Introduction to Java

Discussion 01

Announcements

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1/27 Week 2 Survey Due		1/29 Homework 0B Due		1/31 Pre-Semester Survey Due Lab 2 Due	
	2/3 Week 3 Survey Due Mini-Project 0 Due				2/7 Homework 1 Due Lab 3 Due	

- Welcome to CS 61B!
- Please read our Ed guidelines before you post to make sure everything follows the rules

Meet Your TA!

Add an introduction here. Make sure to make your own copy of the slides before editing, and change the location to your own Drive (not our shared 61B one).

Some things you can include:

- Your name
- Your pronouns
- Your email address
- Your major and year
- Maybe your hobbies, interests, favorites, etc so students can relate to you as a human being
- Maybe a fun picture of you that shows your ✨sparkle ✨

Content Review

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();
        studentOne.getInstructor();
    }
}
```


Instantiating Classes

```
public class CS61BLauncher {
    public static void main(String[] args) {
        CS61BStudent studentOne; // Declare a new variable of class CS61BStudent
    }
}
```


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
    }
}
```


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
    }
}
```


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


    }
}
```


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(); // Does this work?
    }
}
```


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?

    }
}
```


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(); // Does this work?
    }
}
```


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.

Check for understanding: can you reference instance variables in static methods? Can you reference static variables in instance methods?

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

Worksheet

1A Welcome to CS 61B

```
public class CS61B {
      // variables here



      ...



}
```

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`**: all the `CS61BStudents` in this semester's CS61B. Remember that the course has a fixed capacity!

1A Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;




}
```

1. **`university`**: the name of the university, which should be ”UC Berkeley” <u>for all semesters of CS61B</u>

Note that `university` is `static`!

- All CS61B students attend UC Berkeley

1A Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;




}
```

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

1A Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;




}
```

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`**: all the `CS61BStudents` in this semester’s CS61B. Remember that the course has a fixed capacity!

1A Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;




}
```

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`**: all the `CS61BStudents` in this semester’s CS61B. Remember that the course has a fixed capacity!

Notice that we can’t initialize `semester` or `students` yet: we don’t know what the semester or capacity of the class are!

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

      // constructor here




}
```

Each CS61B instance represents one semester of the course.

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”)

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

      public CS61B(int capacity, CS61BStudent[] signups, String semester) {




      }


}
```

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”)

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

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




      }



}
```

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”)

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

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




      }


}
```

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”).

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

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

            }
      }



}
```

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”).

1B Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

      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];
            }
      }



}
```

Define a skeleton for the constructor that takes in:

1. a **`capacity`** for the maximum number of students
2. an array of **`signups`**, students who have signed up to take the course
3. the **`semester`** (ie. ”Spring 2024”).

1C Welcome to CS 61B

```
public class CS61B {
      public static String university = “UC Berkeley”;
      public String semester;
      public CS61BStudent[] students;

      // constructor
      ...

      // methods here



}
```

Add the following methods to the class:

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

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...

      public int makeStudentsWatchLecture() {



      }



}
```

Add the following methods to the class:

1. **`makeStudentsWatchLecture`**: makes every enrolled `CS61BStudent` in this semester of the course watch lecture. Returns how many who actually watched.

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...

      public int makeStudentsWatchLecture() {
            int total = 0;


      }



}
```

Add the following methods to the class:

1. **`makeStudentsWatchLecture`**: makes every enrolled `CS61BStudent` in this semester of the course watch lecture. Returns how many who actually watched.

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...

      public int makeStudentsWatchLecture() {
            int total = 0;
            for (CS61BStudent student : students) {

             }
      }



}
```

Add the following methods to the class:

1. **`makeStudentsWatchLecture`**: makes every enrolled **`CS61BStudent`** in this semester of the course watch lecture. Returns how many who actually watched.

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...

      public int makeStudentsWatchLecture() {
            int total = 0;
            for (CS61BStudent student : students) {
                  boolean watched = student.watchLecture();
                  if (watched) {
                        total += 1;
                  }
             }
      }



}
```

Add the following methods to the class:

1. **`makeStudentsWatchLecture`**: makes every enrolled `CS61BStudent` in this semester of the course watch lecture. Returns how many who actually watched.

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...

      public int makeStudentsWatchLecture() {
            int total = 0;
            for (CS61BStudent student : students) {
                  boolean watched = student.watchLecture();
                  if (watched) {
                        total += 1;
                  }
             }
            return total;
      }



}
```

Add the following methods to the class:

1. **`makeStudentsWatchLecture`**: makes every enrolled `CS61BStudent` in this semester of the course watch lecture. Returns how many who actually watched.

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...


      public static void changeUniversity(String newUniversity) {

      }
}
```

Add the following methods to the class:

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

1C Welcome to CS 61B

```
public class CS61B {
      // variables and constructor
      ...


      public static void changeUniversity(String newUniversity) {
            university = newUniversity;
      }
}
```

Add the following methods to the class:

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

1D Welcome to CS 61B

Modify your existing implementation to support course expansions. Whenever the course expands, students that were originally waitlisted should be enrolled, up until the new capacity. Assume that the new capacity is always less than or equal to the number of students signed up.

1D Welcome to CS 61B

Modify your existing implementation to support course expansions. Whenever the course expands, students that were originally waitlisted should be enrolled, up until the new capacity. Assume that the new capacity is always less than or equal to the number of students signed up.

Recall that arrays have fixed capacity, so we can't simply append to the end of the array.

We can add an additional instance variable to keep track of all students currently signed up for the course, in addition to those enrolled. When the course expands, we can create a new array for the currently enrolled students and the newly enrolled students, similarly to the constructor, and reassign `students` to this array.

Challenge solution: Only keep track of `signups`. Add an additional instance variable for the `capacity` of the course. The students of `signups` below index `capacity` are enrolled, and the ones behind are waitlisted. When expanding the course, we only need to change `capacity`.

Attendance

https://tinyurl.com/cs61b-sp25-disc