**Practical no.10**

**Abhishek Kumar Singh**

**Classes:**

1. **Identifying Classes**:
   * Review the requirements and functionalities described in the Case Study.
   * Identify nouns, verbs, and other key terms that represent entities, attributes, and actions within the system.
   * Consider the relationships and interactions between different components or entities.
2. **Classifying Classes**:
   * **Strong Classes**: These are the main entities or components that have a distinct identity and lifecycle within the system. They typically represent core functionalities or key entities.
   * **Weak Classes**: These are auxiliary or supporting classes that do not have a unique identity on their own and are dependent on strong classes for their existence. They often represent attributes or components that are part of a larger entity.
3. **Drawing the Class Diagram**:
   * Use standard UML notation to represent classes, attributes, methods, and relationships.
   * Identify associations, aggregations, compositions, and other relationships between classes.
   * Clearly define multiplicity, roles, and other constraints to specify the nature and cardinality of relationships.

**Strong Classes:**

1. **Distinct Identity**: Strong classes have a distinct identity and represent core entities or components within the system.
2. **High Cohesion**: Strong classes encapsulate related attributes and behaviors that are closely related and form a cohesive unit.
3. **Low Coupling**: Strong classes are loosely coupled with other classes, meaning they have minimal dependencies on external classes and changes to one class have minimal impact on others.
4. **Independent Lifecycle**: Strong classes have an independent lifecycle and can exist independently within the system.

**Weak Classes:**

1. **Dependent Identity**: Weak classes do not have a unique identity on their own and are dependent on strong classes for their existence.
2. **Low Cohesion**: Weak classes often represent auxiliary or supporting attributes or behaviors that are not closely related and do not form a cohesive unit on their own.
3. **High Coupling**: Weak classes are often tightly coupled with strong classes, meaning they have a high degree of dependency on external classes and changes to one class can have a significant impact on others.
4. **Dependent Lifecycle**: Weak classes have a dependent lifecycle and cannot exist independently within the system without the associated strong classes.

