Lab 6

How to create and use your own Methods.

In this Lab, you will learn what methods are and how to create them. By the end, you should know how to define a method, call a method from within the main, how to pass values using parameters and return statements, and will also learn more on the importance of scope.

In-Lab Activities

*Don't forget to answer the **Guided Inquiry Questions** in your **notebooks**. It is also encouraged to use it frequently while working through activities.*

Warm-Ups

Go to the Google Forms link on the board and complete your Warm-Up Questions. You have 10 minutes to complete this. If you finish early, feel free to begin the Lab Activities, but make sure your partner is ready as well.

Activity 1 – Making a Calculator

In this activity, you will be able to create a basic calculator that includes all the basic math function such as adding, subtracting, etc. In order to make the calculator, you will need to make a new method for each specific task (such as having an addition method).

What is a **method**? How is it helpful in organizing code?

First, write four basic methods:

1. Below the main method, **define** the first method; this one will be for addition. First, create the **method header:**
    a. What will you put as the **method name?** Make sure it clearly expresses what this method will do.
    b. This method will be adding two numbers together and returning the sum – knowing this, what will you make the **return value type**?
    c. What should be included in the **parameter list?**
2. Next, write the **method body**. It should add the two **parameters** together and **return** the sum. What will the **return statement** look like? Keep it simple and don't include any output or input statements quite yet – these will be done back in the main method.
3. Now do the same thing for the next three methods: subtraction, multiplication, and division.

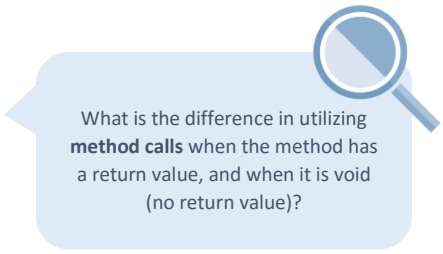
Briefly describe the different parts of a method header: **method name, return value type,** and **parameter list.**

After this, add *three additional* methods of your choice. What other operations could a calculator do? Again, follow the same steps above in order to create these new methods.

For each of the seven operations above, the corresponding static method should be in the class file. Pay attention to curly braces.

Now we need to implement these methods back in the **main method**:

1. Create a menu to allow the user to choose from the 7 options, and allows them to keep choosing after each one, until they decide to quit. What kind of **loop** could you use for this? Will you decide to use a **switch statement** or **if else statement**?
2. For each option, include proper input and output, and then **call the method.** Since each method returns a value, will you have that value **saved in a variable** to display, or **have the returned value itself displayed**? Don't forget to send in the correct values in the **parameter** (you can use variables as well, instead of just fixed values).



Brainstorming Individually

(3-5 minutes) Read the next Activity and start **brainstorming**; think about ways you could start solving this problem and write down any notes – perhaps listing out information on how to create the **methods**. Do this *individually* for 3 to 5 minutes and then come together afterwards to compare, discuss, and code.

Activity 2 – Invoice Calculator

In Lab 3, you wrote a program for a restaurant, which had no methods. Create a program similar to that one (you can even pull up your old code to edit if you wish), but with methods to calculate the billing. The following methods are required:

- calculateDiscount() //This will calculate what discount percent the user gets.
- calculateFinalPrice() //This will calculate the final price after the discount and adds tax to it.

You must determine the parameters and return value type for each method yourself. You may also add your own methods if you believe it will be useful.

CHECKPOINT

45 MINUTES

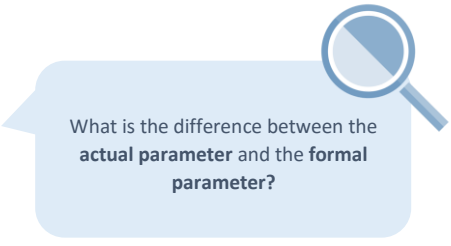
Brainstorming Individually

(3-5 minutes) Read the next Activity and start **brainstorming**; think about ways you could start solving this problem and write down any notes – perhaps listing out information on how to create the **methods**. Do this *individually* for 3 to 5 minutes and then come together afterwards to compare, discuss, and code.

Activity 3 – More methods

Write the following **static methods** and run them in the main to test.

- getHighest() – this method will take 3 int values as parameters and then return the highest of them.
- getLowest() – this method will take 3 int values as parameters and then return the lowest of them.



- getAverage() – this method will take 3 int values as parameters and then calculate and return the Average.

With these methods, you can compare grades for three students. Add code to do this in the **main method**. Don't forget proper input and output and **call your methods** to accomplish this.

Once complete, ask a TA to check.

CHECKPOINT

25 MINUTES

Challenge Activity: The Smart Talking Robot

Pair Programming: Since this is a very challenging program and will take time, switch roles every 20-30 minutes.

In recent times, Artificial Intelligence has become a bigger demand. A simple and fun tool related to AI is a chatbot. There are actual competitions, called the Loebner Prize, based on who can create the most "human-like" chatbots. The key to creating a good chat bot lies in the way it talks and responds in conversation.

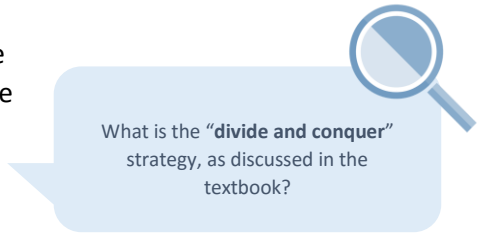
Write a program that will behave similarly to the chatbot you created in Lab 5, except the chatbot's responses will no longer be random – they will be "smart" and somewhat more realistic. For example, if the user says something with the word "father" or "mother", the chatbot could respond with "Oh! What about your parents?". Or if the user says something about their brother or sister, it could say "I have a sister. Her name is Siri." Perhaps the user hits enter without typing anything at all; the chatbot could ask "I'm sorry, I didn't hear anything. What did you say?"

Add these Advanced Responses:

1) If the user writes "I want **apples**." The responses should be "Why do you want **apples**?"
    a. First, find whether or not the string the user entered contains "want". If so, find the position of "want" and create a substring using that position to get the word after it ("apples" in this case) – have this word saved in a new string.
    b. Next, concatenate and use String methods to make a new String that saves the chatbot's response, similar to the one shown above: "Why do you want apples?", where "apples" is whatever word the user typed in originally.
    c. **Another example:** The user types in "I want potatoes." So the chatbot's response is "Why do you want potatoes?"
2) If the user writes "I want to buy a gaming computer" the response should be "Would you be happy if you had **a gaming computer**?" Use the same technique as the steps above describe, but instead of having to check for "want", here you will check for "want to buy".
    a. Notice that since "want to buy" still contains "want" (from Step 1), you will have to be more careful so that multiple responses don't appear. How would you make it so that **only one chatbot response is shown**, rather than both? Think about the **if else statement** and how it is structured with levels – if the first condition comes out to be true, the rest are ignored.

3) If the user enters “I like to **bake cheesecakes**.” The response should be “Why do you like to **bake cheesecakes**?” Look at the pattern and think about the steps you took in the previous chatbot responses. What substring would you check for, and what substring would you save to create this response?

Make a method called “getRespone()” that will determine what string the chatbot will use to respond to the user. The **method body** will be the code you wrote for the steps above. In addition to this, what will be the required **parameter(s)** and **return type**?



At the end of this method, include a pool of 5 responses that can be randomly chosen if **none of the other criteria match** what the user has entered (i.e. if the chatbot cannot use any of the responses you created). For example, one of these could be “I don’t understand what you’re saying. Can you reword that?”

Make sure to include proper **input** and **output** back in the main method, as well as looping, so that the user can try to hold a conversation with your new chatbot.

Once complete, try to add at least three more Advanced Responses of your own. Put each one of these into their own method, and have those methods called by the getResponse() method when your condition is present.

Reflection

Go to the Google Forms link given by your TA to work on your Reflection for Lab 6. This counts as a part of your participation grade, and it is important that you put as much detail as possible.