modify your existing implementation to support `expand`, which allows our infrastructure to handle course expansions. Whenever the course expands, students that were originally waitlisted should be enrolled, up until the new capacity.

Challenge: Support course expansions without additional usages of `new`.

Solution:

```
public class CS61B {

    // Variables (part a)
    public static String university = "UC Berkeley";
    public String semester;
    public CS61BStudent[] students;

    // Constructor (part b)
    public CS61B(int capacity, CS61BStudent[] signups, String semester) {
        this.semester = semester;
        this.students = new CS61BStudent[capacity];
        for (int i = 0; i < capacity; i++) {
            this.students[i] = signups[i];
        }
    }

    // Methods (part c)
    /** Makes every CS61BStudent enrolled in this semester of the course watch lecture. */
    public void makeStudentsWatchLecture() {
        for (CS61BStudent student : students) {
            student.watchLecture();
        }
    }

    /** Takes in a new university name newUniversity and changes the university
    for all semesters of CS61B to newUniversity. */
    public static void changeUniversity(String newUniversity) {
        university = newUniversity;
    }

    /** Expands the course to the given capacity. */
    public void expand(int newCapacity) {
        ... // see next page
    }
}
```

(d): Recall that arrays have fixed capacity, so we can't simply append to the end of the `students` array.

One possibility is to keep an additional instance variable to keep track of *all* the signups, and create an entirely new `students` array every time the course expands, enrolling `newCapacity` students in a similar fashion to the constructor.

Challenge: Instead of only adding the enrolled students in the constructor, add *all* students in `signups` (or just have a reference to `signups`). We add an instance variable `capacity` to keep track of the current capacity of the course. All of the methods that iterate through the `students` array should iterate up to `capacity` instead of the entire length of the array. Because `signups` was in order, this is equivalent to only iterating over the enrolled students. The code might look like the following:

```
public class CS61B {

    public static String university = "UC Berkeley";
    public String semester;
    public CS61BStudent[] students;
    public int capacity;

    public CS61B(int capacity, CS61BStudent[] signups, String semester) {
        this.semester = semester;
        this.students = signups;
        this.capacity = capacity;
    }

    /** Makes every CS61BStudent enrolled in this semester of the course watch lecture. */
    public void makeStudentsWatchLecture() {
        // Every student past capacity is waitlisted, so they are not included.
        for (int i = 0; i < capacity; i++) {
            students[i].watchLecture();
        }
    }

    ...

    /** Expands the course to the given capacity. */
    public void expand(int newCapacity) {
        this.capacity = newCapacity;
    }
}
```