Project 2: Algorithms Have Power

(Inspired and builds upon Evan Peck's assignments)

In this lab, we will create an algorithm to fairly distribute tips in restaurants.

We previously looked at how software developers like you and me build in rules into the algorithm that the computer executes. The algorithms that we design aim to make people's lives better. And it can. But if we are not careful in how we design the algorithm, it can lead to detrimental effect.

When algorithms make decisions, they impact people often by deciding how resources are to be allocated and who gets access to those resources.

The ability to make decisions on how people should get access to resources and to what extent they should get access to those resources is an element of power. Computing, in this sense, embodies power.

Remember, from the earlier class, we can think about power in two ways. The first, called "power to", involves our ability to do something. For example, with computing, we have *power to* connect with friends and family who are miles away. The other form of power, called "power over", covers an entity (e.g., a person or a software system) having power over another entity (a person or some system). For example, instructors have significant *power over* their class. The one having the power does not need to be a human; it can be any artifact. For example, SFSU's class enrollment system has some *power over* your class enrollment plans in the sense that the system dictates what classes you can enroll, what priority you get, or whether to enforce course prerequisite checks.

The decisions we make in designing the algorithm can have impact on the lives of real people. In software, sometimes the decision gets hidden so deep down that unless we pay close attention, we will not know how the decisions are being made. One way to pay attention is to see how the outcome is affecting people and then back track to look at how the software is dictating the outcome.

In this project, you will first build an application to manage checks in a restaurant. You will design methods to go through a group of checks at the end of the day, computing total sales and the total pooled tip amount. You will then come up with an algorithm to divide the pooled tip

amount among workers in a way that you think is fair. There are many approaches to dividing the pooled tip amount.

In this project, you'll practice...

- **creating loops to manage repetitive action (tracking amount mentioned in checks)**
- analyzing various formulations of fairness in dividing resources
- developing an algorithm that you think is fair in dividing the pooled tip amount
- **implementing the tip division algorithm in Java**
- evaluating alternative approaches to dividing the pooled amount

First: Let's look at how our restaurant manages checks.

Customers after their meal (or purchase) sign a check that lists three elements:

1. The marked cost of items they ate/purchased
2. A space to mark the tip amount
3. The total amount that combined the marked cost and the tip amount

At the end of the day, the restaurant will go through all the checks and calculate the total sale amount and the total pooled tip amount. The pooled tip amount is divided among the staff; different restaurants have different strategies to divide the amount. You will write a strategy for your restaurant later.

Since humans are involved, you will have to consider several corner cases, also called "edge cases". This will allow us to write programs such that it works even in situations where users do unexpected things.

Below we have listed some of the *edge cases* that you have to consider:

1. Manager enters a negative number for either sale amount, tip amount, and total amount.
    - In this you should ask for the number again.
2. Some customers write the tip amount but not the total amount
3. Some customers write the total amount but not the tip amount
4. Some customers do not write anything in which case the tip amount is 0
5. Some customers write a tip amount and a total amount that does not tally properly
    - In this case the restaurant accepts the total amount and calculates a tip by subtracting the marked cost from the total amount
    - If the difference comes to be negative, tip is 0
6. Some customers write a lower total amount than the marked cost
    - In this case the restaurant assumes the tip amount as 0 and considers the marked cost as total amount

[R1] Discord Discussion Work:
In the designated Discord Thread please do the following:

Read some of the strategies that restaurants use in dividing the pooled amount:
https://www.buzztime.com/business/blog/6-ways-split-tips-between-employees/

Discuss the strategies and see which feels fair in the designated Discord Server Thread.

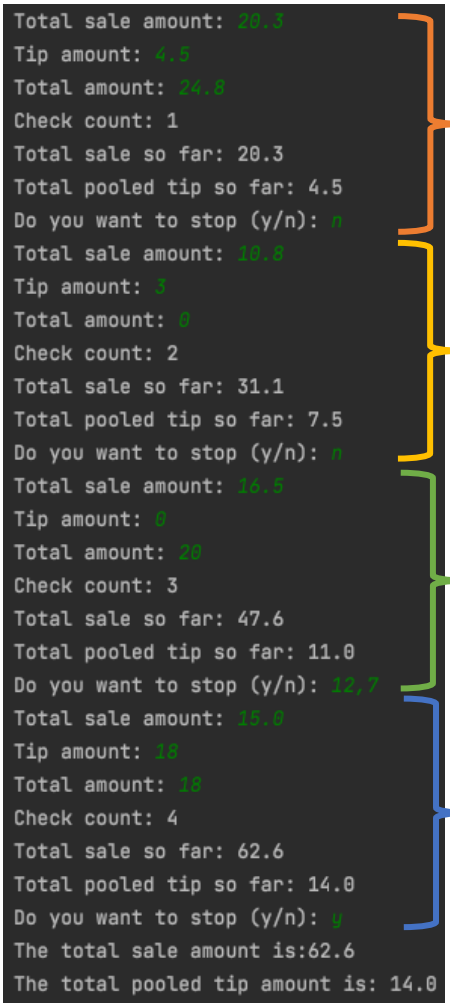
Please note that the **rest of the assignment is individual work.**

[More Content below]

Writing our basic program: Restaurant check management

- Start by creating a class called RestaurantCheckManager
- Write a loop that keeps running until the manager the manager asks to terminate
- Inside the loop, prompt the manager to enter the sale amount
    - If negative number, ask to re-enter value
- Then prompt the manager to enter the tip amount
    - If negative number, ask to re-enter value
- Finally, prompt the manager to enter the total amount
    - If negative number, ask to re-enter value
- Calculate the total sale amount, the total tip amount, and the number of checks
    - Pay attention to the edge cases listed above
- Ask the manager if they want to stop
    - If they type ‘y’ or ‘Y’, terminate the loop
    - Else continue
- After that the program should display the total sale amount and the total tip amount

At this point, your program output must be very similar to the sample output:



In this case, the sale was for $20.3 and the tip amount was mentioned at $4.5. The customer also wrote the total amount correctly ($24.8). The user wants to continue adding so they did not type 'y'.

Here, food worth $10.8 was sold. The customer mentioned $3 for tip but did not write the total amount. We need to infer that the total is $13.8. Total sale so far is $31.1 ($20.3 from above and $10.8 here) and the total tip is $7.5 ($4.5 above and $3 here)

Food worth $16.5 was sold. The customer mentioned did not write the tip amount but wrote the total amount as $20. We need to infer that the tip is $3.5 (20 – 16.5). Total sale so far is $47.6 and the total tip is $11.0. Notice that we continue even when the user typed 12,7 (anything other than 'y' will continue)

Food worth $15 was sold. The customer wrote $18 as tip and $18 as the total. This does not add up. We need to correct the tip as total – sale i.e., 18-5 = **$3.** Now, the user typed 'y' to stop adding more cheques.

After this, you will print out the total sale made that day and the total pooled tip amount.

Below, you will add to this program with tip amount divided across different employees.

The Decision-Maker: How should the pooled tip amount be divided?

Your job is to write an approach that determines how the total tip amount should be split among the employees.

To simplify things, we're going to assume that the restaurant has the following employees*:*

- Servers: 3 (2 were on job that day; 1 was absent)
- Chef: 1
- Sous-chef (or helper chef): 1
- Kitchen aid: 1
- Host/hostess: 1
- Busser: 1

There are several approaches that are used by restaurants. For example, consider a restaurant's division plan of a $150 total tip:

- 30% goes to kitchen – $45
    - 50% to the chef: $22.50
    - 30% to the sous-chef: $13.50
    - 20% to the kitchen aid: $9.00
- 10% goes to the host/hostess – $15
- 10% goes to busser – $15
- 50% goes to servers – $75
    - Split equally even when employees are not present that day, so each get $25

But wait! Don't start yet. First…

[More content below]

Before you code, ask if the existing system fair? What would make it fairer?

While the list above was *one* restaurant's take, there are **many** more potential aspects to consider if you want to create a **fair** algorithm that considers other factors.

[R2] Individual Work*:*

Look at the above plan to split the pooled amount. Do you think it is fair? If so, why? If not, what makes it unfair?

I think that the sample plan is halfway fair because it's trying to acknowledge each of the staff positions. But it's splitting equal parts to all the servers even if one was not working that day, which I feel is not fair to the people who did work.

What other factors would you like to consider when dividing pooled tip amount? Why do you think that factor is important?

To make it more fair, I would divide the server portion only to the ones on duty. And different jobs require different levels of effort and hours, so those must be factored into the split as well.

[R3] Individual Work: Talk to people who you know either worked or are currently working in a restaurant. Create a bullet-point list of factors that came up in conversation. Give each one a numeric priority (-1 to 5 points).

- Factor 1: Hours worked per person - 5
- Factor 2: Role - 5
- Factor 3: Number of tables served - 3
- Factor 4: Customer satisfaction / tip notes - 2
- Factor 5: Experience / Seniority - 1
- Factor 6: Whether the person was working that day - 5
- ....

[R4] Individual Work*: Consider all the factors that have been mentioned. Do these change your initial approach to dividing the total pooled tip amount? If so, what made you change the approach? If not, why not?*

Yes, they definitely do. Initially, I had the opinion that sharing equal role-based percentages would be enough for an equitable distribution of tips. However, upon further consideration and factoring in the number of hours worked and the attendance of a given shift, I concluded that it is far more reasonable to split the tips only among those who were present and worked during that day. In addition, I saw the need to accord more weight to the degree of effort put in by servers and kitchen staff when deciding how the tips are to be distributed.

We are now going to add to our earlier program where we will divide the pooled tip amount and display how much each employee will get.

[R5] Our Algorithm in English

(EDIT THE FOLLOWING LIST TO REFLECT *YOUR* FINAL APPROACH TO DIVIDING THE TOTAL POOLED TIP AMOUNT.)

In our algorithm:

- we are going to give 70% to the servers but only distribute it among those who are working that day
- we are going to give 20% to the kitchen staff where 50% of it will go to the chef, 30% to the sous-chef, and 20% to the kitchen aid
- the rest of the tip amount (10% each) will be divided equally among the host/hostess and the busser

Before you Code, Make Sure It Works Fairly: Theoretical Reflection

[R6] How will you know if your approach is fair? Why do you think it is fair?

Your goal: Briefly justify why your approach is fair.

I personally believe that this particular process is truly just and equitable since it allows tips to be given out in relation to work input and level of effort contribution throughout the shift. This implies that only those individuals who worked and attended duty within the shift will be given a proportion of the tips, and the jobs which involved more work will then be granted a higher proportion of the total tips. Also, this does not provide the opportunity for incentive to those that didn't receive any work done during the day, something which can bring about demoralization and unhappiness on the part of those diligent persons that actually did put in some work.

[R7] Test Cases

(EDIT THE FOLLOWING LIST TO REFLECT THE OUTCOME BASED ON *YOUR* TIP DIVIDING APPROACH. Add at least 3 test cases of your own.**)**

- In a $100 total tip amount, $40 dollars should go to each of the two servers
- In a $150 total tip amount, $7.5 dollars should go to the chef
- In a $10 total tip amount, $0.5 dollars should go to the busser.

[R8] Write Code that Implements Your Algorithm to Divide the Total Tip Amount

Now it's time to translate your algorithm into code.

Your goal: Implement *the algorithm you designed* in IntelliJ.

- Step 1: Use the outlined algorithm you wrote from above.
- Step 2: Implement your algorithm inside the main method. **After** you print out the total sale amount and the total tip amount. The program should display how much each employee will get from the pooled tip amount.
- Step 3: Is it correct? You should check your code with the test cases you outlined above. Input the values that you have identified in the test cases above and evaluate if your program works well. **You must ensure that your code is correct.**
- Step 4: Submit a zip file with the Java file and this completed document.

During your creation, keep a couple of things in mind:

- Use comments to describe what was happening in the program.
- Choose variable names that clearly describe that data that they hold.
- Use spacing to group similar code.

Finally add your reflections to this document.

[R9] Your Reflection: How did your algorithm influence people? How did it have power-over or enable power-to?

(ADD YOUR REFLECTIONS BELOW.)

My algorithm exercises "power-over" others in the sense that it determines who receives how much money which has a direct effect on people's pockets. But it also gives power to workers in the sense that it makes sure those who do the actual work are paid fairly, which is "power-to." The problem is to exercise this power responsibly and openly.

Optional Reading and Watching:

1. Read: Thinking Ethically: https://www.scu.edu/ethics/ethics-resources/ethical-decision-making/thinking-ethically/
2. Watch: Big data and dangerous ideas – a Ted talk by Daniel Hulme: https://www.youtube.com/watch?v=tLQoncvCKxs