Design Patterns

CS 121, second year students

Assignment 3

September 30, 2022. Due to October 20.

Overview

In this assignment you have to restructure previous classes from *Assignments 1, 2*. You must implement the Factory Method and Abstract Factory patterns. Current class structure is recommended, but you may slightly change them keeping the main logic.

Description

- **Factory method task**.

1. Implement container class **Team** using ***@dataclass*** decorator in Python 3 (or standard Java/C#/C++ class style)[1,2].

Team
+id: int +title: str +member_list: list +project_id: int

2. Implement the abstract base class ***Project*** (modify existing). There are two methods which should be *abstract*: ***add_employee(member: Employee)***, ***remove_employee(member: Employee)***.

Project
 # abstract methods

3. Implement **Web, Mobile, Embedded** classes**.** These classes should inherit the ***Project***. You have to add some unique attributes to these classes (decide on your own).
4. Implement ***SoftwareArchitect*** abstract class**.** *Remember that Python supports multiple inheritance, so you can inherit* ***SoftwareArchitect*** *from* ***Employee*** *either.*

SoftwareArchitect
-personal_info: PersonalInfo *# be careful with Employee inheritance*
+fill_project(team: Team, *args) -> None *# abstract factory method* *+create_project(*args) -> Project*

5. Implement concrete **WebArchitect**(***SoftwareArchitect***), **MobileArchitect**(***SoftwareArchitect***) and **EmbeddedArchitect**(***SoftwareArchitect***) classes which should override ***create_project***() method. Overridden method will return a concrete project depending on the architect.

6. Add unit testing via pytest framework.

❖ **Abstract Factory pattern.**

Can be considered as the evolution of the Factory Method. Let's enlarge our Factory Method structure into the Abstract Factory. For now, you must modify classes and create a family of products: Team + concrete project (Web, Mobile and Embedded). You can modify **Team** from the data class (class container) into ordinary class. Don't forget to implement unit testing. Explain your code.

References:

1. https://www.geeksforgeeks.org/defaultdict-in-python/.
2. https://docs.python.org/3.8/library/unittest.html
3. https://pythonworld.ru/moduli/modul-unittest.html
4. https://realpython.com/python-testing/
5. *Eric Freeman and others.* Design Patterns (Head First).
6. *Sean Bradley*. Design Patterns In Python. Common GoF (Gang of Four) Design Patterns Implemented In Python.
7. *Robert Martin*. Clean Code: A Handbook of Agile Software Craftsmanship.
8. *Mark Lutz*. Python, 5 ed.
9. *Dan Bader*. Clean Python.
10. https://refactoring.guru/