

## CS456 Homework Assignment 2

**Due date:** \_\_\_\_\_

**General info:** Turn in all code on both paper and by email (to [dbahr@Regis.edu](mailto:dbahr@Regis.edu) with “CS456 Homework” in the subject line). Use old-fashion paper for everything else. For calculations, please show all your work.

In the real world, your ideas and your work must be communicated effectively to your boss, to your client, and to potential investors. Therefore, please use succinct but clear and well-written English for your answers (as if I was your boss). Sometimes you will need to look up salient facts to effectively support your ideas and answers. Please briefly quote any sources.

---

**Problem #1:** Give an example of a program that you use frequently but that you do not like. What about that software is poorly engineered?

**Problem #2:** In high school geometry you may have learned that constructing a nonagon with a straight edge and compass is impossible. However, it is straight-forward to construct a polygon that is arbitrarily close to a nonagon. Keeping the halting problem in mind, why does the nonagon (as an analogy) bode well for the future of software engineering?

**Problem #3:** Using your best software engineering, create code that includes a SmallFish, a BigFish, and a FishPond. The FishPond should have a method that prints the name of any fish that is passed to it. Explain why your code is well-engineered.

Hint: Class casting.

**Problem #4:**

(a) In class I showed you procedural code for a tuna sandwich robot. Keeping flexibility and modularity in mind, write object-oriented code for a robot that builds a tuna sandwich on rye bread. Note: Your code should include *all* classes and methods, but you may leave any robot-controlling methods empty.

(b) Assume your kid sister is annoying and insists on ham and cheese on white bread instead. Point out what methods or other code would have to change in your OO program. Can you point out any *specific* features that make your code well-engineered?

Hints: (1) The “Band” code presented in class is very similar to this problem. Study it for guidance. (2) Examples of appropriate classes and methods:

```
Sandwich mySandwich = new TunaSandwich(); //note implied inheritance
mySandwich.build(); //instantiates ingredients
//and adds them to sandwich
```

```
Ingredient tomato = new Tomato(); //note implied inheritance
tomato.addIngredientToSandwich(); //controls robot
```

Start by creating some ingredient classes. Then create the TunaSandwich class. You may do this problem many ways, but consider making build() an abstract method of Sandwich. The concrete implementation of build() would instantiate the ingredients and add them to the sandwich. You might also want to make addIngredientToSandwich() an abstract method.

And remember, you do not have to fill in all of the concrete methods for this problem because (a) we do not actually possess a sandwich building robot, and (b) filling in the code details would be impossible without the actual robot. If you follow my general hints, the concrete versions of addIngredientToSandwich() will be empty (because they control the robot), but all other methods will be complete.