The main idea behind this activity is to create a program that accepts multiple inputs from a user and uses that information to control the operation of the program. For this project you will need to utilize conditional statements and loops.

We highly recommend you start work on this project now even though you're going to need to use loops and we haven't completed covering them yet. Plan (and code) as much as you can now and add the parts requiring loops later. Incremental development is better than waiting until the last minute and rushing to incorporate all the requirements.

Preliminaries

The owner of a candle shop has asked for your help. The shop sells three types of candles (e.g., birthday candles, tea lights and pillars). The shop owner wants a program that allows them to set a name, a unique type number, the price, and the burn time for each candle they sell. For example, a candle named "Tea Light" might be a type 1 with a cost of $0.50 and a burn time of 60 minutes. Using this information the program will be able to perform certain calculations when customers buy different numbers of the candles. You have been provided the Candle class

Download Candle class

that contains everything you need to know about a Candle object. You must use this class and cannot change it.

Part A - Develop an algorithm (5 points)

Use either a flowchart or pseudocode to develop an algorithm to satisfy the following requirements:

1. Prompt the user (the shopkeeper) to enter the name, type, price and burn time for each of the three different types of candles, types 1, 2, and 3. Once you have this information, instantiate three Candle objects, one for each type using the information the user provided. This step can be considered the store setup.
2. Assume a customer wants to make a purchase. Prompt the user to enter the number of candles of each type the customer wants to buy. Since you can't buy a fraction of a candle, the number bought for each type of candle must be an integer. (You may assume that only integer values between 0 and 10 will be entered when this program is tested.)

3. Calculate the total price of all the candles bought using the information provided in step 1 and by calling the appropriate getter to find out the price of each type of candle.
4. The shop owner wants to offer a discount based on the total purchase price. Deduct this discount from the purchase price.
    I. If the price is more than $20 but $35 or less they receive a 5% discount.
    II. If the price is more than $35 but $55 or less they receive a 7% discount.
    III. If the price is more than $55 but $100 or less they receive a 10% discount
    IV. If the price is more than $100 they receive a 20% discount.
5. Calculate the total burn time of all the candles if they were burned consecutively (i.e., one after the other). Again, use the appropriate getter of the Candle class to determine the individual burn time.
6. Calculate the total cost-per-minute for that purchase using the discounted purchase price and the total burn time.
7. Output some kind of meaningful display that includes the number of candles of each type bought, their name and type, the total price (before and after discount if applied), the total burn time, and the cost-per-minute to burn all the candles all the way down. You can be as creative as you want with this!

As you start working on this project you should keep in mind there are several ways to accomplish this and, as with any any design problem, your solution may evolve. Iteration is a key concept in software design and implementation! It's important that you plan and visualize your solutions before you start to code. One way to start is to list all the items of information you need to keep track of in your program. Then think of the different actions that need to occur and how they relate to each other and to the other information. Another way to start is to plan what you want your ultimate output to look like then work backwards to determine how to create it. It is perfectly fine to start with a general solution then add details and descriptions. Your original effort might not be optimal or elegant but if it accomplishes the task you're on the right track. Once you have a working solution you can start thinking of ways to accomplish that same task in a better way. But remember - make it work first, you can always make it pretty later. A pretty solution that doesn't work isn't useful to anyone.

Store your algorithm in a file (file format can be text, pdf, or doc) with the name **CandleShopSteps**. As the coding develops, don't be afraid to change your algorithm if you realize it needs improvement. All improvements don't have to be done at once - small steps still moves you towards your goal.

Part B - Write the code (20 points)

Once you've written your algorithm it is time to turn it into a Java program that can be executed:

- Name your class **CandleShop**.
- The program will need to import the Scanner class from the java.util package.
- The class should have a main method that will include the logic for the algorithm you developed in Part A. You may choose to add additional methods that are called in the main method.

Don't forget to test it with various kinds of input and manually verify that your program's calculations are correct.

Here is one test scenario to get you started - the shopkeeper sets the price of three types of candles to $5.99, $10.99 and $14.99 with burn time of 3, 4 and 5 hours respectively. If a customer buys 3 candles of the first type, 2 candles of each of the second and third types the total price before discount would be $69.93 and after the discount the purchase price is $62.94. The total burn time is 27 hours (1620 minutes) with a cost of $0.04 per minute.

We recommend you start your class so it looks something like this:

```
public class CandleShop {
    Run | Debug
    public static void main(String[] args) {
        // *************
        // Project 3
        // *************
    }

}
```

Warnings

- Sometimes your double amounts will be printed with excessive decimal places like 62.937000000000005. You're not doing anything wrong; this is just an artifact of using the double data type to represent currency. There is no need for you to change this output. If you choose to, you can refer to module 2 lab for an example on how we used casting as a workaround to round the double number to two decimal places.

- If you used the nextInt() or nextDouble() methods to read in user input followed by nextLine() to read more values you might notice that the program is skipping some prompts. This occurs when the nextInt()/nextDouble() methods are used to read in user input. These methods do not capture the new line after the user enters an integer or double value. Which results in the Scanner consuming the new line the next time it is called which makes it seem that the program is skipping the next Scanner code. To fix this problem, you can call the nextLine() method immediately after each nextInt()/nextDouble(). Here is a code snippet as an example:

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter meal number:");
int mealNumber = sc.nextInt();
sc.nextLine();// to skip the newline left-over from nextInt()
System.out.println("Enter meal name:");
String mealName = sc.nextLine();
System.out.println("Enter meal price:");
double mealPrice = sc.nextDouble();
sc.nextLine();// to skip the newline left-over from nextDouble()
```

Part C - Visualizing the purchase (10 points)

Once you have a working solution, you can begin to use loops to create a histogram of stars that show the breakdown of the different candle types and the total number of candles purchased.

For example if a customer buys five candles of type 1, seven candles of type 2 and three candles of type 3, you might format your histogram like the following:

```
Histogram of number purchased:
5 type 1 (Birthday) candles: *****
7 type 2 (TeaLight) candles: *******
3 type 3 (Pillar) candles: ***
```

We are building the horizontal "bars" of the histogram by including one star (*) per candle of each type being purchased. That means our program has to calculate the number of asterisks to include. I decided to include the name of each candle type in my output but you can chose something different as long as the stars are correct. Hint: Loops can be useful when you need to repeat a certain action a given or unknown number of times, for instance, in building a string by concatenation.

Bonus (+ 5 points)

Enhance the program by validating some of store setup values using the following rule: the program does not accept negative or zero values for price, burn time, or number of candles.

One approach would be to detect an error condition, print a useful error message, then exit the program. Another, more comprehensive, approach would be to require the user to enter a new value until a valid one is entered before moving on to get the next input. Consider how you could use a loop to continue to require the user to enter input if they enter an invalid value. A boolean variable might be a good way to control such a loop.

Bonus-Bonus (+5 points)

Instead of a star, use a unique symbol for each type of candle to create more interesting histograms! For example:

```
Histogram of number purchased:
5 type 1 (Birthday) candles: #####
7 type 2 (TeaLight) candles: @@@@@@@
3 type 3 (Pillar) candles: +++
```

Bonus-Bonus-Bonus (+5 points)

The shop owner wants to have a loyalty program where customers could earn points based on their purchases. Add the functionality to determine the number of points earned based on the total number of candles purchased if every even multiple of ten candles purchased earns one point. For example, purchasing 7 candles would earn no points, purchasing 11 would earn 1 point, purchasing 37 would earn 3 points, etc. Include the number of points earned with each purchase in your output from Part A, paragraph 7.

Challenges

You might face challenges as you attempt this project. This is normal. Trust us: you are not the first one to feel this way when presented with an involved programming project. We are confident you can figure it out, and complete it. Try working out the solution on a piece of paper; the math isn't involved or complicated. Then start thinking about how to move your calculations into code. Make sure to ask for clarification if something is not quite clear and remember to utilize the resources available for you such as office hours.

Coding Style (3 points)

As we mentioned in class, formatting your code makes a difference when someone else has to look at it and review it. It is important to practice good formatting habits. Here are some guidelines to follow:

- Appropriate variable names. In general, do not use single letters like X or create strange abbreviations. Make your variables names descriptive (e.g., hrlyWage, rent, ...). Always use camelCase!
- Proper indentation.
- Good commenting explains what code is doing.
- The solution is well-organized, possibly even elegant, and "reads" like a story that another programmer can follow.

Submission Requirements (2 points)

1. Submit the pseudocode file (named **CandleShopSteps**) for Part A.
2. Submit the code file for part B (**CandleShop**).
3. Submit a PDF document with the name **project3Info.pdf** that includes the following assignment information:
    1. List of references used to acquire information, if any.
    2. Explanation of any special or additional features you added, if any.
    3. Explanation of status, stopping point, and issues if incomplete.
    4. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
    5. Did you learn anything that will help you perform better on future assignments?