**Problem Statement No. 7**

**[A]**

An extension ladder consists of a rope, pulley, and latch for raising, lowering, and locking the extension. The latch secures the ladder in place when locked, allowing safe climbing. To release the latch, the extension must be slightly raised using the rope, after which the ladder can be freely raised or lowered. The latch makes a clacking sound when passing over the rungs and can be re-engaged while raising by reversing direction at the moment it passes a rung.

\*] Task: Construct a state diagram representing the behavior of the extension ladder, showing different states (Locked, Unlocked, Raising, Lowering, Latch Engaging) and the transitions between them.

**State Diagram Explanation :**

A **State Diagram** (or **State Machine Diagram**) is a type of **behavioral UML diagram** that describes the different **states** an object can be in and how it transitions between those states. It is useful for modeling **sequential processes** where the system behaves differently based on its current state.

Each **state** represents a condition in which the system exists for a period of time. **Transitions** occur between states based on specific **events** or **conditions**.

**State Diagram for an Extension Ladder**

**Understanding the Problem**

An **extension ladder** has a **rope, pulley, and latch** system that allows users to:  
✅ **Lock** the ladder in place.  
✅ **Unlock** it for movement.  
✅ **Raise or lower** it by pulling the rope.  
✅ **Re-engage** the latch when raising.

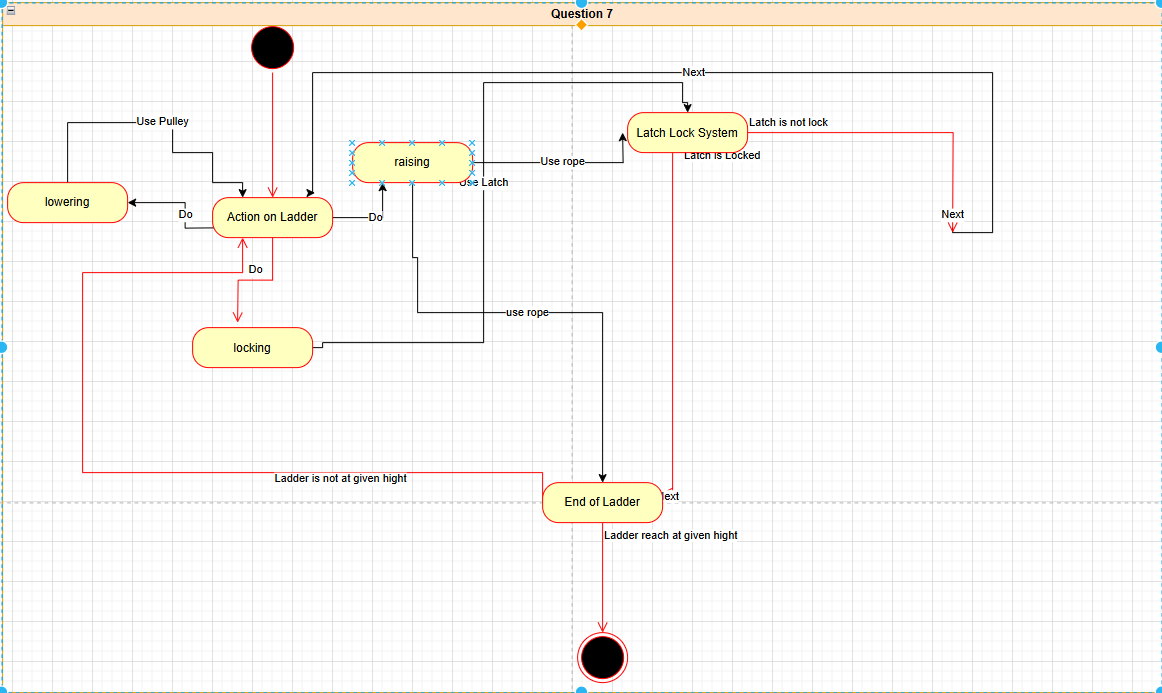
Each of these actions corresponds to a state change.

**State Explanation**

1. **Locked (Latched)**
   * The ladder is in a stable position.
   * The latch holds the extension in place.
   * The user can safely climb.
2. **Unlocked (Disengaged)**
   * The latch is released.
   * The ladder is free to move.
   * The user can raise or lower it.
3. **Raising**
   * The ladder is being **extended upwards** using the rope.
   * The latch moves past the rungs, making a clacking sound.
4. **Lowering**
   * The ladder is being **retracted downwards** using the rope.
   * The latch continues to disengage as the ladder moves.
5. **Latch Engaging**
   * The latch is **re-engaged** when passing a rung.
   * The system returns to the **Locked** state.

**Transitions Between States**

1. **Locked → Unlocked:**
   * The user **raises the extension slightly** to release the latch.
2. **Unlocked → Raising:**
   * The user **pulls the rope up** to extend the ladder.
3. **Unlocked → Lowering:**
   * The user **releases the rope** to retract the ladder.
4. **Raising → Latch Engaging:**
   * The **latch passes a rung**, allowing it to be re-engaged.
5. **Latch Engaging → Locked:**
   * The latch locks onto a rung, securing the ladder.
6. **Lowering → Locked:**
   * The ladder reaches its lowest point, and the latch secures it.

**Answer**

**Problem Statement No. 8**

**[A]**

**UML Class Diagram for a Hockey League**

A **hockey league** consists of **at least four teams**, where each **team** has **6 to 12 players**, including a **captain**. A team also has a **name and record**. **Players** have a **number and position**, and they participate in **games**. Each **game** has a **score and location**. Teams may also be led by a **coach**, who has **accreditation and years of experience** and can coach multiple teams. **Coaches and players** are **people**, and **people have names and addresses**.

**Task:** Construct a **UML Class Diagram** that represents this hockey league structure, including entities such as HockeyLeague, Team, Player, Game, Coach, and Person, along with their relationships.

**UML Class Diagram Explanation**

A **UML Class Diagram** is used to represent the **structure** of a system by showing its **classes, attributes, methods, and relationships**. It helps in understanding how different entities in a system interact with each other.

**Key Elements of a UML Class Diagram**

1. **Classes:** Represent objects with attributes (data members) and behaviors (methods).
2. **Attributes:** Characteristics or properties of a class.
3. **Associations:** Define relationships between classes (one-to-one, one-to-many, etc.).
4. **Generalization (Inheritance):** Shows parent-child relationships between classes.

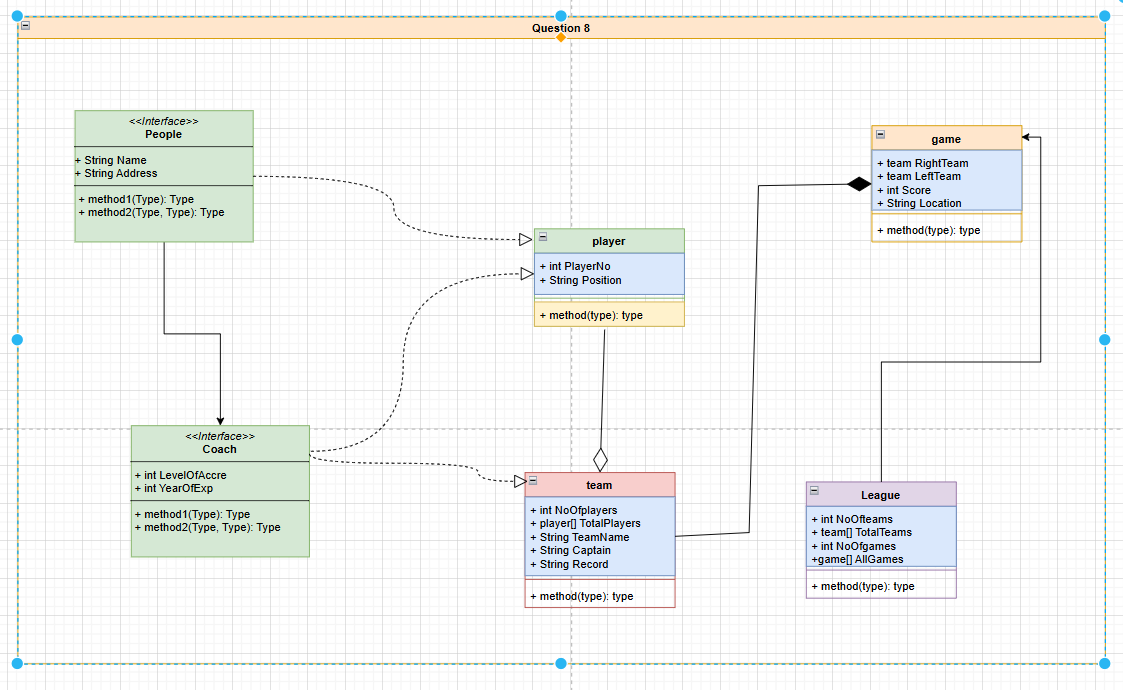
**UML Class Diagram for a Hockey League**

**Understanding the Problem**

A **hockey league** consists of multiple teams, where:  
✅ Each **team** has **6-12 players**, and **one player is the captain**.  
✅ Teams have **a name and a record**.  
✅ Players have a **number and a position**.  
✅ Hockey teams **play games** against each other.  
✅ Each game has a **score and a location**.  
✅ **Teams are sometimes led by a coach**.  
✅ A coach has **accreditation, experience, and can coach multiple teams**.  
✅ **Coaches and players are people**, and **people have names and addresses**.

**Class Definitions and Relationships**

1. **HockeyLeague**
   * Contains multiple **Teams**.
2. **Team**
   * Has **6-12 Players**.
   * Has **one Captain (Player)**.
   * Has **a record and a name**.
   * May be led by **one Coach**.
3. **Player** (inherits from Person)
   * Has **a number and a position**.
   * Participates in **Games**.
   * Can be a **captain** of a team.
4. **Game**
   * Has a **score and location**.
   * Involves **two Teams**.
5. **Coach** (inherits from Person)
   * Has **accreditation** and **years of experience**.
   * Can **coach multiple Teams**.
6. **Person (Super Class)**
   * Common attributes: **name, address**.
   * **Coach** and **Player** inherit from this.

**Answer:**