Introduction to Objects

Instructor: Ido Avnon
TAs: Abby Williams
          Chloë Mi Cartier
          Connor Sun
          Cynthia Pan
          Katharine Zhang
          Marcus Sanches
          Rohini Arangam

BEFORE WE START

Talk to your neighbors:
Best places to study on campus?

Questions during Class?
Raise hand or send here

sli.do  #cse122
Lecture Outline

• Announcements

• C2 Overview

• OOP Review

• Example

• Abstraction
Announcements

• Programming Assignment 2 (P2) due Saturday!
• Culminating Project 2(C2) out later today!
• Quiz 1 Yesterday... how did it go?
Lecture Outline

- Announcements
- C2 Overview
- OOP Review
- Example
- Abstraction
Lecture Outline

• Announcements

• C2 Overview

• OOP Review

• Example

• Abstraction
Object Oriented Programming (OOP)

- **Procedural programming**: Programs that perform their behavior as a series of steps to be carried out
  - Classes that **do** things

- **Object-oriented programming (OOP)**: Programs that perform their behavior as interactions between objects
  - Classes that **represent** things
  - We’re going to start writing our own objects!
Classes & Objects

• **Classes** can define the **template** for an object
  - <textarea> Like the blueprint for a house!</textarea>
    “What does it mean to be this thing?”

• **Objects** are the actual **instances** of the class
  - <textarea> Like the actual house built from the blueprint!</textarea>
    “It is an example of this thing!”

We create a new instance of a class with the **new** keyword
e.g., `Scanner console = new Scanner(System.in);`
State & Behavior

• **Objects** can tie related *state* and *behavior* together

• **State** is defined by the object’s *fields* or *instance variables*
  - *Scanner’s state* may include what it’s scanning, where it is in the input, etc.

• **Behavior** is defined by the object’s *instance methods*
  - *Scanner’s behavior* includes “getting the next token and returning it as an *int*”, “returning whether there is a next token or not”, etc.
What could be the possible state of a "Human" class?
What could be the possible behavior of a "Human" class?

Start presenting to display the poll results on this slide.
Lecture Outline

• Announcements
• C2 Overview
• OOP Review
• Example
• Abstraction
Representing a Coordinate Point

How would we do this given what we knew last week?

Maybe `int x, int y`?

Maybe `int[]`?
Representing a point

```java
int x, int y
```

- Easy to mix up x, y
- Just two random ints floating around—easy to make mistakes!

```java
int[]
```

- Not really what an array is for
- Again, just two ints—just have to “trust” that we’ll remember to treat it like a point

Let’s make a class instead!
Instance Variables

• Fields are also referred to as **instance variables**

• Fields are defined in a class

• Each instance of the class has their own copy of the fields
  - Hence *instance* variable! It’s a variable tied to a specific instance of the class!
Instance Variables in Java

```java
public class MyObject {
    // fields
    type1 fieldName1;
    type2 fieldName2;
    ...
}
```
Instance Methods

• **Instance methods** are defined in a class

• Calling an instance method on a particular *instance* of the class will have effects on *that* instance
Instance Methods

• **Instance methods** are defined in a class
• Calling an instance method on a particular *instance* of the class will have effects on that instance
Instance Methods

• **Instance methods** are defined in a class
• Calling an instance method on a particular *instance* of the class will have effects on *that* instance
Instance Methods in Java

```java
public class MyObject {

    // instance methods
    public returnType methodName(...) {
        ...
    }
}
```
Syntax

```java
public class MyObject {
    // fields
    type1 fieldName1;
    type2 fieldName2;
    ...

    // instance methods
    public returnType methodName(...) {
        ...
    }
}
```
Lecture Outline

• Announcements
• SearchEngine Recap
• OOP Review
• Example
  • Abstraction
Abstraction

The separation of ideas from details, meaning that we can use something without knowing exactly how it works.

You were able use the Scanner class without understanding how it works internally!
Client v. Implementor

We have been the clients of many objects this quarter!

Now we will become the implementors of our own objects!