**Module 4: Activity Diagrams**

1. **Purpose of Activity Diagrams**
   * Describes the **stepwise flow** of activities and actions​(Module 7 - Activity2023…).
   * Used to model workflows (organizational or computational processes) and flow between or within use cases​(Module 7 - Activity2023…).
   * Helps identify preconditions and postconditions of use cases​(Module 7 - Activity2023…).
2. **Key Elements of Activity Diagrams**
   * **Actions/Activities**: Actions are atomic executable steps; activities can be decomposed and are non-atomic​(Module 7 - Activity2023…).
   * **Transitions**: Show the control flow between actions​(Module 7 - Activity2023…).
   * **Split/Merge**: Represent decision points in the flow where control paths diverge or converge​(Module 7 - Activity2023…).
   * **Fork/Join**: Show concurrency, where multiple actions occur simultaneously​(Module 7 - Activity2023…).
   * **Swimlanes**: Structure the flow by assigning actions to actors (who does what)​(Module 7 - Activity2023…).
3. **Flow Control in Activity Diagrams**
   * **Decision Nodes**: Used for mutually exclusive conditions (e.g., [if this], [else])​(Module 7 - Activity2023…).
   * **Merge Nodes**: Reunite alternative control flows​(Module 7 - Activity2023…).
   * **Fork/Join**: Indicates parallel processing (fork for starting multiple paths, join for synchronization)​(Module 7 - Activity2023…).
   * **Termination Nodes**:
     + **Activity Final**: Ends the entire activity, including all flows​(Module 7 - Activity2023…).
     + **Flow Final**: Ends only the current flow, allowing other flows to continue​(Module 7 - Activity2023…).
4. **Pros and Cons of Activity Diagrams**
   * **Pros**:
     + Maps use case scenarios directly to actions.
     + Intuitive for procedural programmers and includes constructs for concurrency and task assignment​(Module 7 - Activity2023…).
   * **Cons**:
     + Confusion with statecharts and changing terminology between UML versions​(Module 7 - Activity2023…).
     + Limited tool support​(Module 7 - Activity2023…).

**Module 4: Sequence Diagrams**

1. **Purpose of Sequence Diagrams**
   * Used to model **temporal ordering** and interaction between actors and systems​(Module 8 - Sequence Dia…).
   * Shows the **lifetimes** of objects and the **communication** (synchronous or asynchronous) between them​(Module 8 - Sequence Dia…).
   * Represents scenarios from use cases, highlighting operations, system events, and sequential ordering of operations​(Module 8 - Sequence Dia…).
2. **Key Elements of Sequence Diagrams**
   * **Lifeline Boxes**: Represent interacting objects or actors​(Module 8 - Sequence Dia…).
   * **Messages/Operations**:
     + **Synchronous**: Sender waits for a response before continuing​(Module 8 - Sequence Dia…).
     + **Asynchronous**: Sender continues without waiting for a response​(Module 8 - Sequence Dia…).
   * **Alternative Paths/Conditions**: Show choices in the flow based on conditions​(Module 8 - Sequence Dia…).
   * **Loops/Repetitions**: Represent repeated operations within the interaction​(Module 8 - Sequence Dia…).
3. **When to Use Sequence Diagrams**
   * For **actor-system interactions**, especially during the analysis phase​(Module 8 - Sequence Dia…).
   * To show **object communication**, creation, and destruction during the design phase​(Module 8 - Sequence Dia…).
   * Can be used to model the scenarios in use cases, showing communication between actors and systems​(Module 8 - Sequence Dia…).
4. **Best Practices for Sequence Diagrams**
   * Create a sequence diagram for each scenario in a use case​(Module 8 - Sequence Dia…).
   * Focus on **high-level abstraction** and avoid getting lost in unnecessary details​(Module 8 - Sequence Dia…).
5. **Pros and Cons of Sequence Diagrams**
   * **Pros**: Effective for illustrating the flow of operations and events between actors and systems​(Module 8 - Sequence Dia…).
   * **Cons**: It’s easy to over-focus on diagrams, leading to a lack of actual code development​(Module 8 - Sequence Dia…).