LEC 16

CSE 122

JUnit Testing

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

Almost there!
Exciting plans for Spring break?

Music: Miya's 23wi CSE 122 Playlist

Instructor Miya Natsuhara

TAs

Ayush	Atharva	Ernie	Ambika
Connor	Julia	Di	Elizabeth
Poojitha	Megana	Logan	Joe
Andrew A	Joey	Shivani	Jin
Andrew C	Eesha	Michelle	Ben
Jasmine	Lilian	Steven	Evelyn
Darel	Thomas	Kevin	Kent
Gabe	Leon	Ken	
Karen	Melissa	Vivek	
Colton	Audrey	Autumn	

Lecture Outline

- Announcements
- Importance of Testing
- JUnit
  - How Much Testing is Enough?
- Example: Brainstorm Test Cases (TFTPlayer)
- Challenge: Floating Point Precision

Announcements

- Reminder: Final Exam, Wed 3/15 @ 12:30 – 2:20 pm
  - Details for logistics and study resources have been posted
  - Sections this week and next are mostly focused on review and practicing writing code by hand!
  - Review Lecture: Next Wednesday
  - Review Session: Tuesday 3/14 @ 4:30pm-6:50pm BAG 131
- Programming Assignment 3 due Thursday (3/2)
- Creative Project 3 released Friday (3/3)
  - Last one!!!!!

Lecture Outline

- Announcements
- **Importance of Testing** 
- JUnit
  - How Much Testing is Enough?
- Example: Brainstorm Test Cases (TFTPlayer)
- Challenge: Floating Point Precision

(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
  - How Much Testing is Enough?
- Example: Brainstorm Test Cases (TFTPlayer)
- Challenge: Floating Point Precision

JUnit Basics

- `import` statements to give you access to JUnit method annotations and assertion methods!
- Method Annotations
  - `@Test`
  - `@DisplayName`
  - …
- Assertion Methods
  - `assertEquals`
  - `assertTrue`
  - `assertFalse`
  - …

JUnit Testing

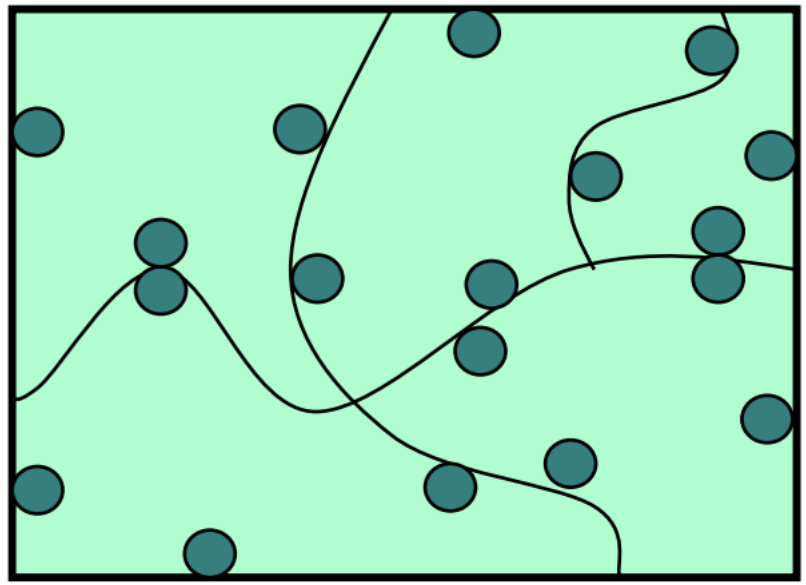
```
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
import java.util.*;

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);
    }
}
```

JUnit Tips

- Write a test method for individual methods *and* test methods that combine different method calls in different combinations!
- Write a test method per distinct case (i.e., empty case, one element, even, odd, some edge case, …)
- Use `assertEquals(expected, actual, message)` to provide a description of what case that line is testing
- Testing code is just code. Use good coding practices (e.g., helper methods to reduce redundancy) to help you write code.
  - It can take time, but if you do it well, developing your solution can be a breeze!

(PCM) How Many Test Cases Is Enough?

- In general, more tests –> more confidence!
- Try to think adversarially and try to break your own code with tests
- Specification Testing (based on the spec) vs. Clear-box Testing (based on how you know your implementation works)
  - Specification Testing you can do *before* writing your solution!
  - Clear-box Testing you do *after* you've written your solution.
- Test a wide variety of different cases
  - Think about **boundary or "edge" cases** in particular, where the behavior should change



Lecture Outline

- Announcements
- Importance of Testing
- JUnit
  - How Much Testing is Enough?
- **Example: Brainstorm Test Cases (TFTPlayer)**
- Challenge: Floating Point Precision

Practice : Pair

sli.do #cse122

What test cases can you test for the TFTPlayer class from the PCM?

Rules from the PCM changed a bit from when the lesson was first posted! All should be consistent now!

Rules

- **Start Game:** 10 gold, 0 experiences, level 1
- **Buy Experience:** Spend 4 gold to earn 4 experience
  - Must have sufficient gold. Can't buy XP if already max level (9)
  - Every 20 experience points is converted to 1 level
- **Gain Gold:** Every player earns 3 gold plus some interest
  - **Interest:** Gain 1 additional gold for each 5 gold owned, capped at interest of 5 gold
  - Example: Have 24 gold. Gold gained would be 7 (3 free gold + 4 interest gold)

Spend 2 minutes brainstorming specification testing

Then 2 minutes brainstorming clear-box testing

Lecture Outline

- Announcements
- Importance of Testing
- JUnit
  - How Much Testing is Enough?
- Example: Brainstorm Test Cases (TFTPlayer)
- **Challenge: Floating Point Precision** 

Challenge: 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)`