LEC 16

# CSE 122

# JUnit Testing

Questions during Class?

Raise hand or send here

sli.do #cse122

## BEFORE WE START

*Talk to your neighbors:*

*Favorite place to sit on campus?*

Instructors Tristan Huber & Hunter Schafer

TAs Ambika, Andrew, Audrey, Autumn, Ayush, Ben, Colton, Di, Eesha, Elizabeth, Evelyn, Jacob, Jaylyn, Jin, Joe, Kevin, Leon, Megana, Melissa, Mia, Poojitha, Rishi, Rucha, Shivani, Shreya, Steven, Suhani, Yijia, Ziao

# Lecture Outline

- Announcements
- Importance of Testing
- JUnit
- Example: Tic Tac Toe

# Announcements

- Reminder: Final Exam, Wed 6/6 @ 2:30 – 4:20 pm
  - In person, on paper
  - Review in section!
  - Review Lecture: Next Wednesday
  - Review session: exact timing tbd
- Programming Assignment 3 due Thursday (5/25)
- Creative Project 3 released Friday (5/26)
  - Last one!!!!!

# Lecture Outline

- Announcements
- **Importance of Testing**
- JUnit
- Example: Tic Tac Toe

# (PCM) Importance of Testing

Software, written by people, controls more and more of our day-to-day lives.

Bugs (just like the ones we all write) are just as easy to write in this software.

Stakes can be quite high so bugs can have catastrophic effects

*Source: Hackaday*

# Practice : Pair

sli.do #cse122

## Bugs you've experienced

Can you think of a bug(s) you've experienced or heard of that have had serious effects?

If you can't, can you think of any absurd bugs you've seen?

# Lecture Outline

- Announcements
- Importance of Testing
- **JUnit**
- Example: Tic Tac Toe

# JUnit Basics

- `import` statements to give you access to JUnit method annotations and assertion methods!
- Method Annotations
  - `@Test`
  - `@DisplayName`
  - `...`
- Assertion Methods
  - `assertEquals(expected, actual)`
  - `assertTrue(boolean)`
  - `assertFalse(boolean)`
  - `...`

# JUnit Testing

```
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
import java.util.*;

public class ArrayListTest {
    @Test
    public void testAddAndGet() {
        List<String> list = new ArrayList<>();
        list.add("Hunter Schafer");
        list.add("Miya Natsuhara");
        list.add("CSE 122");

        assertEquals("Hunter Schafer", list.get(0));
        assertEquals("Miya Natsuhara", list.get(1));
        assertEquals("CSE 122", list.get(2));

        assertTrue(list.size() == 3);
    }
}
```

put object into some expected state

Use assert statements to check if observed state is what we expect

# Using JUnit

- Each `@test` method should be independent
  - ie. set up its own state, make all relevant assertions
- An `@test` fails if any `assert` statement fails
- JUnit executes `@test` methods in an arbitrary order

# Using JUnit - Tips

- one `@test` method per distinct case (i.e., empty case, one element, even, odd, some edge case, …)
- Good coding practices still apply
  - Eg. you can write helper methods in your test file

# Lecture Outline

- Announcements
- Importance of Testing
- JUnit
- **JExample: Tic Tac Toe**

# Practice : Pair

sli.do #cse122

**What test cases can you think of for the TicTacToe spec?**

# Closed or open box tests?

Closed box testing - write tests based on a **specification** independent of any implementation.

Open box testing - write tests for a particular implementation.

Test Driven Development - write tests *before* the implementation

# Bonus Topic: Floating Point Numbers

- Another name for `doubles` are floating point numbers
- Floating point numbers are nice, but imprecise
  - Computers can only store a certain amount of precision (can't store 0.3333333333 repeating forever)
  - Finite precision can lead to slightly incorrect calculations with floating point numbers

```
0.7 + 0.1
0.7999999999999999
```

- Take-away: Essentially can never rely on == for doubles. Instead, must define some notion of how far away they can be to be tolerated as the same
  - JUnit: `assertEquals(expected, actual, delta)`