**Software tools individual assignment**

**By: henok teame id-1728/19**

**Part one**

**True/False Statements**

1. **False:** Boundary objects can usually communicate with both controllers and actors, but they can also interact with entity objects in certain scenarios.
2. **False:** A use case typically has clear boundaries that define its scope and the interaction between the system and actors.
3. **True:** Controllers are designed to manage interactions between boundary objects, entity objects, other controllers, and actors.
4. **False:** Entity objects primarily communicate with other entity objects or controllers, but they do not usually communicate directly with boundary objects.

**Short Answer Questions**

1. **What is a requirement?**
   * A requirement is a specification of what a system should do or a constraint on the system's operation. It defines the functionalities, performance, and attributes that a system must have to meet the needs of its users or stakeholders.
2. **Why is writing use cases an iterative process?**
   * Writing use cases is iterative because as the understanding of the system evolves, new use cases may be discovered, or existing use cases may need to be refined. Iteration allows for continuous improvement and adaptation to ensure that all user interactions are accurately captured.
3. **Explain actor, goal, and scenario:**
   * **Actor:** An actor represents any entity (human or system) that interacts with the system. Actors can be users, external systems, or devices.
   * **Goal:** A goal is the objective or outcome that the actor wants to achieve through their interaction with the system.
   * **Scenario:** A scenario is a specific sequence of actions and interactions between the actors and the system that leads to the achievement of the actor's goal.
4. **What are the classifications of objects discovered?**
   * Objects discovered during analysis and design can be classified into:
     + **Boundary Objects:** Objects that interface with actors, such as user interfaces, APIs, or input/output devices.
     + **Entity Objects:** Objects that represent business entities or data within the system, typically corresponding to tables in a database.
     + **Control Objects:** Objects that manage the flow of control between boundary and entity objects, handling the business logic and process control.

**Part two**

**True/False Statements**

1. **False:** HTML is not a super type of XML. They are different markup languages. HTML is used for creating web pages, while XML is used for defining data structures in a flexible way.
2. **True:** XML allows us to define custom tags that are not predefined, enabling the creation of structured data specific to particular applications.
3. **False:** XML is not a scripting language. It is a markup language designed to store and transport data.
4. **False:** Domain modeling can identify multiple classes that represent various entities and their relationships within a particular domain.

**Short Answer Questions**

1. **What is XML?**
   * XML (eXtensible Markup Language) is a markup language that defines rules for encoding documents in a format that is both human-readable and machine-readable. XML is used to structure, store, and transport data across different systems.
2. **Advantages of XML?**
   * **Customizable:** XML allows users to define their own tags, making it flexible for various applications.
   * **Platform-independent:** XML can be used across different platforms and systems.
   * **Self-descriptive:** XML documents contain metadata that describes the data they hold, making them easy to understand and parse.
   * **Supports Hierarchical Structure:** XML supports nested elements, which is useful for representing complex data structures.
3. **Explain low-level design:**
   * Low-level design (LLD) focuses on the implementation details of a system. It involves designing the specific modules, classes, and functions that make up the system, defining how these components interact at a granular level. LLD translates high-level design into detailed, executable code.
4. **Explain how to identify a class:**
   * To identify a class during system design, consider the following:
     + **Identify Nouns:** Review the requirements and use cases to identify nouns, which often represent potential classes (e.g., Customer, Order).
     + **Group Similar Entities:** Look for entities with similar properties or behaviors that can be grouped together into a class.
     + **Determine Responsibilities:** Define what responsibilities the potential class should have, ensuring that each class has a single, well-defined purpose.
     + **Check for Reusability:** Consider if the class can be reused in different parts of the system, which is a good indication of a well-designed class.