Week 3 Quiz 2

1. What functionality does the Factory pattern provide?

   - Ensuring at most a single instance of class is created and providing global access to it
   - Serializing objects as they are created
   - Enabling multiple identical objects to be created at once
   - Decoupling an object's creation from its usage by allowing subclasses to alter the type of objects that are created

   The factory pattern allows subclasses to alter the way objects of a superclass are created depending on the desired features of the object

2. Imagine you are building an application for an online store. When customers checkout you will need to calculate tax. Depending on where the customer lives, you will need to follow one of many possible formulas to calculate tax. Which pattern introduced in this module best suits this situation?

   - Strategy Pattern
   - Template Method Pattern
   - Factory Pattern
   - Singleton Pattern

   The strategy pattern allows for the creation of multiple functions that can be used interchangeably. In this instance, a different function could be defined to calculate tax for each

3. Which of the following tasks are LLMs well suited to do when helping you implement software design patterns?

   - Brainstorming potential design patterns to use based on a project's needs

     LLMs are great tools for brainstorming potential design patterns

   - LLMs can help alter existing code to implement design patterns

     LLMs often will be able to rewrite existing code to align it with a provided design pattern.

   - LLMs can provide examples of how a design pattern can be implemented

     LLMs can generate example code that demonstrates how a design pattern is used.

   - Deciding for you which patterns to implement

4. Which of the four patterns covered in this module is being implemented by the code below?

```
class Company:

    def process_time_series(self, conn):
        self.load_time_series(conn)
        self.preprocess_data()
        self.calculate_bollinger_bands()
        self.postprocess_data()

    def preprocess_data(self):
        pass

    def postprocess_data(self):
        pass

class DomesticCompany(Company):

    def postprocess_data(self):
        print(f'Postprocessing data for DomesticCompany: {self.name}')

class ForeignCompany(Company):

    def postprocess_data(self):
        print(f'Postprocessing data for ForeignCompany: {self.name}')
```

- Template Method Pattern
- Strategy Pattern
- Factory Pattern
- Singleton Pattern

This code defines a template method that is then overridden by subclasses, which is the structure of the Template Method class.

5. Which of the following prompts is most likely to result in high quality code to help refactor a piece of code using the template method pattern?

- Write the template method pattern.
- You are an expert on software design patterns. Write the template method pattern
- You are an expert in software design patterns. I am building a Python application to track stock performance of a variety of foreign and domestic companies. The code below processes the data for each company. Refactor this code using the template method pattern to allow different approaches to processing company data for domestic and foreign companies.

  ...code pasted here...
- You are an expert in software design patterns. Refactor the code below to use the template method pattern.

  ...code pasted here...

  This prompt assigns a role, includes the code to be refactored, provides details about the context of the project, and includes the sample code to be refactored.