JUnit Testing

Talk to your neighbors:
Almost there!
Exciting plans for Winter break?

Instructor: Hunter Schafer / Miya Natsuhara

Music: Hunter/Miya’s Playlist

TAs
- Ajay
- Andrew
- Anson
- Anthony
- Audrey
- Chloe
- Colton
- Connor
- Elizabeth
- Evelyn
- Gaurav
- Hilal
- Hitesh
- Jake
- Jin
- Joe
- Karen
- Kyler
- Leon
- Melissa
- Noa
- Parker
- Poojitha
- Samuel
- Sara
- Simon
- Sravani
- Tan
- Vivek

Questions during Class?
Raise hand or send here
sli.do #cse122
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, Tues 12/13 @ 12:30 – 2:20 pm
  - Details for logistics and study resources will be posted Friday
  - Sections this week and next are mostly focused on review and practicing writing code by hand
  - Review Lecture: Next Wednesday
  - Review Session: Monday 12/12 in the evening, details posted soon

• Programming Assignment 3 due Thursday

• Quiz 3 next Tuesday
  - Adjustments for Quiz 3 Retakes, details Friday
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

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 Testing

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

class ArrayListTest {
    @Test
    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 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 that line of the test is supposed to do

• More specific tests are helpful in debugging

• 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 test -> 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)
• Consider testing entirely different cases
  - Think about **boundary cases** in particular, where should the behavior change
Lecture Outline

• Announcements

• Importance of Testing

• JUnit
  - How Much Testing is Enough?

• Example: Brainstorm Test Cases (TFTPlayer)

• Challenge: Floating Point Precision
What test cases can you test for the TFTPlayer class from the PCM?

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 1 gold plus some interest
  - **Interest**: Gain 1 additional gold for each 10 gold owned, capped at interest of 5 gold
  - Example: Have 24 gold. Gold gained would be 3 (1 free gold + 2 interest gold)
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

\[ 0.7 + 0.1 = 0.79999999999999999 \]

• 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)