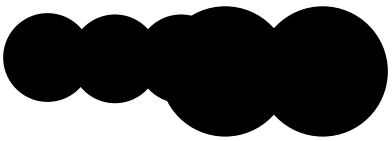
Asst. 3: Make me a sandwich!



1 Preparation

You are expected to be familiar with the following textbook sections and lectures *before* coming to the lab. You are also recommended to complete the exercises below before starting to code. Bring the answers to the lab, if you would like feedback before coding your solution.

This assignment involves invoking methods. There will be a brief explanation of this process during the lab. We will learn more about it soon.

Textbook sections

- 3.1 - 3.5, 3.13

Most relevant lectures

- L5, L6

Exercises

1. Read through the entire assignment, to get a good understanding of the problem you need to solve.
2. Write a full algorithm to solve the `Sandwich` problem. Do not hand-write Java code. The algorithm should have:
   - input(s) – including requirements and names for the inputs
   - a list of variables
   - numbered, precise steps
   - output(s)

2 The sandwich problem

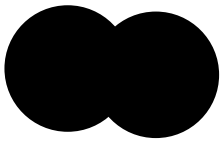
You will create an algorithm and a program to solve the following problem. A customer would like to order a sandwich. They will select various options, and the program will then remind them what **type** and **size** of sandwich they ordered, as well as displaying a **price** for the sandwich.

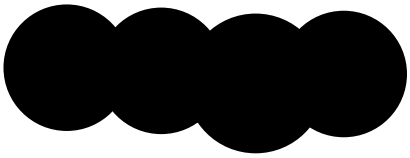
Types: There are four types of sandwich. Three are standard sandwich choices, or the customer can create a custom sandwich.

Sizes: There are three sizes, with different base prices. Half-size costs $4.00, full-size costs $6.50, and overstuffed costs $8.00.

Bread and sauce: Custom (build your own) sandwiches have the option to upgrade bread or sauce. Special artisan bread costs an additional $1.00. Extra sauce costs an additional $0.50.

Toppings: Only when building a custom sandwich: There are eight regular toppings to choose from. The customer may select 3 of these at no charge, and then each additional regular topping costs $0.50. There are five premium toppings to choose from. The customer may





select 1 of these at no charge, and then each additional premium topping costs $1.50. (You do *not* need to keep track of which toppings the customer wants. Only the number of regular and premium toppings is important.)

2.1 Specifications

Write a program to solve the above problem.

User input: When prompting the user for input, you are required to use one of the provided print methods, defined by the instructor in `Sandwich.java`. You are only permitted to scan in integer types from the user.

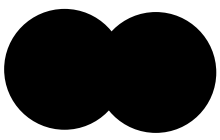
Output: Your program must print the correct price in dollars of the desired sandwich, as well as the name of the sandwich. This means the name of the type of sandwich ordered, and the name of the selected size. If a custom sandwich is ordered, you do not need to print out selected toppings and upgrades.

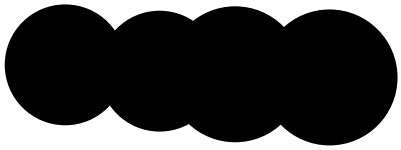
Sample console (standard sandwich) when the program has terminated. Note that if the user selects "Build your own" (custom sandwich), they need to also be prompted to select regular and premium toppings, and they must be asked if they would like to upgrade their sauce or bread.

```
MENU:
(Please select one of the following numerical options.)
1 Chicken Parmesan
2 Turkey Pesto
3 Breakfast Sandwich
4 Build your own
2

SIZE:
(Please select a size.)
1 Half-size, $4.00
2 Full-size, $6.50
3 Overstuffed, $8.00
1

You have ordered a half-size turkey pesto sandwich.
Please pay $4.00.
```

2.2 Instructions

1. Compare your algorithm with your partner's. Discuss if you will merge the two algorithms, taking components from each, or if you will simply choose to follow one partner's algorithm.
2. Download the file `Sandwich.java` from D2L. Open this file in SciTE.
3. Modify the file header to include identifying information as required by the **Lab Guide**.





4. Write a descriptive comment in the file header, explaining the inputs, outputs, and functioning of your program.
5. Compile and run the program before entering any code. Note the print methods that are defined below the main method. Note the statements inside the main method that *invoke* these print methods.
6. You will **only modify the main method!** Do not change any of the other methods (the code below the main method).
7. Now comment out all the calls to (/invocations of) the print methods. Do not comment out the method definitions themselves. As you write your code, you will need to call each of the predefined print methods.
8. Translate your algorithm into Java code, one line at a time.
   (a) Test each selection structure after you write it, to see that it works as intended
   (b) Do not write all your code at once, without testing.
   (c) Write descriptive comments as you go.
   (d) TIP: Start with only the standard sandwiches (and sizes), and once you get that working, move on to allowing custom sandwiches.
   (e) Use proper indentation with your selection structures. Choose the simplest conditions you can think of.

Submission

Recall that submission instructions are in the **Lab Guide**. You are required to submit **one** `.zip` folder containing:

- the properly documented and styled source code file `Sandwich.java`

Make sure you always also save a copy of the finished code to your personal H: drive.