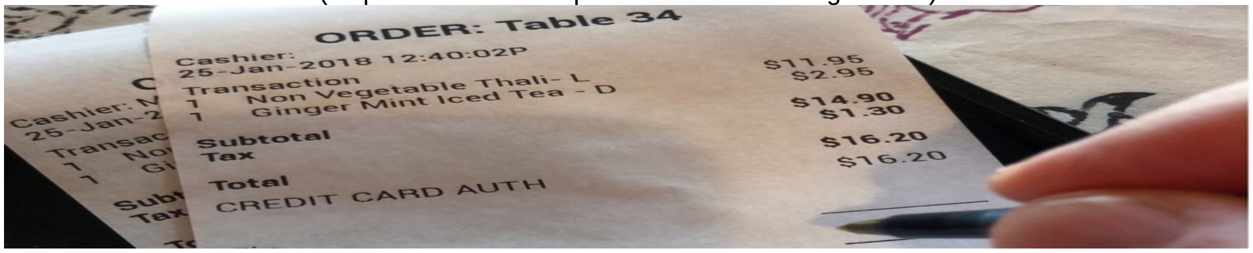
Project: Algorithms Have Political Power

(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 effects.

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 the *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 classes. 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 in, what priority you get, or whether to enforce course prerequisite checks.

The decisions we make in designing the algorithm can have an 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 backtrack 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
- Develop 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 for dividing the amount. You will write a strategy for your restaurant later.

Since humans are involved, you will have to consider several. These cases, also called "edge cases", allow us to write our program such that it works even in extreme situations.

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

1. Some customers write the tip amount but not the total amount
2. Some customers write the total amount but not the tip amount
3. Some customers do not write anything in which case the tip amount is 0
4. 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, the tip is 0.
5. 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 the total amount

[R1] Discussion Session Work:

In your discussion sessions please do the following. If you do not attend the discussion sessions, please have this conversation in your respective Discussion Leader's private Discord channels.

Discussion Session Task:
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 to the group.

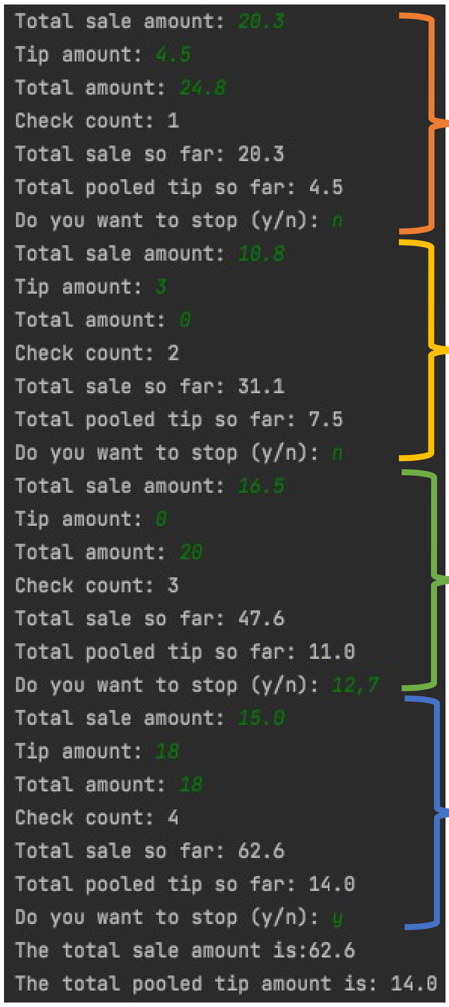
Please note that the **rest of the assignment is individual work.**

[More Content below]

Writing our basic program: Restaurant check management

- Start by creating a class called RestaurantCheckManager (already done in the project zip)
- Write a loop that keeps running until the manager asks to terminate.
- Inside the loop, prompt the manager to enter the sale amount
- Then prompt the manager to enter the tip amount
- Finally, prompt the manager to enter the total amount
- 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:



```
Total sale amount: 20.3
Tip amount: 4.5
Total amount: 24.8
Check count: 1
Total sale so far: 20.3
Total pooled tip so far: 4.5
Do you want to stop (y/n): n
Total sale amount: 10.8
Tip amount: 3
Total amount: 0
Check count: 2
Total sale so far: 31.1
Total pooled tip so far: 7.5
Do you want to stop (y/n): n
Total sale amount: 16.5
Tip amount: 0
Total amount: 20
Check count: 3
Total sale so far: 47.6
Total pooled tip so far: 11.0
Do you want to stop (y/n): 12,7
Total sale amount: 15.0
Tip amount: 18
Total amount: 18
Check count: 4
Total sale so far: 62.6
Total pooled tip so far: 14.0
Do you want to stop (y/n): y
The total sale amount is:62.6
The total pooled tip amount is: 14.0
```

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

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

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

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., 18-15 **= $3.** Now, the user typed ‘y’ to stop adding more cheques.

After this, you will print out the total sale

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: 4 (2 were on the job that day, 1 worked a ½ day, and 1 was absent)
- Chef: 1
- Sous-chef (or helper chef): 1
- Kitchen aid: 1
- Host/hostess: 1
- Busser: 2

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

- 30% goes to the 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 – $7.5 to each busser.
- 50% goes to servers – $75.
    - Split equally even when employees are not present that day or leave early for any reason, so each gets $18.75.

But wait! Don't start yet. First…

[More content below]

Before you code, ask if the existing system is 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?
As a person who used to work as a barista, I feel that the way tips are split up between servers is unfair. Tips should be split up based on the hours that each server has worked that week. For example, if a server worked 30 hours that week, they should be getting a greater portion of tips compared to a server that only worked 12 hours that week.

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

I think adding a factor like "hours worked" should be considered when dividing up the pooled tip amount. This rewards employees what they deserve for working more hours compared to other people.

[R3] Individual Work: If possible, talk to people who you know either worked or are currently working in a restaurant. If not, you can ask people in general. Create a bullet-point list of factors that came up in conversation. Give each one a numeric priority (-1 to 5 points). Try to talk to at least **3** different people.

- Factor 1: Tips should be split up between all the employees evenly, not just the servers. (-1: Chefs get paid a proper wage, but servers earn as low as $2 an hour, so they have to heavily rely on tips)
- Factor 2: Bussers and Hostesses should be considered too. They work at the front of the house. (+1)
- Factor 3: Quality of the service is also important. Servers who get tipped more often due to good service can get a greater portion of tips. (+2)
- Factor 4:
- Factor 5:
- Factor 6:
- ….

[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?*

I feel like the hours worked by Host/Hostesses and Bussers should also be considered to get a fair share of tips. I would also like to reward servers who give better service, but I feel like the algorithm would be more complicated as a result.

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

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

In our algorithm:

- 30% is going to the kitchen staff. 50% goes to the chef, 30% goes to the sous-chef, and 20% goes to kitchen-aid
- 10% goes to hostesses
- 10% goes to bussers
- 50% goes to the servers, but is evenly distributed based on how many hours each server has worked that week.
- First ask the manager how many servers they have in the restaurant
- Then ask how many hours that server has worked that week
- Use the formula (total hours worked by server)/(total hours worked by all servers)*(total tips * 0.5 ← share of tips to servers)

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.

My approach is fair because it takes into account the amount of hours worked for each server. Servers who clock in more hours get a greater share of the tips as compared to servers who worked less hours that week.

[R7] Test Cases

(MODIFY and/or REPLACE THE FOLLOWING LIST TO REFLECT THE OUTCOME BASED ON *YOUR* TIP-DIVIDING APPROACH. Add at least 3 test cases of your own.**)**

- 30 (50 30 20) 10 10 50 (hours worked)
- In a $100 total tip amount: $30 goes to the kitchen (chef: $15, sous-chef $9, kitchen-staff $6). $10 goes to hostess. $10 goes to busser. $50 goes to servers.
- In a $150 total tip amount: $45 goes to the kitchen (chef: $22.50, sous-chef $13.5, kitchen-staff $9). $15 goes to hostess. $15 goes to busser. $75 goes to servers.
- In a $40 total tip amount: $12 goes to the kitchen (chef: $6, sous-chef $3.60, kitchen-staff $2.40). $4 goes to hostess. $4 goes to busser. $20 goes to servers.
- In a $67.41 total tip amount: $20.22 goes to the kitchen (chef: $10.11, sous-chef $6.07, kitchen-staff $4.04). $6.74 goes to hostess. $6.74 goes to busser. $33.71 goes to servers.
- Servers tips are $50. If a server A worked 20 hours and server B worked 8 hours, server A gets $35.71 in tips and server B gets $14.29 in tips

---

[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 the 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? For example, consider how your tipping strategy could affect your employee's ability to make choices or how they interact with customers or fellow employees.

(ADD YOUR REFLECTIONS BELOW.)

I felt that my algorithm treated everyone fairly in this situtation. It allowed all the server employees to get an equal amount of tips based on the hours they worked. I feel that with this algorithm, servers are more incentivized to work longer hours and pick up other people's shifts to get more tips.

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