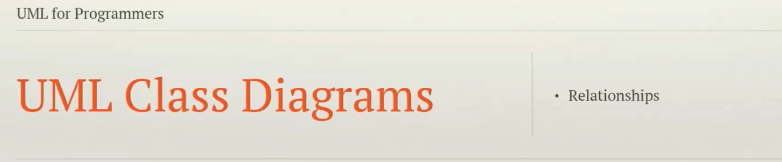
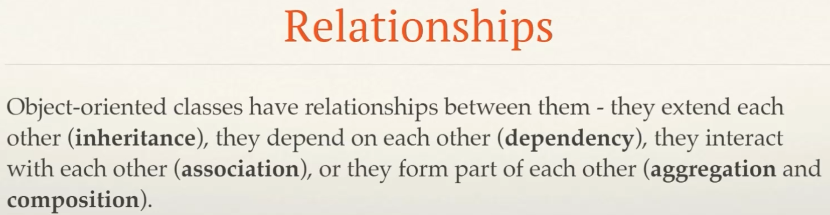
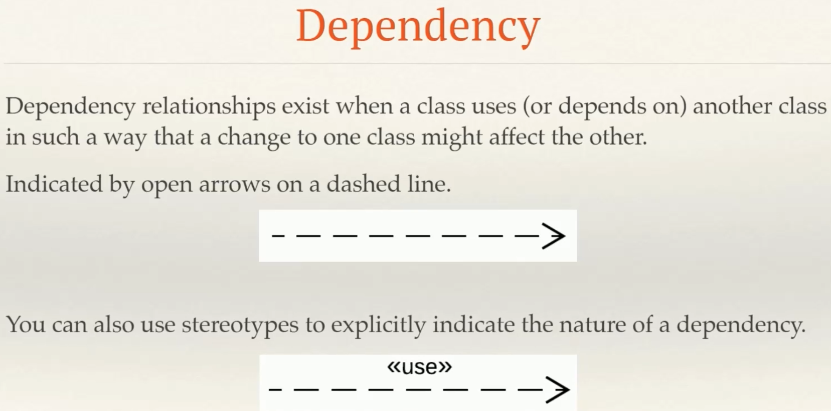
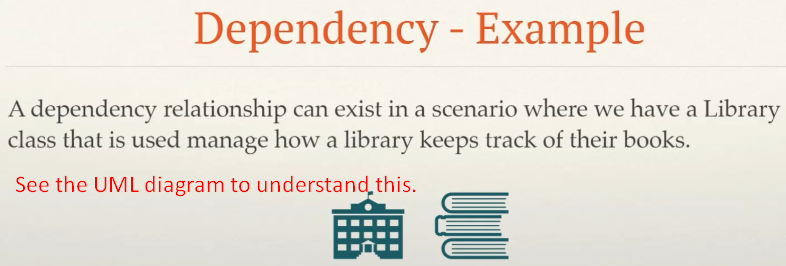
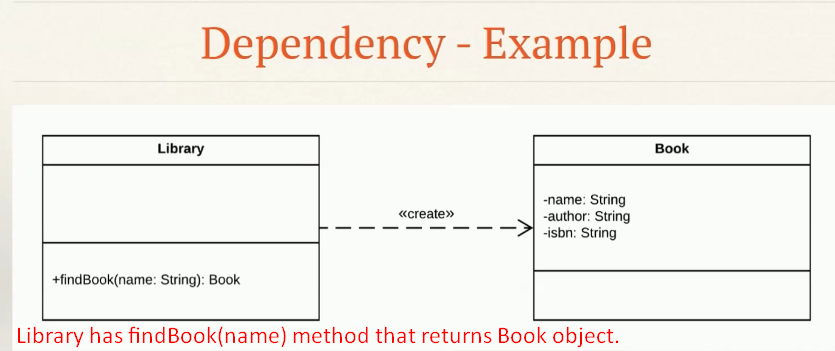
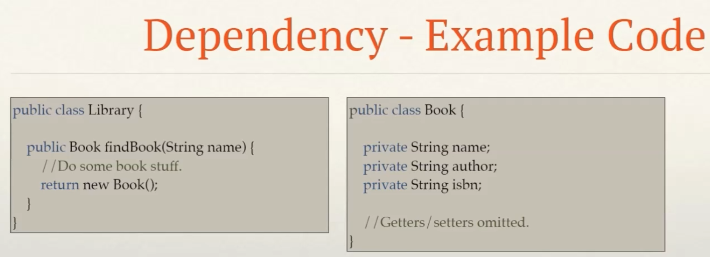
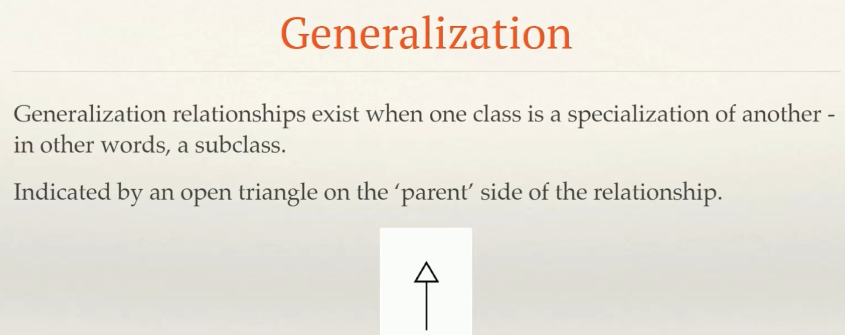
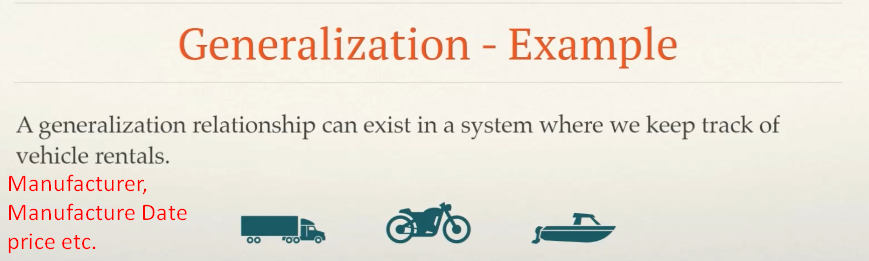
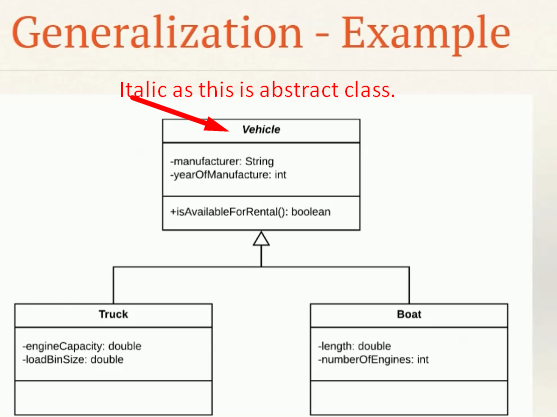
1. 
2. So far we have seen the basics of UML Class Diagrams.
3. **Agenda**:
   1. We will delve into a slightly more advanced topic - **relationship**.
      1. Relationship b/w classes &
      2. How to represent those relationships in UML.
4. Classes often don’t exist in **isolation**.   
   Many of them depend on other classes for data or operations.
5. In other words, there are relationships among classes:
   1. Classes can **extend** each other.
   2. They can **depend** on each other.
   3. They **interact** with each other.
   4. They can even form **part of** each other.
6. **Relationship:**
   1. 
7. **Arrows**:
   1. Relationships in UML are represented by a variety of arrows and each arrow represents a different relationship.
   2. These arrows are used to link different classes to each other on a Class Diagram.
8. **Multiplicity**:
   1. 
   2. We will go into details when we will start looking at the relationships having multiplicity.
9. Let’s start with **Dependency Relationship**.

Dependency Relationship

1. **Dependency relationship** exists when one class uses another class in such a way a change to one of the classes may affect the other class.  
   **NOTE**: In this relationship, class doesn’t need any property of other class but just needs reference to other class object either by method argument or class itself creates objects of other class.  
   If we take a property of other class then it becomes Association (Aggregation, Composition) Relationship.
2. **Example**:
   1. When a class calls method on other class object.
   2. Or when a class uses another class as method parameter.
   3. Or when a class creates other class object.
3. **Representation**:
   1. This relationship is represented by an open arrow on dashed line.  
      
   2. You can also be more specific and use a number of different supported stereotypes to indicate the exact nature of the dependency relationship.  
      Stereotypes are indicated within guillemets (double-angled brackets).  
      
4. 
5. 
6. 
7. **NOTE**: The findBook(name) method can change if we add a new argument constructor to Book class.

Generalization

1. In this relationship, one class is a specialization of another class - more general class.
2. 
3. 
4. 
5. 