LEC 01

CSE 122

Java Review & Functional Decomposition

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:
What is your favorite restaurant around UW?

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/Reminders
- Review Java
- Functional Decomposition
- Code Quality
- First Assignment
  - Grading

Announcements

- Hope you had fun in your first quiz section yesterday!
- Programming Assignment 0 (P0) released later today, due next Thurs (1/12)
  - Focused on Java Review
- Java Review Session Monday 1/9
  - 11:30am-12:20pm in SAV 260
  - 3:30pm-4:20pm in SIG 134
  - Both sessions will be recorded!
- IPL will open on Monday!
- My Office Hours posted
  - Mondays 1:30-3:00
  - Fridays 12:30-1:30

Reminders

- Fill out the Introductory Survey
- Post an introduction video on your section's Ed thread!
- ☆ Complete the pre-class material (PCM) for Wednesday (see calendar)
- Attend quiz section on Tuesday!

Lecture Outline

- Announcements/Reminders
- Review Java
- Functional Decomposition
- Code Quality
- First Assignment
  - Grading

Reminders: Review Java Syntax

Java Tutorial reviews all the relevant programming features you should familiar with (even if you don't know them in Java).

- Printing and comments
- Variables, types, expressions
- Conditionals (if/else if/ else)
- Loops (for and while)
- Strings
- Methods
- File I/O
- Arrays

Practice : Think

sli.do #cse122

In-Class Activities

- **Goal**: Get you actively participating in your learning
- Typical Activity
  - Question is posed
  - **Think** (1 min): Think about the question on your own
  - **Pair** (2 min): Talk with your neighbor to discuss question
    - If you arrive at different conclusions, discuss your logic and figure out why you differ!
    - If you arrived at the same conclusion, discuss why the other answers might be wrong!
  - **Share** (1 min): We discuss the conclusions as a class
- During each of the **Think** and **Pair** stages, you will respond to the question via a sli.do poll
  - Not worth any points, just here to help you learn!

Practice : Think

sli.do #cse122

What is the output of this Java program?

```
public class Demo {
   public static void main(String[] args) {
      int[] nums = {2, 3, 5, 9, 14};

      int totalDiff = 0;
      for (int i = 1; i <= nums.length; i++) {
         totalDiff += (nums[i] - nums[i - 1]);
      }
      System.out.println("Total Diff = ” + totalDiff);
   }
}
```

A) Total Diff = 12

B) Total Diff = 11

C) Total Diff = 7

D) Error

Practice : Pair

sli.do #cse122

What is the output of this Java program?

```
public class Demo {
   public static void main(String[] args) {
      int[] nums = {2, 3, 5, 9, 14};

      int totalDiff = 0;
      for (int i = 1; i <= nums.length; i++) {
         totalDiff += (nums[i] - nums[i - 1]);
      }
      System.out.println("Total Diff = ” + totalDiff);
   }
}
```

A) Total Diff = 12

B) Total Diff = 11

C) Total Diff = 7

D) Error

Case Study: Temperatures

Write a program to prompt the user for how many days' temperatures they want to enter, then asks the user for the temperatures of that many days. The program should then report the highest temperature that occurred and what day it occurred on. The program then should allow the user to ask for the average temperature across a range of days, and report how many of those days were above average.

Review skills

- User input
- For loops
- Arrays
- Maximum calculation
- Cumulative sum

```
How many days' temperatures? 7
Day 0's high temp: 45
Day 1's high temp: 44
Day 2's high temp: 39
Day 3's high temp: 48
Day 4's high temp: 37
Day 5's high temp: 46
Day 6's high temp: 53
The highest temperature was on day 6 and was
53.

What range of days are you interested in? 2 5
Average temperature was 42.5
2 days were above average.
```

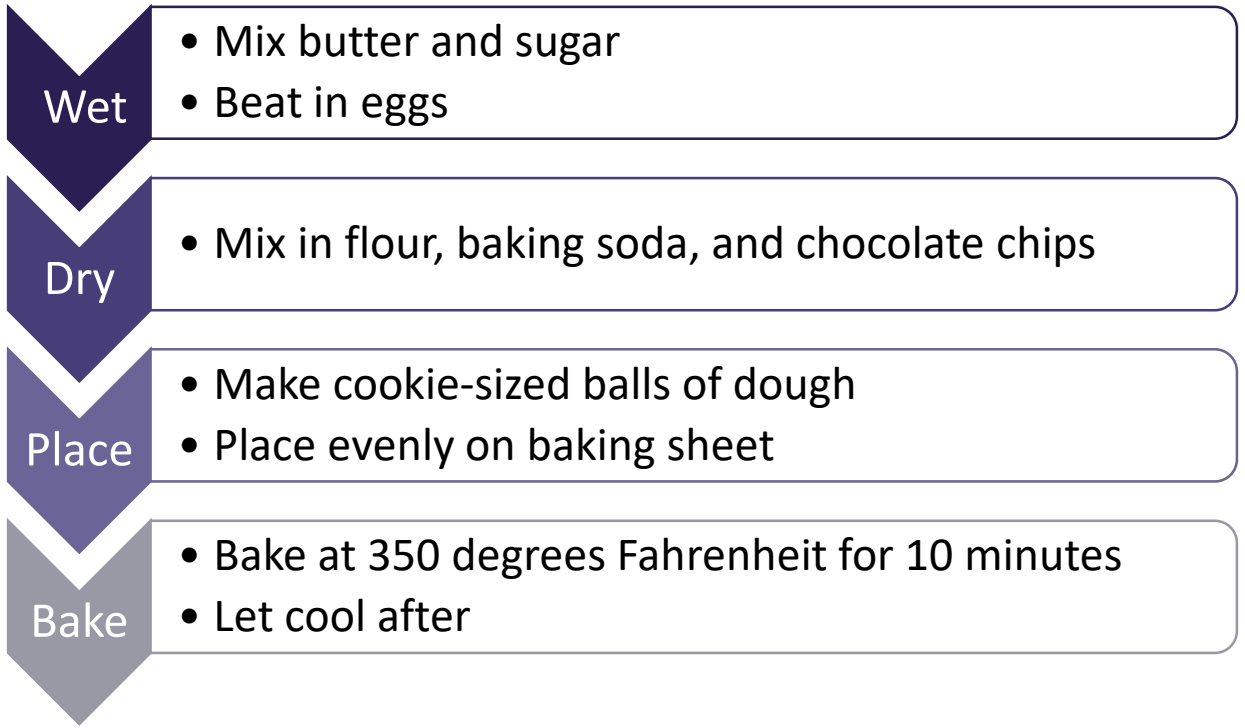
Lecture Outline

- Announcements/Reminders
- Review Java
- **Functional Decomposition**
- Code Quality
- First Assignment
  - Grading

Functional Decomposition

Functional decomposition is the process of breaking down a complex problem or system into parts that are easier to *conceive, understand, program,* and *maintain.*

"Bake the cookies" ⟶



Functional Decomposition

In our code, functional decomposition often means breaking a task into smaller methods (also called functions).

Example: Temperatures

- Read in temp data
- Finding the hottest day
- Finding the average
- Reporting stats on a range

Avoid Trivial Methods

Introduce methods to decompose a complex problem, not just for the sake of adding a method.

Bad example:

```
public static void printMessage(String message) {
   System.out.println(message);
}
```

Good Example:

```
public static double round(double num) {
   return ((int) Math.round(num * 10)) / 10.0;
}
```

Rule of thumb: A method should do at least two steps

- Ask yourself: Does adding this method make my code easier to understand?

Lecture Outline

- Announcements/Reminders
- Review Java
- Functional Decomposition
- **Code Quality** 
- First Assignment
  - Grading

Code Quality

"Programs are meant to be read by humans and only incidentally for computers to execute." – Abelson & Sussman, SICP

Code is about *communication.* Writing code with good **code quality** is important to communicate effectively.

Different organizations have different *standards* for code quality.

- Doesn't mean any one standard is wrong! (e.g., APA, MLA, Chicago, IEEE, ...)
- Consistency is very helpful within a group project
- See our Code Quality Guide for the standards we will all use in CSE 122

CSE 122 Code Quality

Examples relevant for this week

- Naming conventions
- Descriptive variable names
- Indentation
- Long lines
- Spacing
- Good method decomposition
- Writing documentation

Practice : Pair

What does this code do? How could you improve the quality of this code? (No Slido poll)

```
public static int l(String a,char b){
       int j=-1;for(int a1=0;a1<a.length();a1   ++) {
  if (a.charAt(a1) == b) {
        j = a1;
            } } if(j==-1){return -1;} else {
 return j;} }
```

Practice : Pair

What does this code do? How could you improve the quality of this code? (No Slido poll)

```
public static int l(String a, char b)  {
    int j=-1;
    for(int a1=0;a1<a.length();a1     ++) {
if (a.charAt(a1) == b) {
    j = a1;
        }
    }
    if(j==-1) {
        return -1;
        } else {
            return j;
        }
}
```

Lecture Outline

- Announcements/Reminders
- Review Java
- Functional Decomposition
- Code Quality
- **First Assignment**
  - Grading

Graded Course Components

- Your grade will consist of the following categories:
- Each mark is graded on the scale:
  - **E**(xcellent)
  - **S**(atisfafactory)
  - **N**(ot yet)

Category	#	Marks per	Total Marks
Programming Assignments	4	4 (Behavior, Concepts, Quality, Testing/Reflection)	16
Creative Projects	4	1	4
Quizzes	3	3 (3 questions)	9
Exam	1	6 (6 questions)	6

Course Grades

Instead of curving the class, we'll use a bucket system:

- Marks earned place in an initial bucket, additional S+ marks improve grade.
- Must meet *all* requirements of a bucket for initial placement.
- These are *minimum* GPA guarantees – grade can always be higher than min promise.

Minimum Grade	Creative Projects	Programming Assignments	Quiz/Exam Problems
Total Marks	*4 ESN*	*16 ESN*	*15 ESN*
3.5	All (4) S+; 3 E	All (16) S+; 12 E; 3 Es per dim.	13 S+; 11 E
3.0	All (4) S+; 2 E	14 S+; 8 E; 2 E per dim.	10 S+; 7 E
2.5	3 S+; 1 E	12 S+; 4 E; 1 E per dim.	8 S+; 3 E
2.0	2 S+	10 S+	7 S+
1.5	1 S+	8 S+	5 S+
0.7	1 S+	4 S+	3 S+

Programming Assignment 0 – Warm Up

- Released today, due next Thursday (1/12) at 11:59 pm on Ed
  - Can submit as many times as you want before initial submission date with **Mark** button
  - Build good habits: Don't "shotgun debug"
  - While you do have a resubmission for this assignment, important to meet due date to get as much feedback as possible.
- Focused on reviewing Java concepts and Functional Decomposition
  - Different structure than most assignments with *multiple smaller problems*
  - Green checkmark on slide means that problem is done. Green checkmark on whole lesson means assignment is fully done.
- See Grading Rubric for how each dimension is assessed.
- IPL opens Monday!