14 Lecture - CS504

Important Subjective

Q: Define the term "Inheritance" in OOAD and explain its significance. **A**: Inheritance is a key concept in OOAD where a class (subclass) inherits properties and behaviors from another class (superclass). It promotes code reuse and allows creating specialized classes based on existing ones, leading to a more organized and extensible design. **Q: What is the purpose of the "Factory Method" design pattern in OOAD? Provide an example of its application.** **A**: The Factory Method pattern provides an interface for creating objects, allowing subclasses to decide which class to instantiate. For example, in a game, a Factory Method can be used to create different types of enemy objects based on the level difficulty, making the game design more flexible. **Q: Explain the term "Polymorphism" in OOAD and describe its advantages.** **A**: Polymorphism allows objects of different classes to be treated as objects of the same class. It enhances flexibility and extensibility by enabling a single interface to handle multiple implementations, making code more concise and maintainable. **Q: How does OOAD contribute to the development of maintainable and scalable software systems?** **A**: OOAD facilitates creating modular and reusable components, which leads to a well-organized and maintainable system. It allows developers to manage complexity, adapt to changing requirements, and easily extend the software's functionality, making it scalable over time. **Q: Discuss the importance of UML diagrams in OOAD. Provide examples of two UML diagrams and their uses.** **A**: UML diagrams help visualize system components and their relationships, aiding in better understanding and communication. For example, the Class Diagram shows the static structure of classes and their associations, while the Sequence Diagram represents the dynamic behavior of objects during interactions. **Q: What is the "Single Responsibility Principle" (SRP) in OOAD, and how does it impact software design?** **A**: SRP states that a class should have only one reason to change. It promotes cohesion and ensures that each class is responsible for a specific functionality, leading to a more maintainable and understandable codebase. **Q: Explain the concept of "Composition" in OOAD with an example.** **A**: Composition represents a strong "whole-part" relationship between classes, where the child class cannot exist independently of the parent class. For instance, a Car class composed of Engine and Wheel classes, where the Car manages the Engine and Wheels and cannot exist without them. **Q: How does OOAD help in identifying and managing software requirements effectively?** **A**: OOAD involves techniques like Use Case Analysis, where system functionalities are identified and represented as interactions between actors and the system. This helps in understanding and documenting requirements clearly, leading to better software development. **Q: Discuss the advantages of using the "Observer" design pattern in OOAD and provide a real-world scenario where it can be applied.** **A**: The Observer pattern facilitates loosely coupled communication between objects, promoting flexibility and reusability. For example, in a weather monitoring application, multiple displays can observe changes in weather data without directly depending on each other, ensuring a scalable and modular design. **Q: How does Encapsulation enhance data security and maintainability in OOAD?** **A**: Encapsulation hides the internal implementation details of a class, exposing only necessary interfaces. This protects data from direct manipulation and ensures that changes to the internal state are controlled through defined methods, reducing the risk of unintended errors and simplifying maintenance.