CS4402 Learning Journal 6

The Learning Journal is a tool for self-reflection on the learning process. In addition to completing directed tasks, you should use the Learning Journal to document your activities, record problems you may have encountered and to draft answers for Discussion Forums and Assignments. The Learning Journal should be updated regularly (on a weekly basis), as the learning journals will be assessed by your instructor as part of your Final Grade.

Your learning journal entry must be a reflective statement that considers the following questions:

* Describe what you did. This does not mean that you copy and paste from what you have posted or the assignments you have prepared. You need to describe what you did and how you did it.
* Describe your reactions to what you did
* Describe any feedback you received or any specific interactions you had. Discuss how they were helpful
* Describe your feelings and attitudes
* Describe what you learned

Another set of questions to consider in your learning journal statement include:

* What surprised me or caused me to wonder?
* What happened that felt particularly challenging? Why was it challenging to me?
* What skills and knowledge do I recognize that I am gaining?
* What am I realizing about myself as a learner?
* In what ways am I able to apply the ideas and concepts gained to my own experience?

Finally, describe one important thing that you are thinking about in relation to the activity.

Answer all questions that apply, however, you are expected to complete the Learning Journal assignment in 400-600 words.

During this week of learning, I explored a few object-oriented programming concepts.

There are below concepts that I have come across for object-oriented programming.

1. Objects: In OOP, everything is an object. Think of objects as real-world entities. Car objects with properties like color, speed, and behaviors like start and stop.

2. Classes: Classes are like blueprints for creating objects. If Car is a class, you can create multiple Car objects with similar characteristics and behaviors.

3. Encapsulation: This concept is like putting a protective layer around an object to hide its internal details. It's about bundling data (attributes) and methods (functions) that operate on the data into a single unit, the class.

4. Inheritance: Inheritance allows you to create a new class that's a modified version of an existing class. It promotes code reuse. For example, you can have a base class Vehicle and derive classes like Car and Motorcycle from it.

5. Polymorphism: This is about objects being able to take on many forms. In OOP, polymorphism means you can use a common interface to represent different data types or objects. For example, you can have a generic function that works with both Car and Motorcycle objects.

6. Abstraction: Abstraction is simplifying complex reality by modeling classes based on the essential properties and behaviors they share. It helps in managing the complexity of software systems.

7. Constructors and Destructors: Constructors are methods that get called when an object is created. Destructors are called when an object is destroyed. They are used for initialization and cleanup.

8. Methods and Properties: Methods are functions defined in a class, and properties are variables. For example, a Person class can have a name property and a sayHello() method.

9. Access Modifiers: These determine the visibility of methods and properties. Common access modifiers are public, private, and protected.

10. Overriding and Overloading: Overriding means providing a specific implementation for a method in a derived class, while overloading means defining multiple methods with the same name but different parameters in the same class.

The benefit of OOP are various. It includes but not limiting to the modularity and reusability.

The abstraction level and polymorphism modelling the real-world scenarios that facilitate programs are easy to understand and maintain.

On the other hand, within systems that are extreme complex, this paradigm might be come the complexity itself. Besides that, the performance overhead and it might introduce too many boiler plate codes. The OOP might not fit all the types of programming tasks. Sometime its easy to use a procedural approach for certain tasks.

Its feeling great to go back and get familiar again with object-oriented programme especially after see so many simplified codes written using scripting language. That I appreciate deeply the simplicity of programming is residing the decision-making process of choose the right stacks for the tasks.