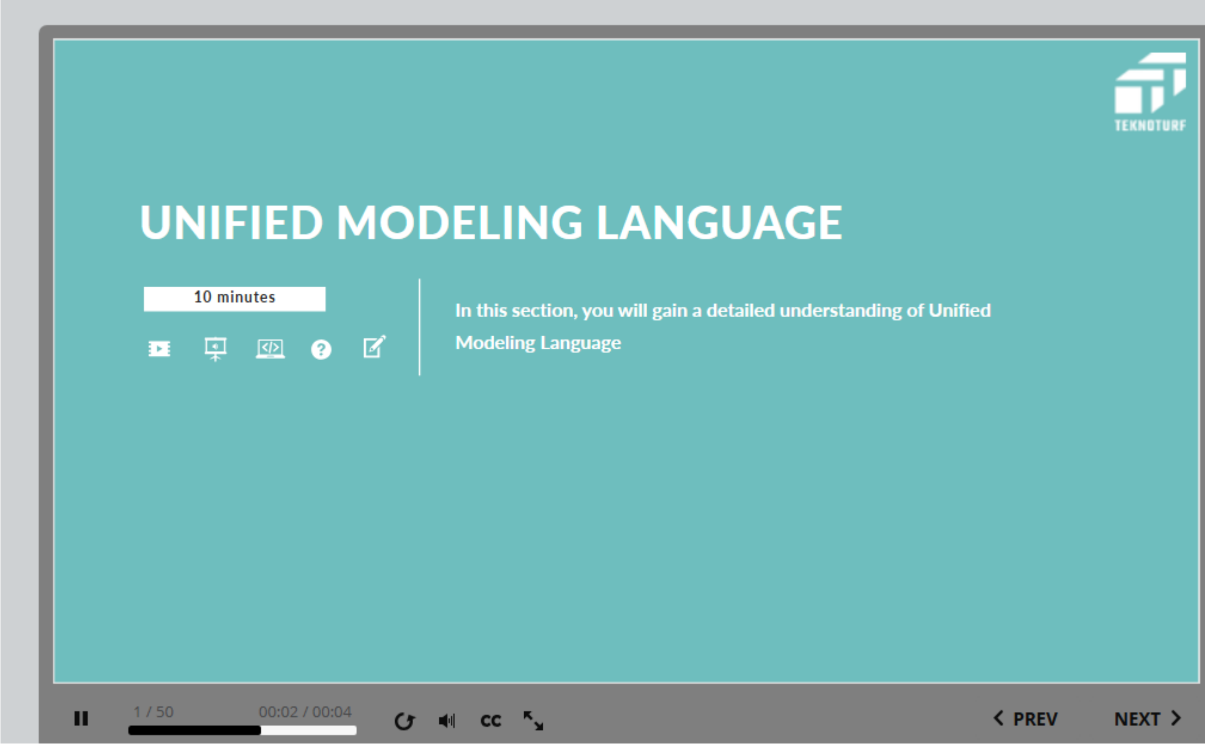
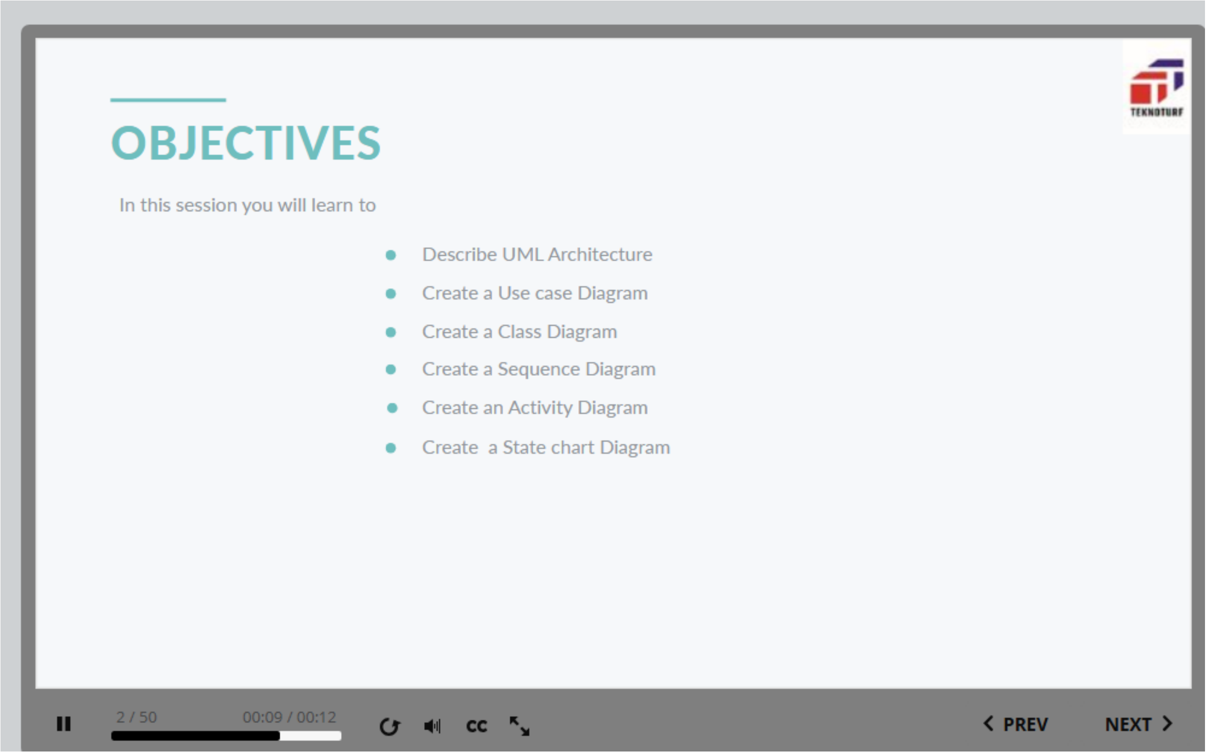
Trhudr



UNIFIED MODELING LANGUAGE

10 minutes

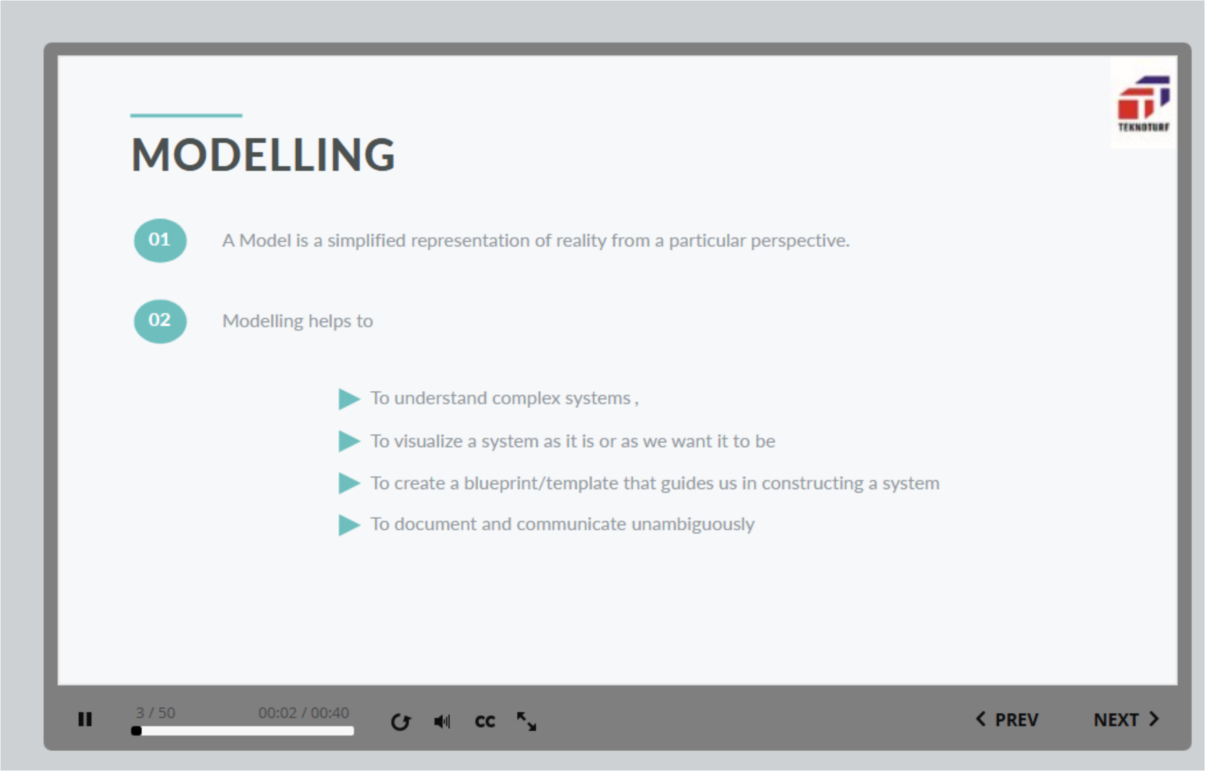
In this section, you will gain a detailed understanding of Unified Modeling Language



OBJECTIVES

In this session you will learn to

- Describe UML Architecture
- Create a Use case Diagram
- Create a Class Diagram
- Create a Sequence Diagram
- Create an Activity Diagram
- Create a State chart Diagram

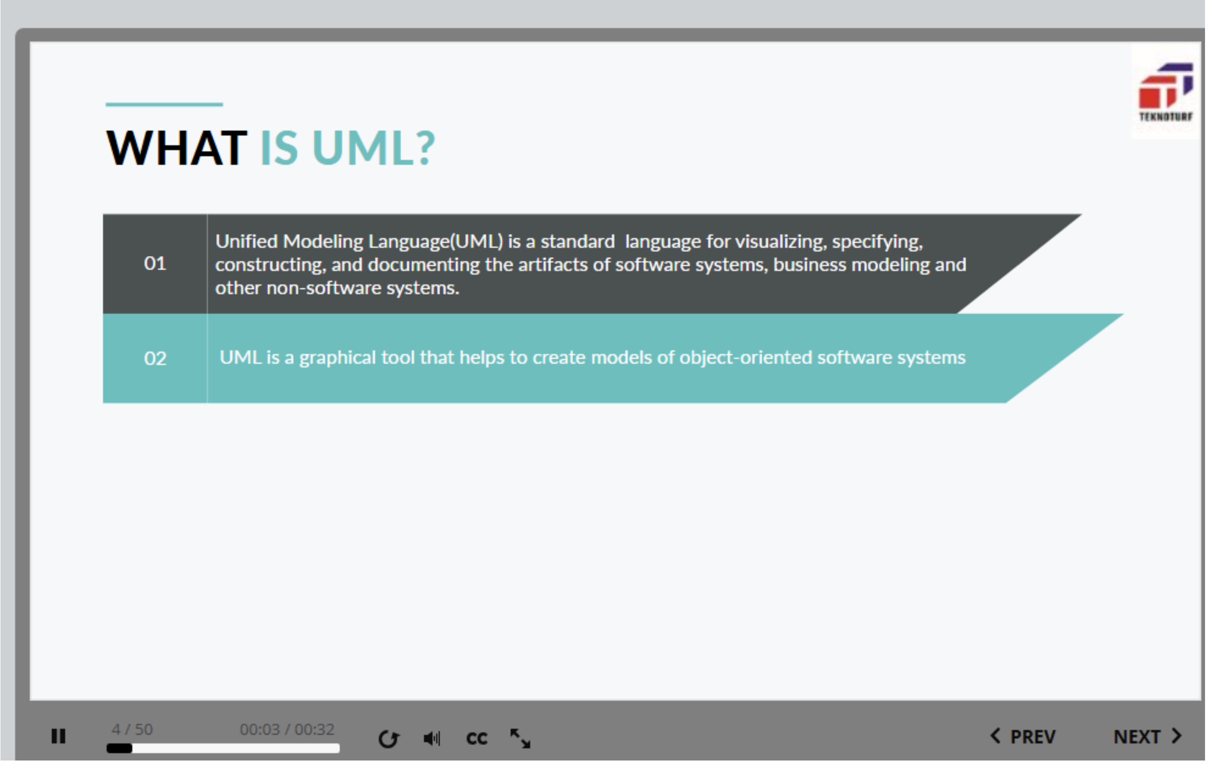


MODELLING

01 A Model is a simplified representation of reality from a particular perspective.

02 Modelling helps to

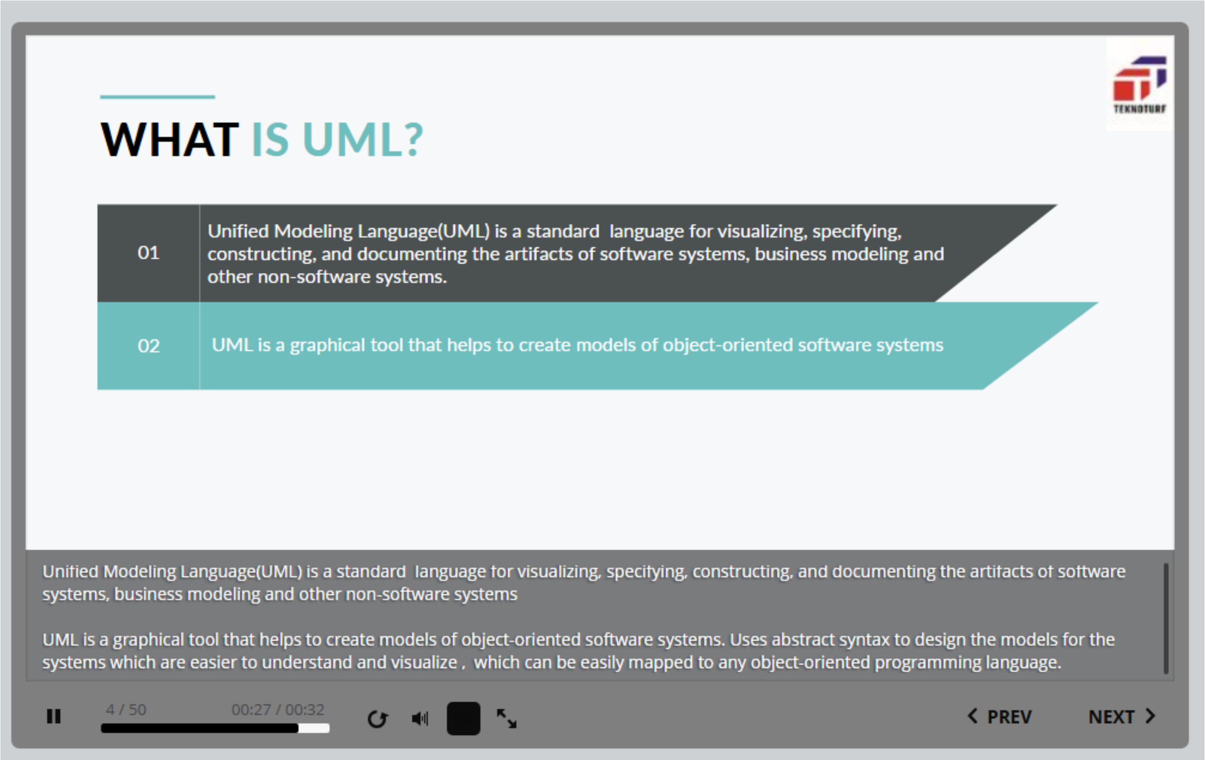
- To understand complex systems ,
- To visualize a system as it is or as we want it to be
- To create a blueprint/template that guides us in constructing a system
- To document and communicate unambiguously



WHAT IS UML?

01 Unified Modeling Language(UML) is a standard language for visualizing, specifying, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems.

02 UML is a graphical tool that helps to create models of object-oriented software systems



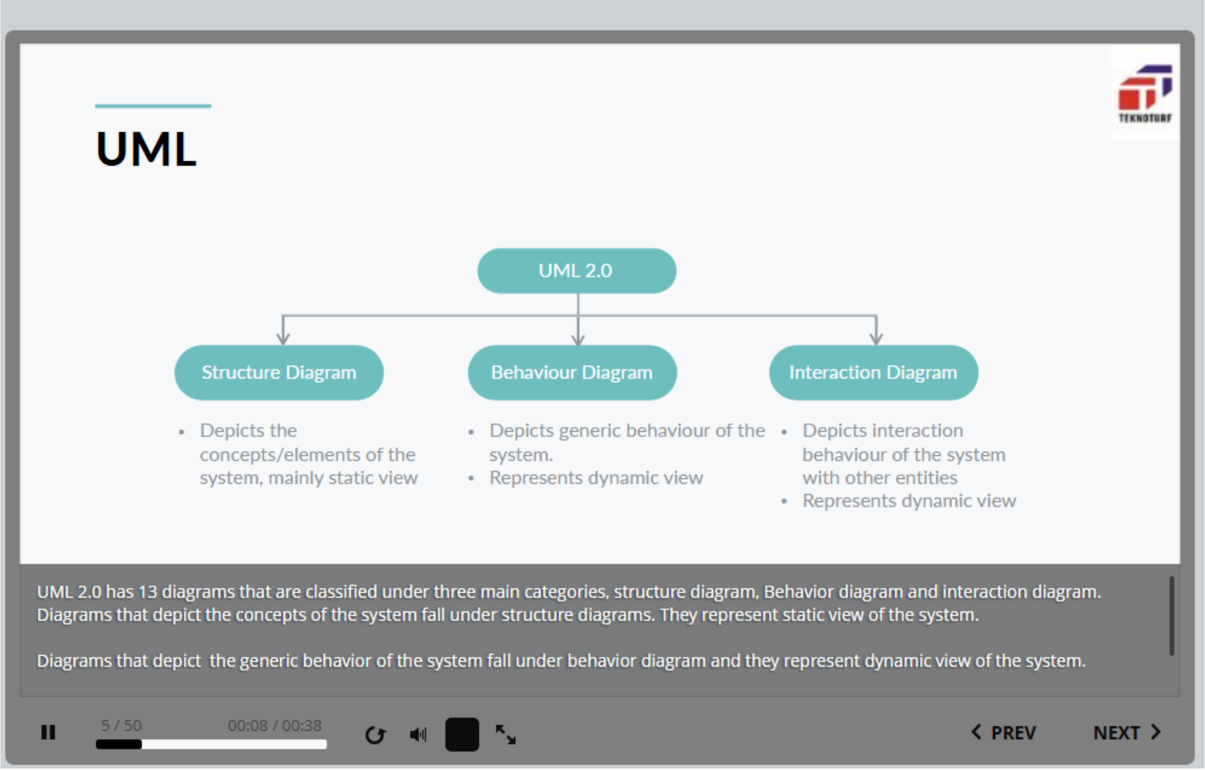
WHAT IS UML?

01 Unified Modeling Language(UML) is a standard language for visualizing, specifying, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems.

02 UML is a graphical tool that helps to create models of object-oriented software systems

Unified Modeling Language(UML) is a standard language for visualizing, specifying, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems

UML is a graphical tool that helps to create models of object-oriented software systems. Uses abstract syntax to design the models for the systems which are easier to understand and visualize , which can be easily mapped to any object-oriented programming language.



UML

UML 2.0

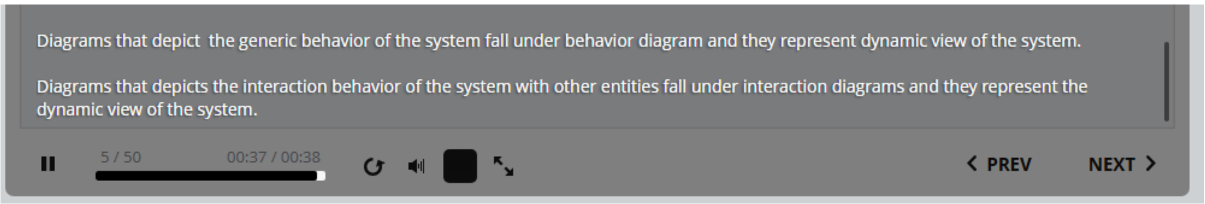
Structure Diagram
- Depicts the concepts/elements of the system, mainly static view

Behaviour Diagram
- Depicts generic behaviour of the system.
- Represents dynamic view

Interaction Diagram
- Depicts interaction behaviour of the system with other entities
- Represents dynamic view

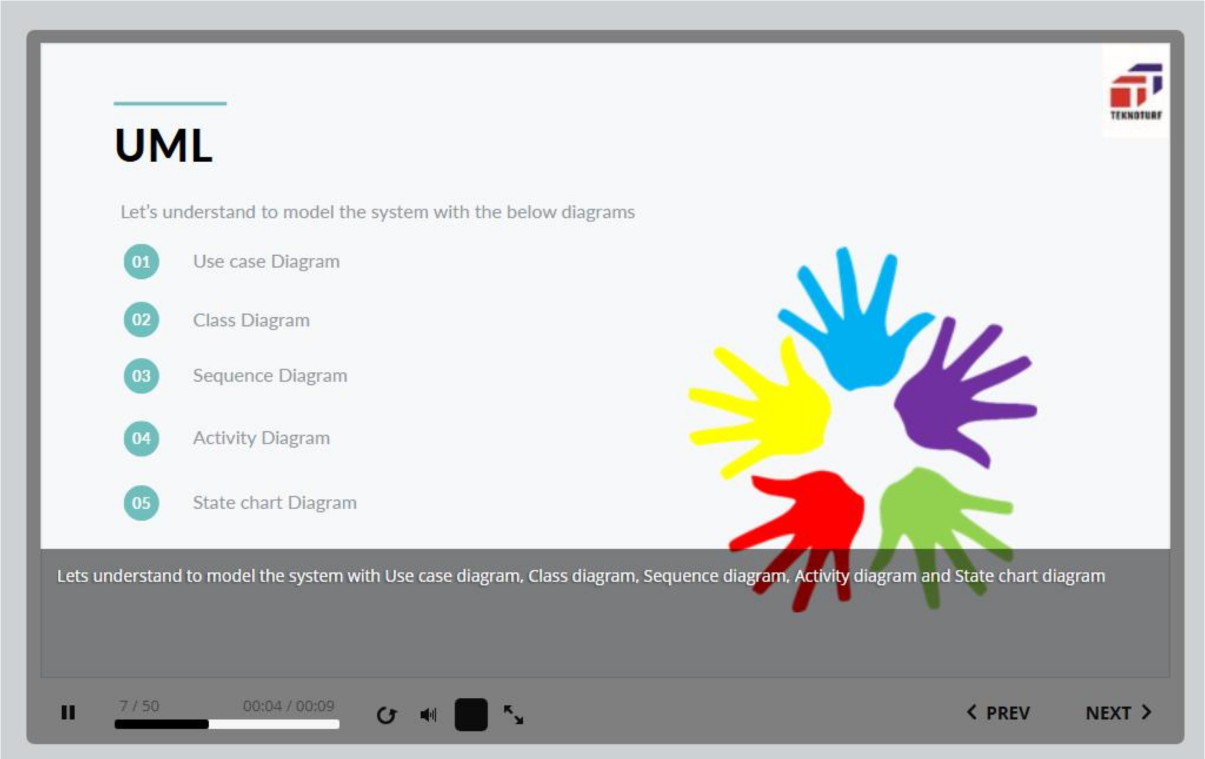
UML 2.0 has 13 diagrams that are classified under three main categories, structure diagram, Behavior diagram and interaction diagram. Diagrams that depict the concepts of the system fall under structure diagrams. They represent static view of the system.

Diagrams that depict the generic behavior of the system fall under behavior diagram and they represent dynamic view of the system.



Diagrams that depict the generic behavior of the system fall under behavior diagram and they represent dynamic view of the system.

Diagrams that depicts the interaction behavior of the system with other entities fall under interaction diagrams and they represent the dynamic view of the system.



UML

Let's understand to model the system with the below diagrams

01 Use case Diagram

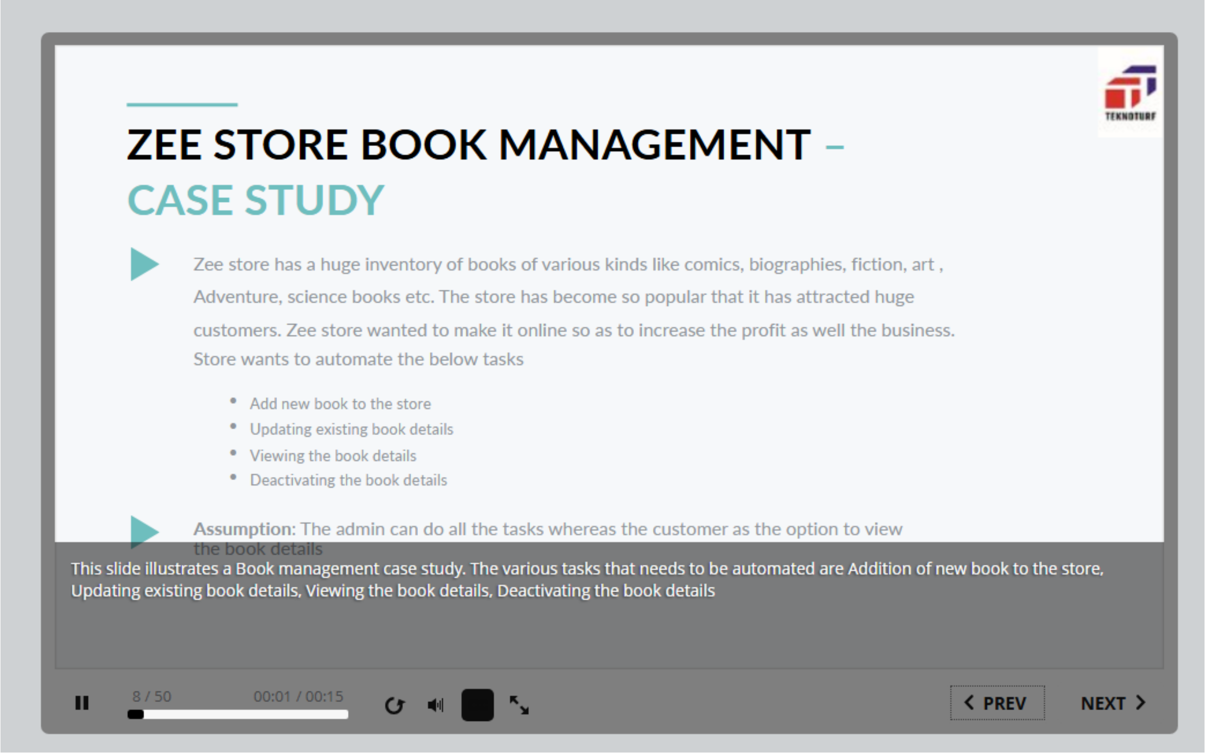
02 Class Diagram

03 Sequence Diagram

04 Activity Diagram

05 State chart Diagram

Lets understand to model the system with Use case diagram, Class diagram, Sequence diagram, Activity diagram and State chart diagram



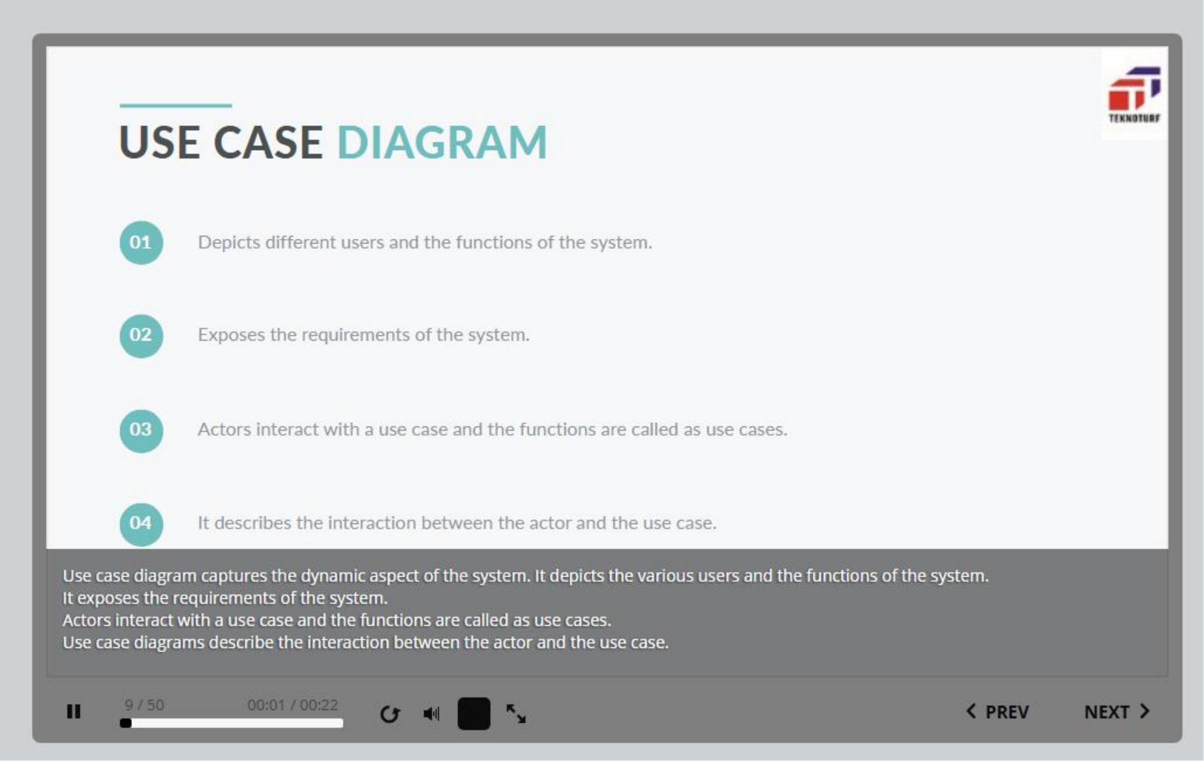
ZEE STORE BOOK MANAGEMENT – CASE STUDY

Zee store has a huge inventory of books of various kinds like comics, biographies, fiction, art , Adventure, science books etc. The store has become so popular that it has attracted huge customers. Zee store wanted to make it online so as to increase the profit as well the business. Store wants to automate the below tasks

- Add new book to the store
- Updating existing book details
- Viewing the book details
- Deactivating the book details

Assumption: The admin can do all the tasks whereas the customer as the option to view the book details

This slide illustrates a Book management case study. The various tasks that needs to be automated are Addition of new book to the store, Updating existing book details, Viewing the book details, Deactivating the book details



USE CASE DIAGRAM

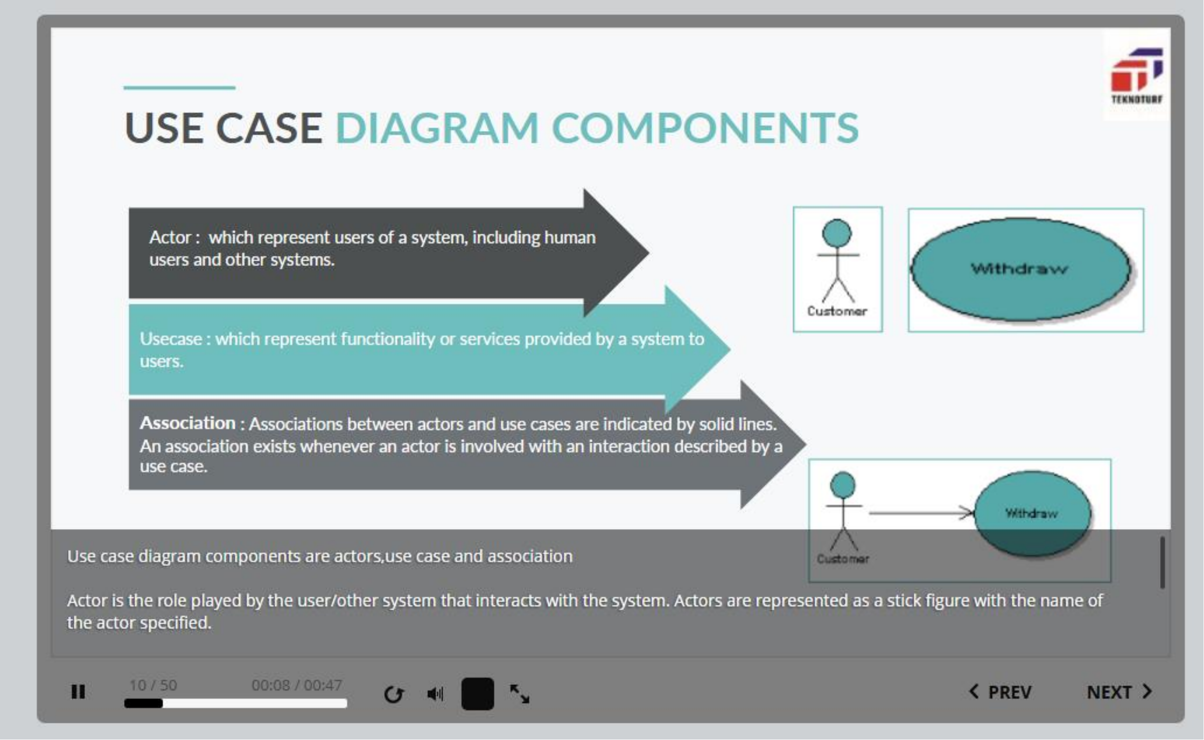
01 Depicts different users and the functions of the system.

02 Exposes the requirements of the system.

03 Actors interact with a use case and the functions are called as use cases.

04 It describes the interaction between the actor and the use case.

Use case diagram captures the dynamic aspect of the system. It depicts the various users and the functions of the system.
It exposes the requirements of the system.
Actors interact with a use case and the functions are called as use cases.
Use case diagrams describe the interaction between the actor and the use case.



USE CASE DIAGRAM COMPONENTS

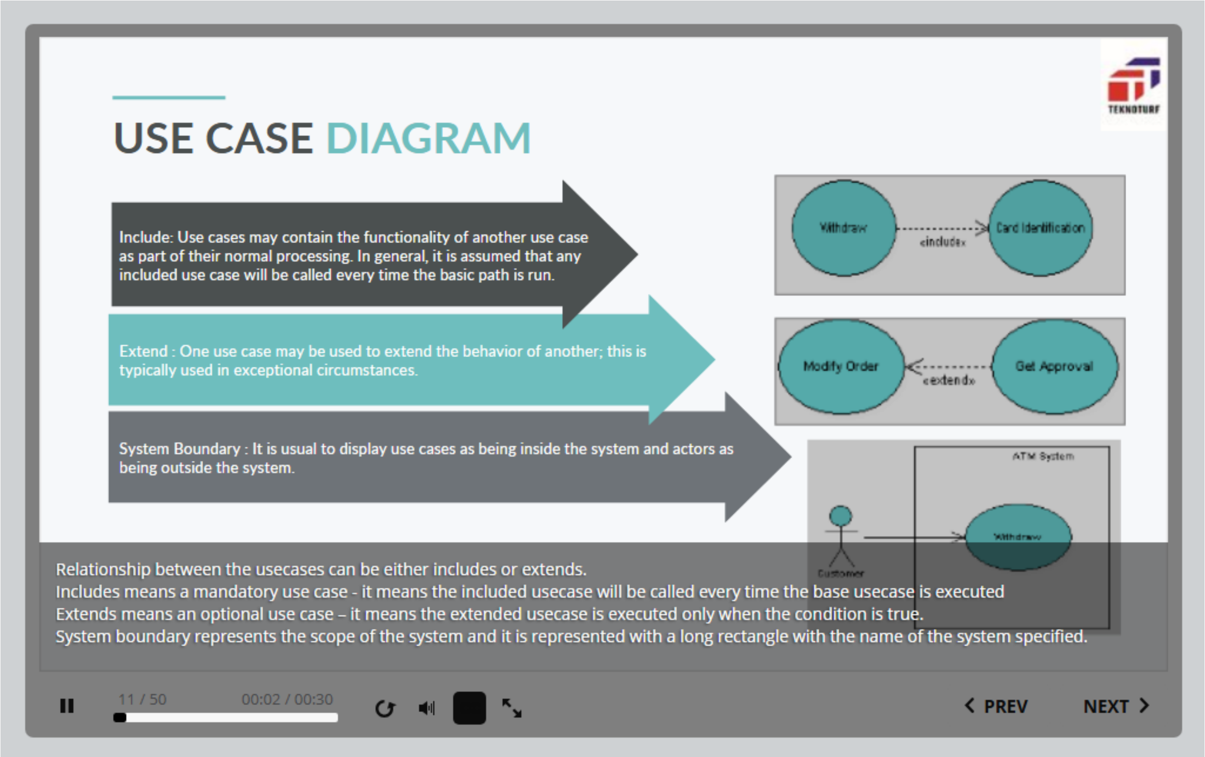
Actor : which represent users of a system, including human users and other systems.

Usecase : which represent functionality or services provided by a system to users.

Association : Associations between actors and use cases are indicated by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

Use case diagram components are actors,use case and association

Actor is the role played by the user/other system that interacts with the system. Actors are represented as a stick figure with the name of the actor specified.



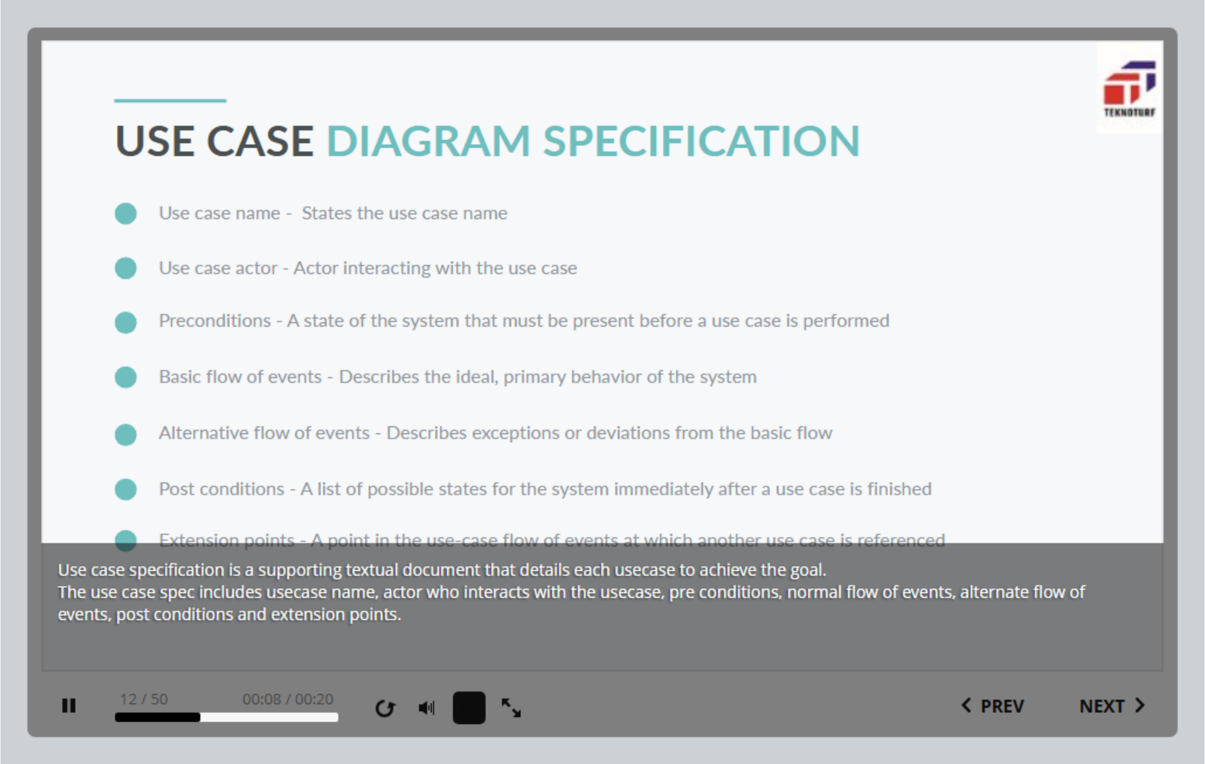
USE CASE DIAGRAM

Include: Use cases may contain the functionality of another use case as part of their normal processing. In general, it is assumed that any included use case will be called every time the basic path is run.

Extend : One use case may be used to extend the behavior of another; this is typically used in exceptional circumstances.

System Boundary : It is usual to display use cases as being inside the system and actors as being outside the system.

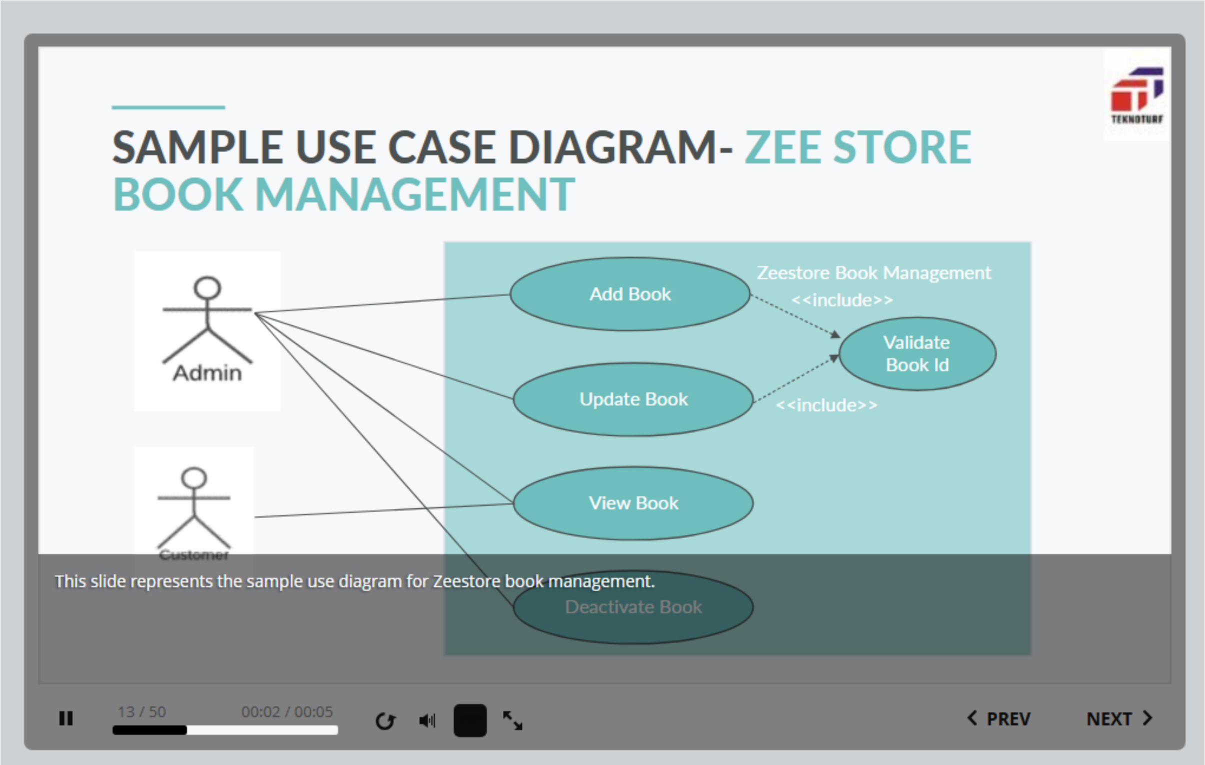
Relationship between the usecases can be either includes or extends.
Includes means a mandatory use case - it means the included usecase will be called every time the base usecase is executed
Extends means an optional use case – it means the extended usecase is executed only when the condition is true.
System boundary represents the scope of the system and it is represented with a long rectangle with the name of the system specified.



USE CASE DIAGRAM SPECIFICATION

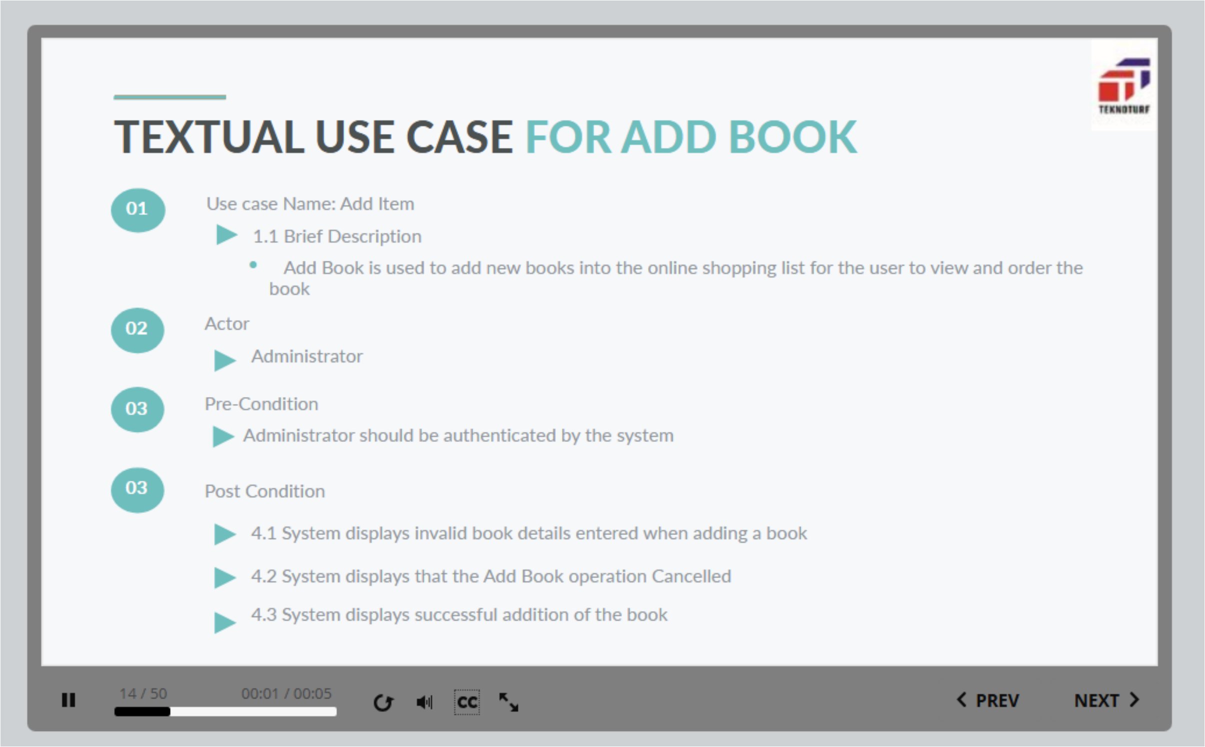
- Use case name - States the use case name
- Use case actor - Actor interacting with the use case
- Preconditions - A state of the system that must be present before a use case is performed
- Basic flow of events - Describes the ideal, primary behavior of the system
- Alternative flow of events - Describes exceptions or deviations from the basic flow
- Post conditions - A list of possible states for the system immediately after a use case is finished
- Extension points - A point in the use-case flow of events at which another use case is referenced

Use case specification is a supporting textual document that details each usecase to achieve the goal.
The use case spec includes usecase name, actor who interacts with the usecase, pre conditions, normal flow of events, alternate flow of events, post conditions and extension points.



SAMPLE USE CASE DIAGRAM- ZEE STORE BOOK MANAGEMENT

This slide represents the sample use diagram for Zeestore book management.



TEXTUAL USE CASE FOR ADD BOOK

01 Use case Name: Add Item

- 1.1 Brief Description
  - Add Book is used to add new books into the online shopping list for the user to view and order the book

02 Actor

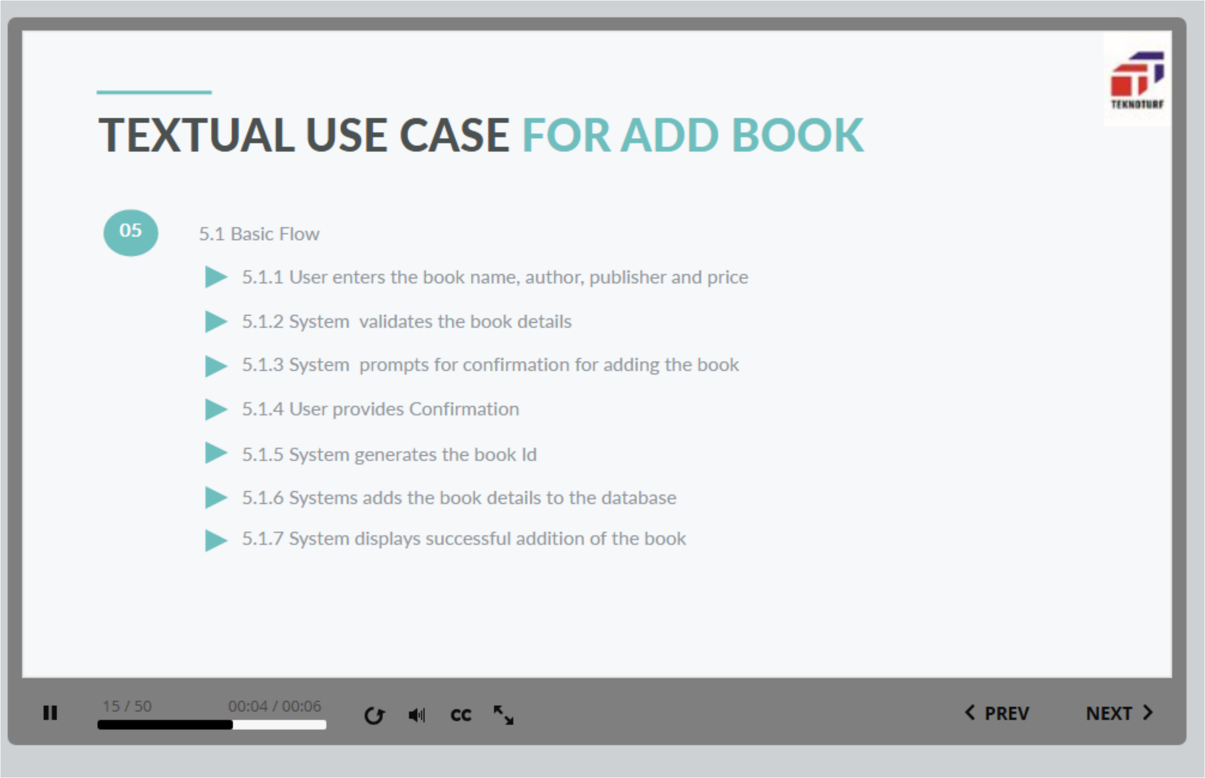
- Administrator

03 Pre-Condition

- Administrator should be authenticated by the system

03 Post Condition

- 4.1 System displays invalid book details entered when adding a book
- 4.2 System displays that the Add Book operation Cancelled
- 4.3 System displays successful addition of the book

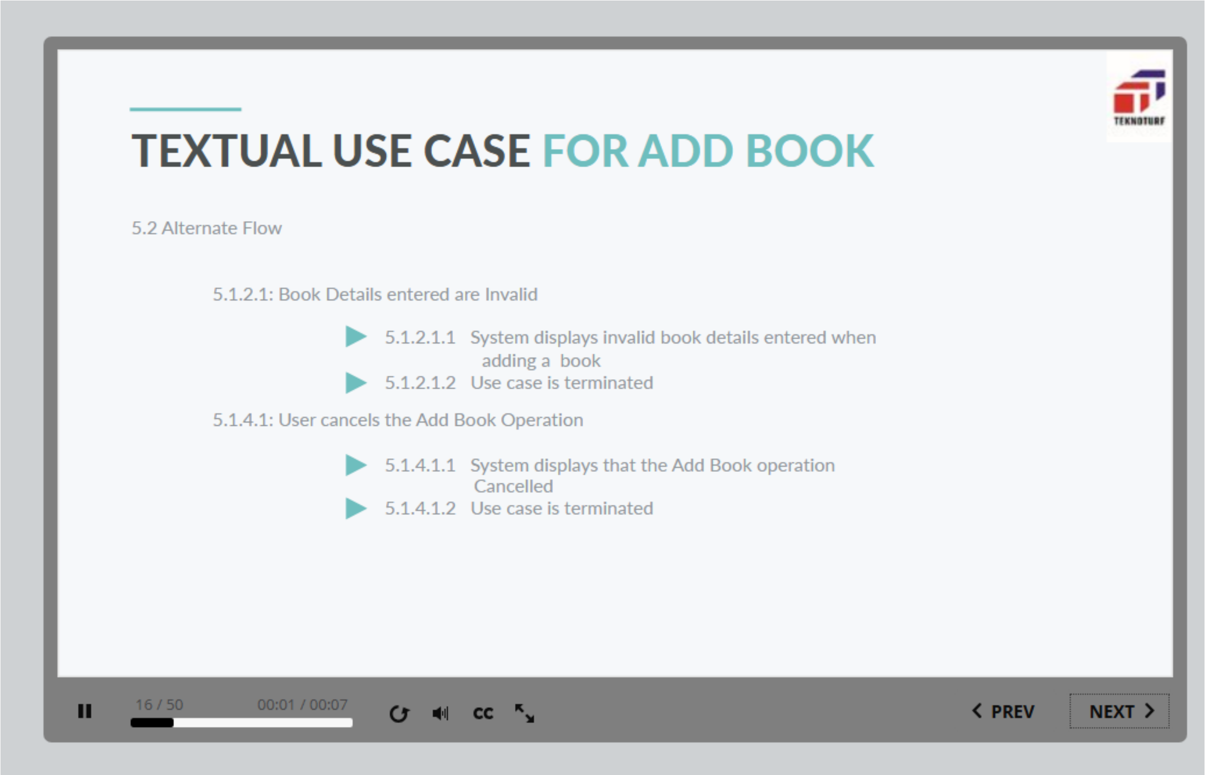


TEXTUAL USE CASE FOR ADD BOOK

05

5.1 Basic Flow

- 5.1.1 User enters the book name, author, publisher and price
- 5.1.2 System validates the book details
- 5.1.3 System prompts for confirmation for adding the book
- 5.1.4 User provides Confirmation
- 5.1.5 System generates the book Id
- 5.1.6 Systems adds the book details to the database
- 5.1.7 System displays successful addition of the book



TEXTUAL USE CASE FOR ADD BOOK

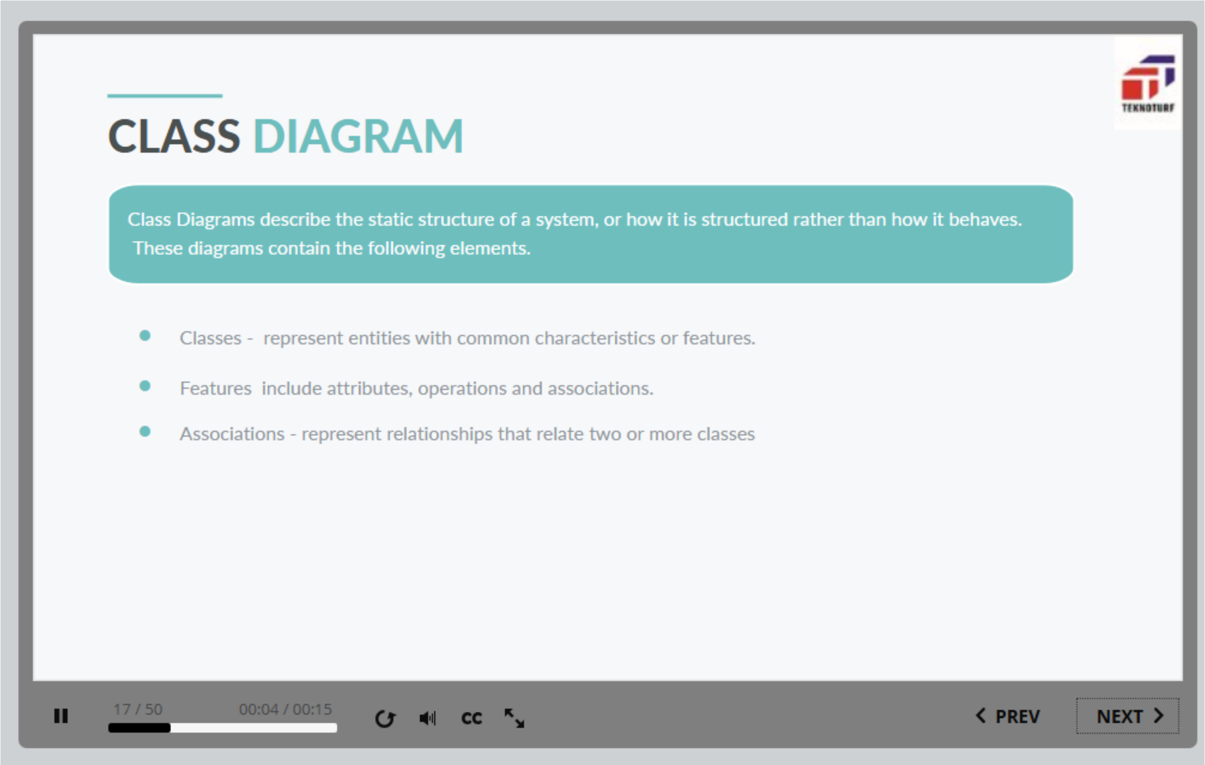
5.2 Alternate Flow

5.1.2.1: Book Details entered are Invalid

- 5.1.2.1.1 System displays invalid book details entered when adding a book
- 5.1.2.1.2 Use case is terminated

5.1.4.1: User cancels the Add Book Operation

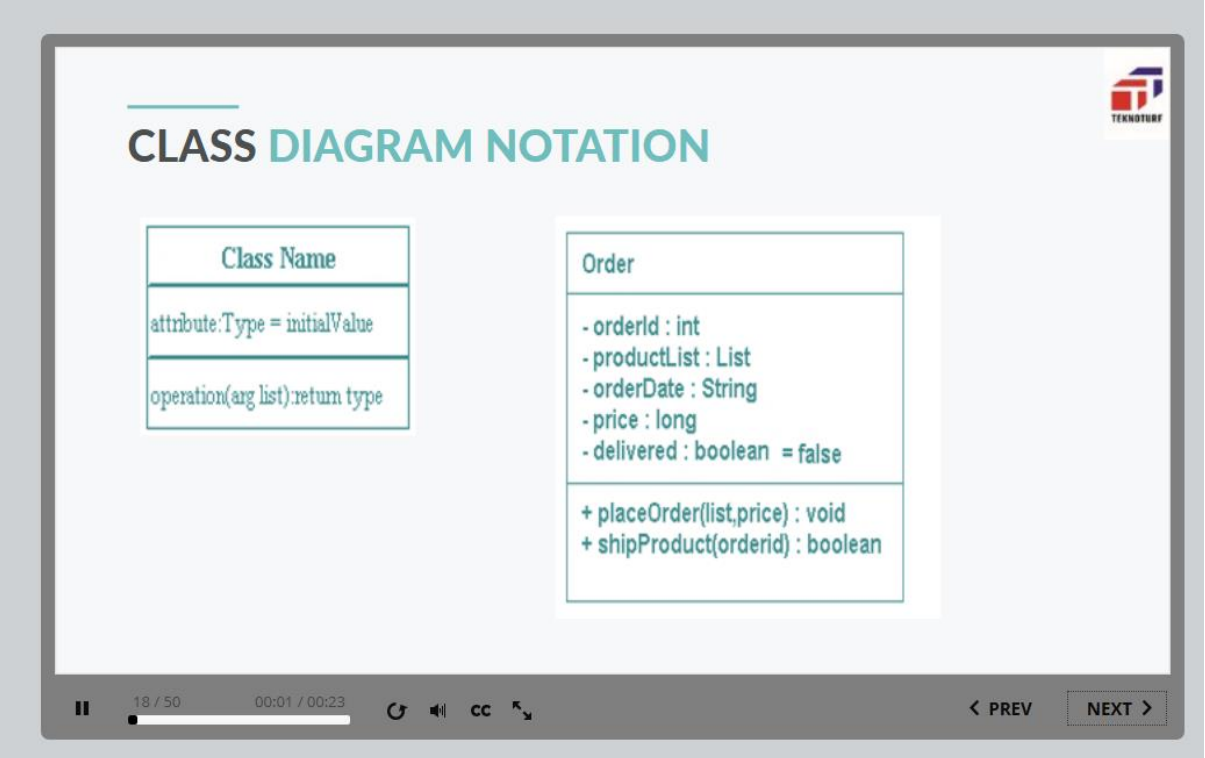
- 5.1.4.1.1 System displays that the Add Book operation Cancelled
- 5.1.4.1.2 Use case is terminated



CLASS DIAGRAM

Class Diagrams describe the static structure of a system, or how it is structured rather than how it behaves. These diagrams contain the following elements.

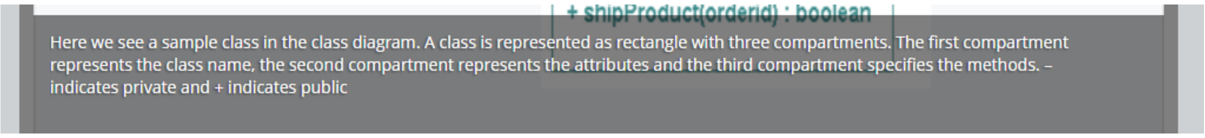
- Classes - represent entities with common characteristics or features.
- Features include attributes, operations and associations.
- Associations - represent relationships that relate two or more classes



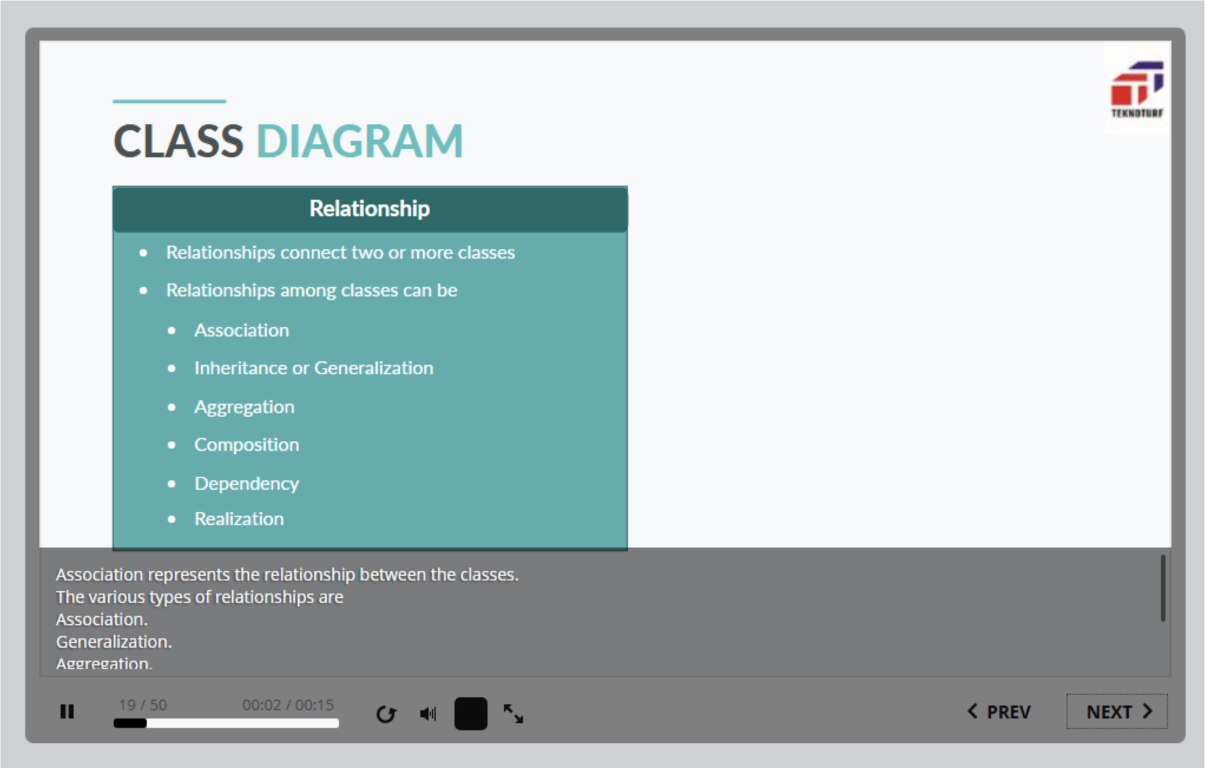
CLASS DIAGRAM NOTATION

Class Name
attribute:Type = initialValue
operation(arg list):return type

Order
- orderId : int - productList : List - orderDate : String - price : long - delivered : boolean = false
+ placeOrder(list,price) : void + shipProduct(orderid) : boolean



Here we see a sample class in the class diagram. A class is represented as rectangle with three compartments. The first compartment represents the class name, the second compartment represents the attributes and the third compartment specifies the methods. – indicates private and + indicates public

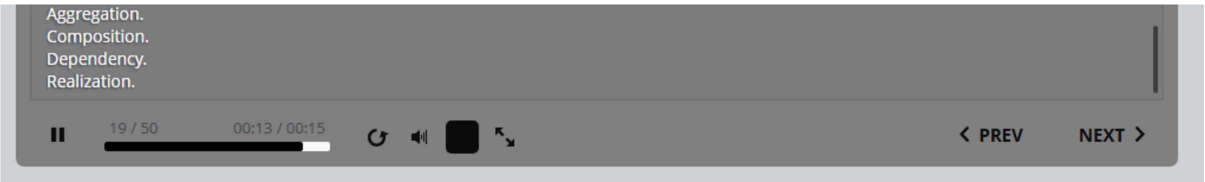


CLASS DIAGRAM

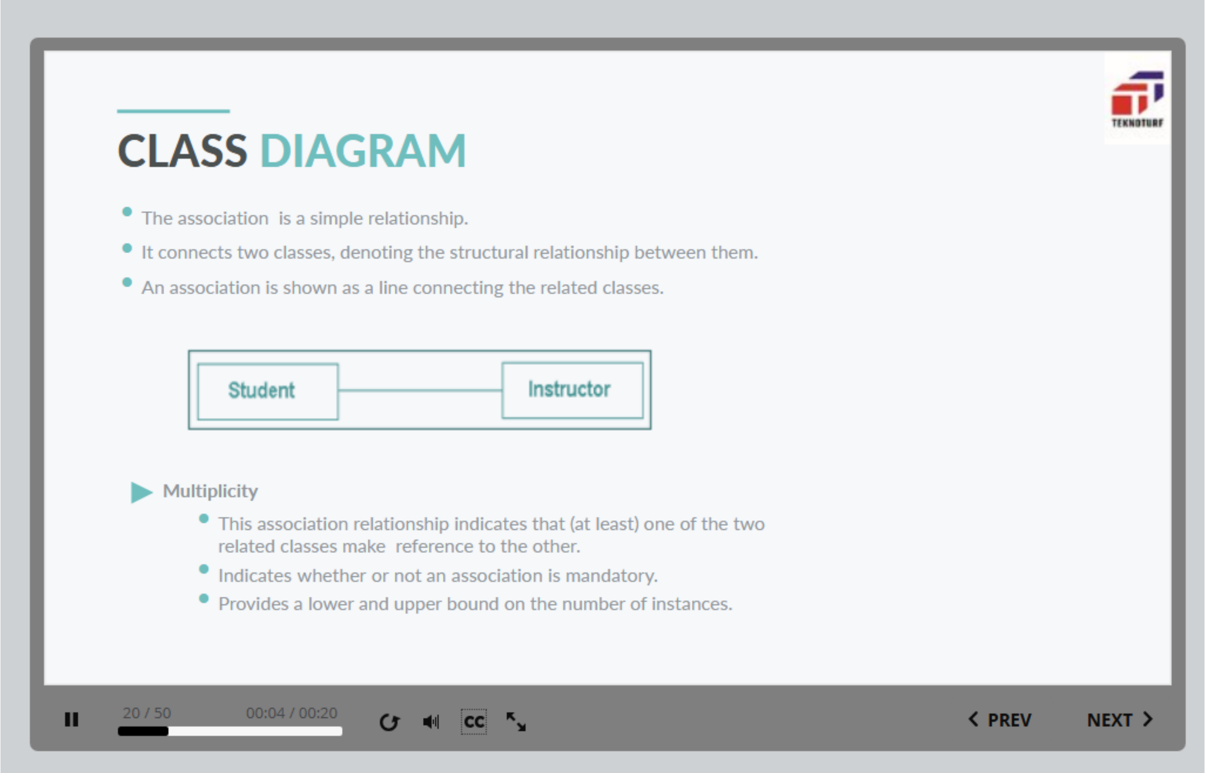
Relationship

- Relationships connect two or more classes
- Relationships among classes can be
  - Association
  - Inheritance or Generalization
  - Aggregation
  - Composition
  - Dependency
  - Realization

Association represents the relationship between the classes.
The various types of relationships are
Association.
Generalization.
Aggregation.



Aggregation.
Composition.
Dependency.
Realization.

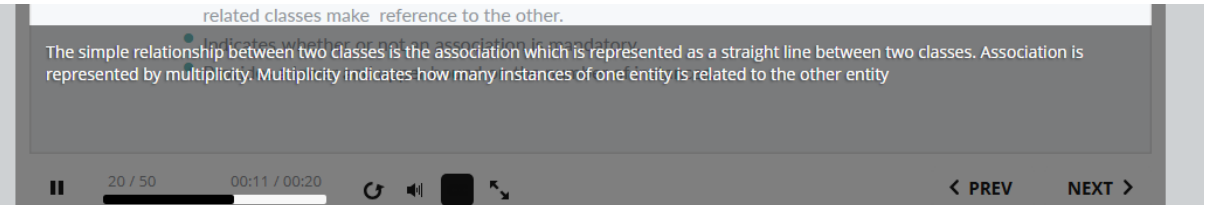


CLASS DIAGRAM

- The association is a simple relationship.
- It connects two classes, denoting the structural relationship between them.
- An association is shown as a line connecting the related classes.

▶ Multiplicity

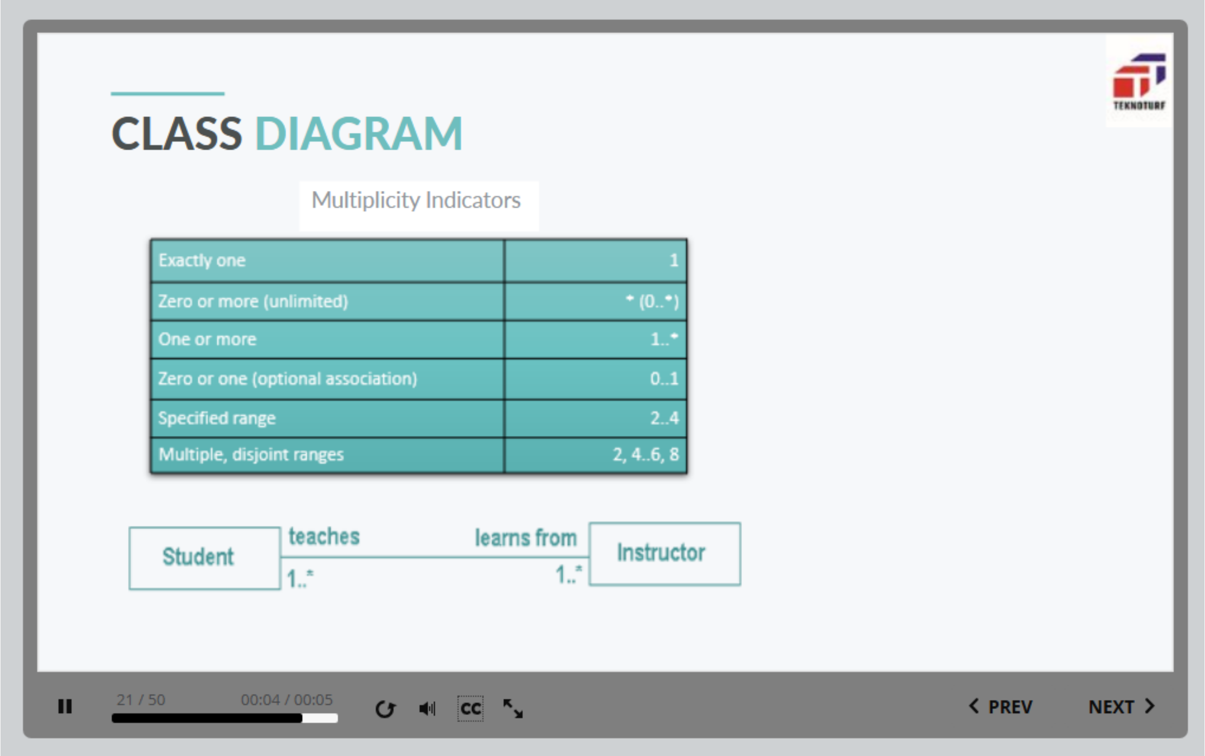
- This association relationship indicates that (at least) one of the two related classes make reference to the other.
- Indicates whether or not an association is mandatory.
- Provides a lower and upper bound on the number of instances.



related classes make reference to the other.

- Indicates whether or not an association is mandatory.

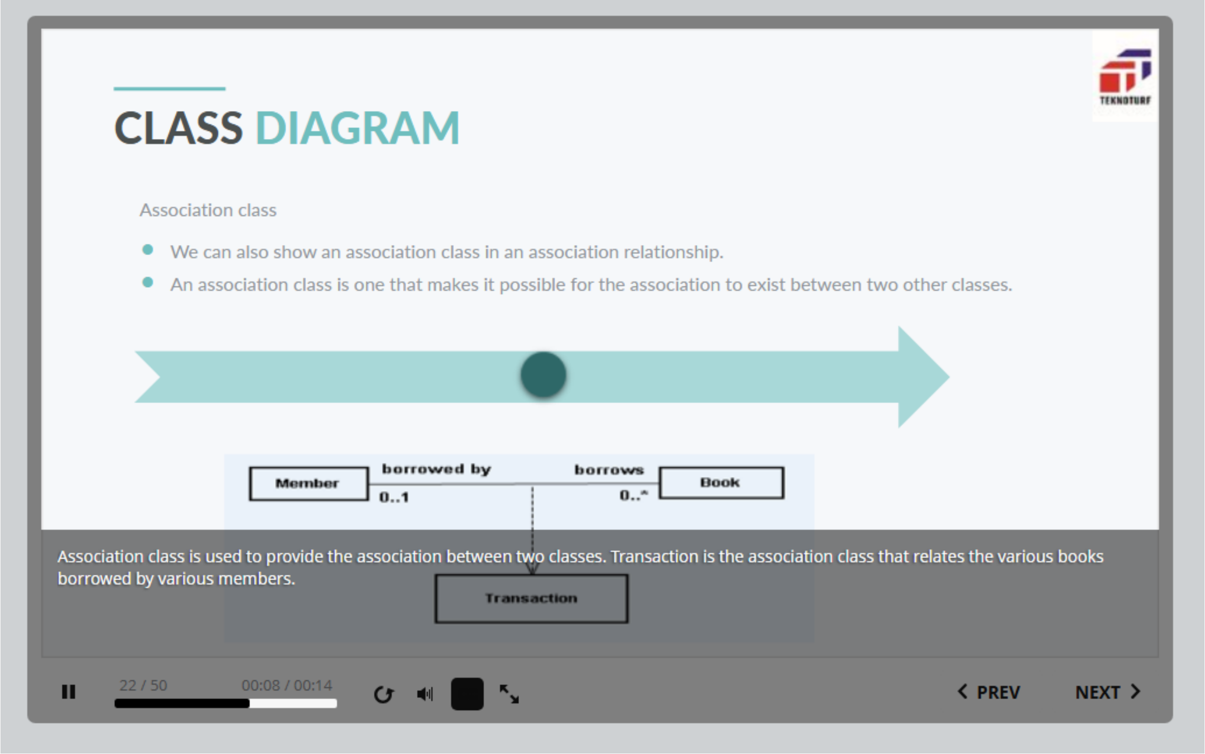
The simple relationship between two classes is the association which is represented as a straight line between two classes. Association is represented by multiplicity. Multiplicity indicates how many instances of one entity is related to the other entity



CLASS DIAGRAM

Multiplicity Indicators

Exactly one	1
Zero or more (unlimited)	* (0..*)
One or more	1..*
Zero or one (optional association)	0..1
Specified range	2..4
Multiple, disjoint ranges	2, 4..6, 8

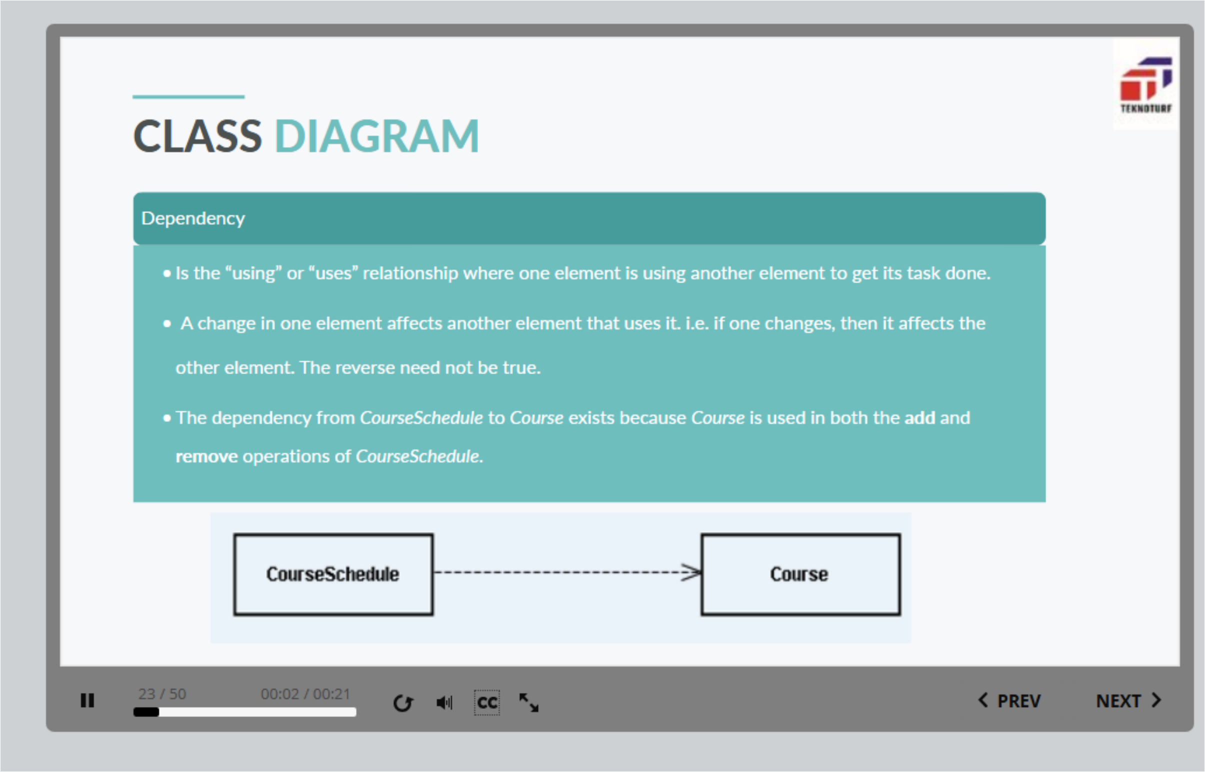


CLASS DIAGRAM

Association class

- We can also show an association class in an association relationship.
- An association class is one that makes it possible for the association to exist between two other classes.

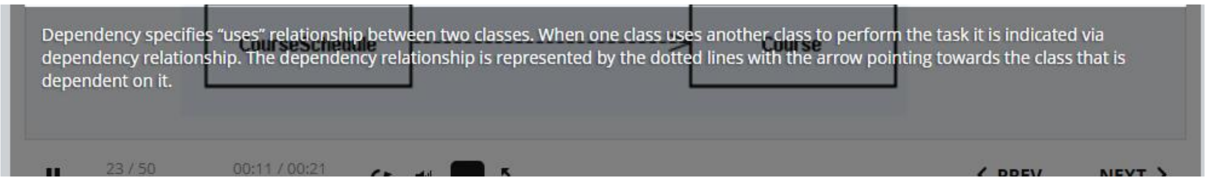
Association class is used to provide the association between two classes. Transaction is the association class that relates the various books borrowed by various members.



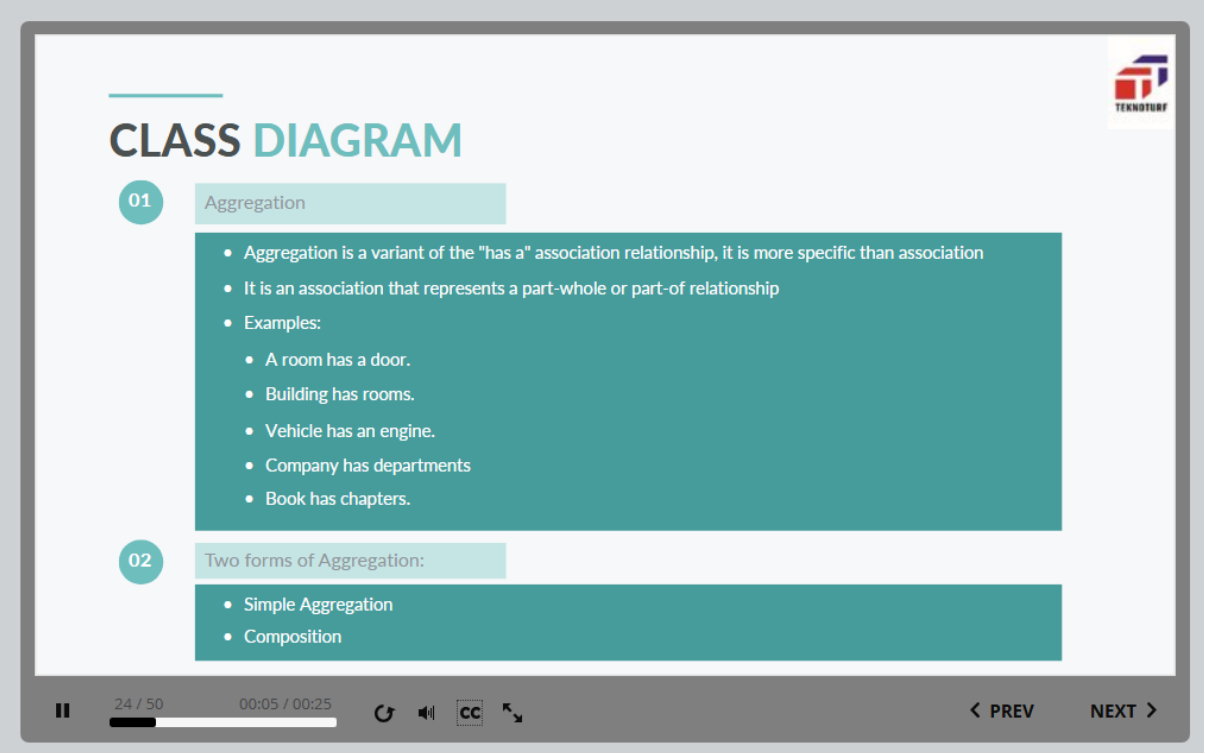
CLASS DIAGRAM

Dependency

- Is the “using” or “uses” relationship where one element is using another element to get its task done.
- A change in one element affects another element that uses it. i.e. if one changes, then it affects the other element. The reverse need not be true.
- The dependency from *CourseSchedule* to *Course* exists because *Course* is used in both the **add** and **remove** operations of *CourseSchedule*.



Dependency specifies “uses” relationship between two classes. When one class uses another class to perform the task it is indicated via dependency relationship. The dependency relationship is represented by the dotted lines with the arrow pointing towards the class that is dependent on it.



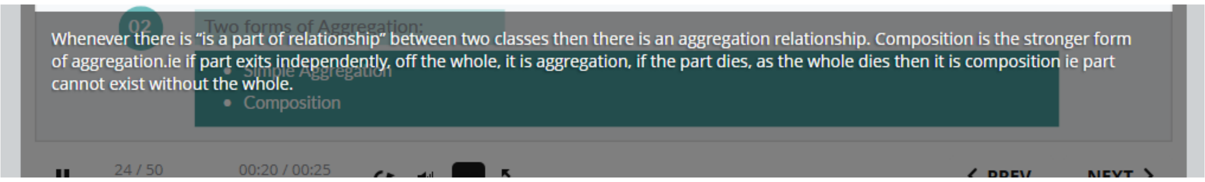
CLASS DIAGRAM

01 Aggregation

- Aggregation is a variant of the "has a" association relationship, it is more specific than association
- It is an association that represents a part-whole or part-of relationship
- Examples:
  - A room has a door.
  - Building has rooms.
  - Vehicle has an engine.
  - Company has departments
  - Book has chapters.

02 Two forms of Aggregation:

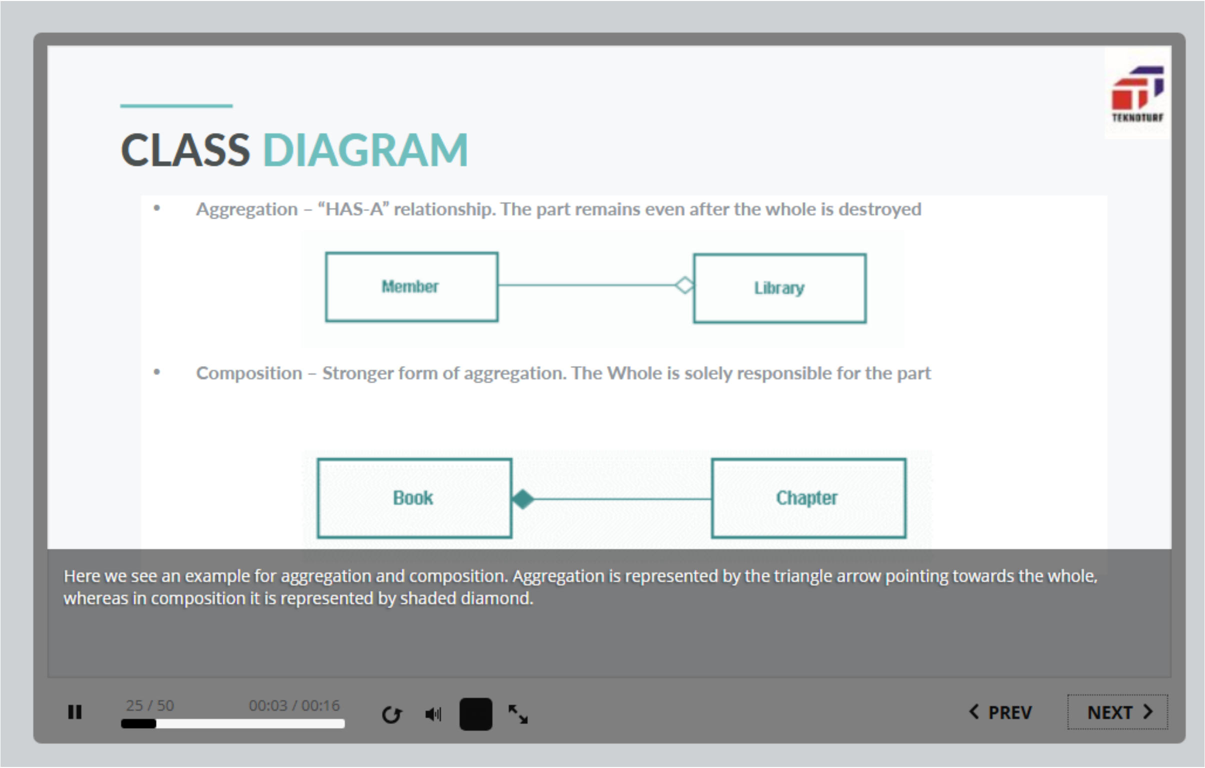
- Simple Aggregation
- Composition



Whenever there is “is a part of relationship” between two classes then there is an aggregation relationship. Composition is the stronger form of aggregation.ie if part exits independently, off the whole, it is aggregation, if the part dies, as the whole dies then it is composition ie part cannot exist without the whole.

Two forms of Aggregation:

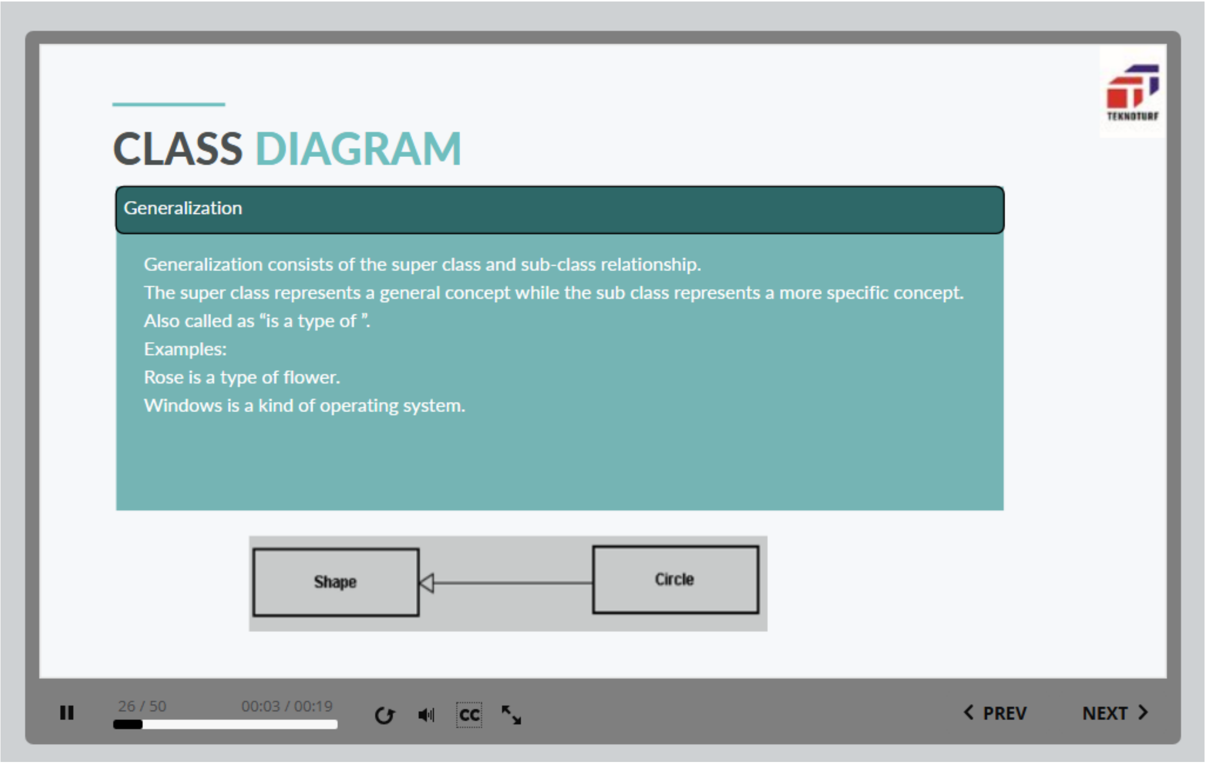
- Simple Aggregation
- Composition



CLASS DIAGRAM

- Aggregation – “HAS-A” relationship. The part remains even after the whole is destroyed
- Composition – Stronger form of aggregation. The Whole is solely responsible for the part

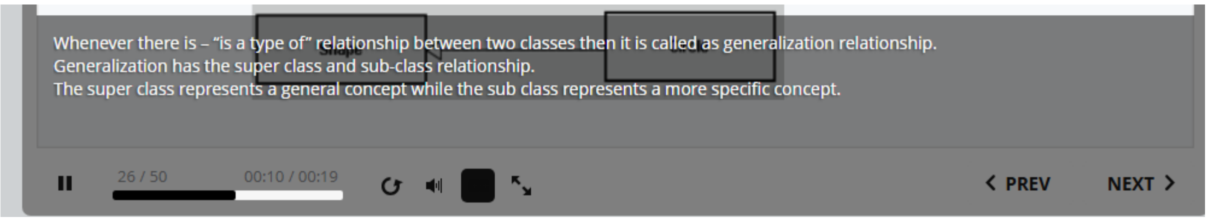
Here we see an example for aggregation and composition. Aggregation is represented by the triangle arrow pointing towards the whole, whereas in composition it is represented by shaded diamond.



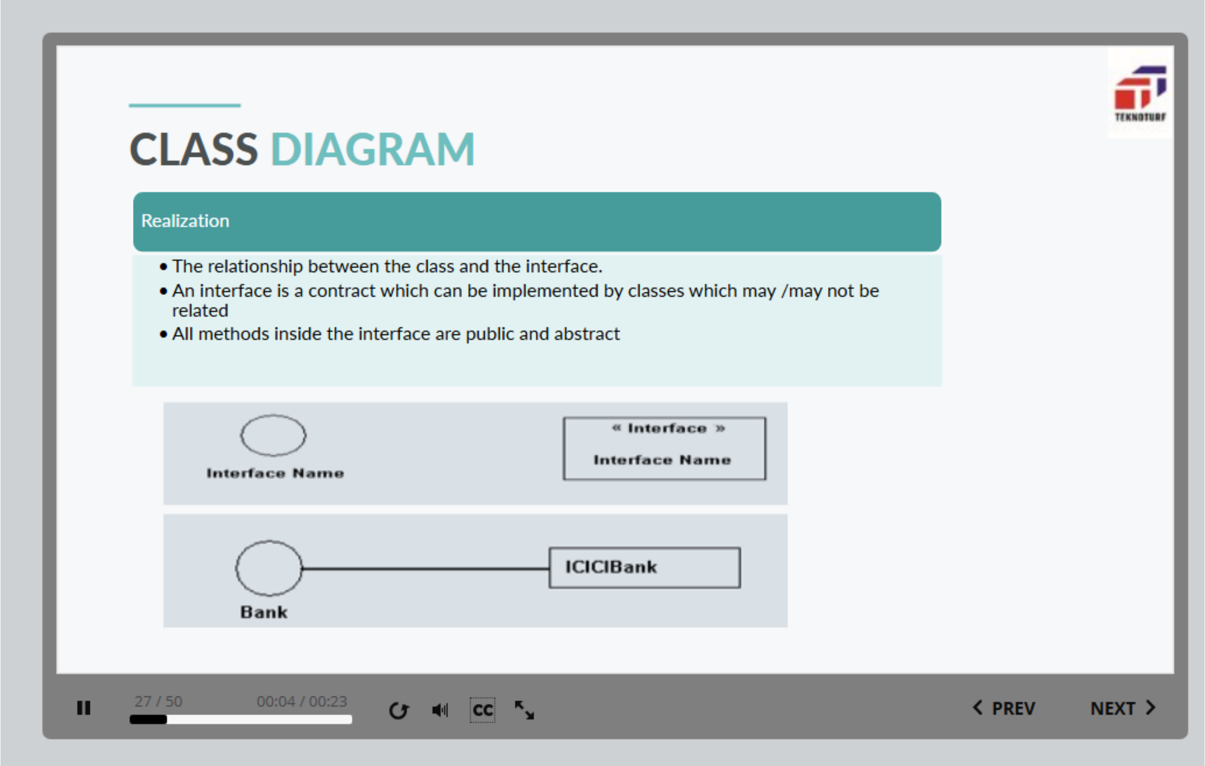
CLASS DIAGRAM

Generalization

Generalization consists of the super class and sub-class relationship.
The super class represents a general concept while the sub class represents a more specific concept.
Also called as “is a type of ”.
Examples:
Rose is a type of flower.
Windows is a kind of operating system.



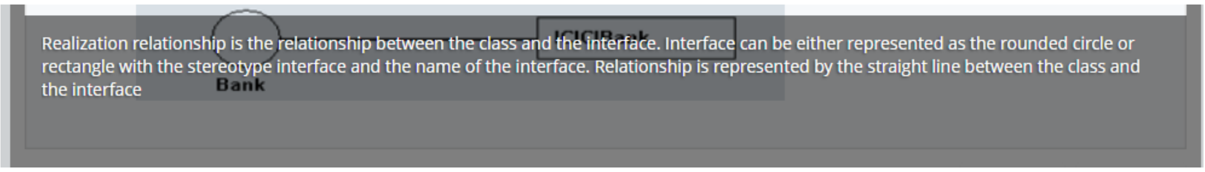
Whenever there is – “is a type of” relationship between two classes then it is called as generalization relationship.
Generalization has the super class and sub-class relationship.
The super class represents a general concept while the sub class represents a more specific concept.



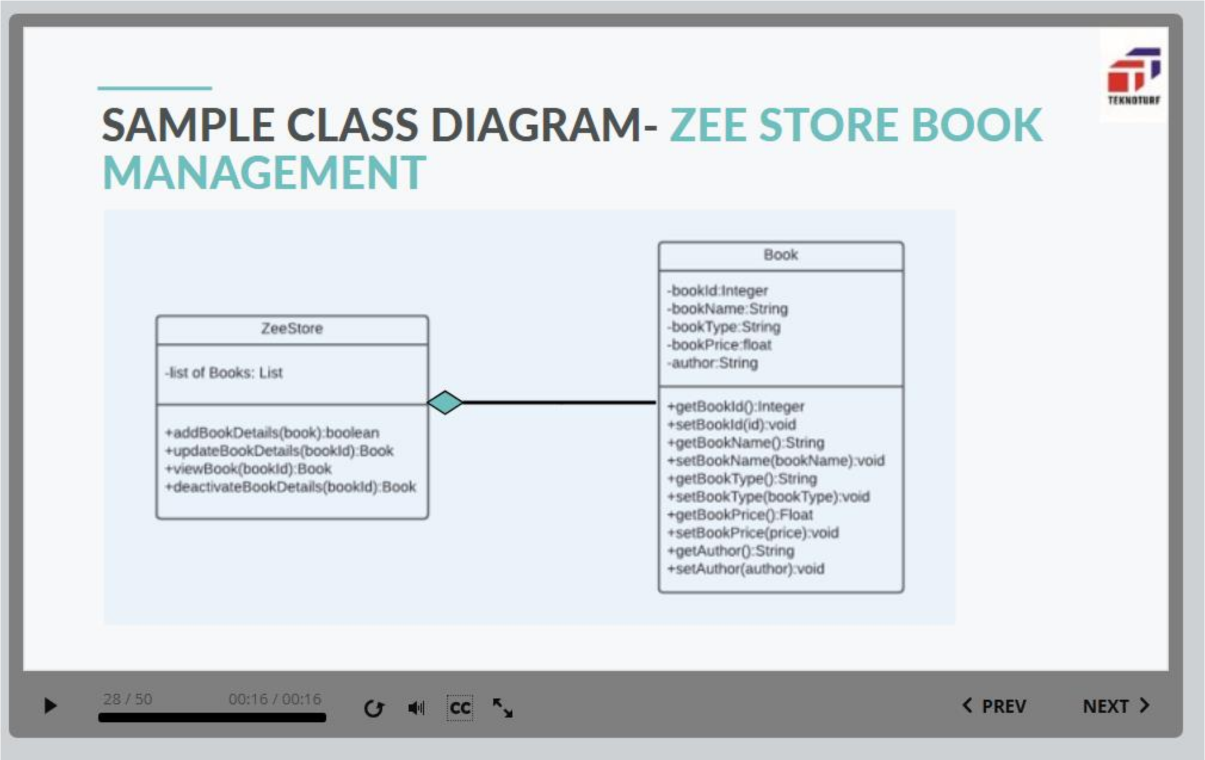
CLASS DIAGRAM

Realization

- The relationship between the class and the interface.
- An interface is a contract which can be implemented by classes which may /may not be related
- All methods inside the interface are public and abstract



Realization relationship is the relationship between the class and the interface. Interface can be either represented as the rounded circle or rectangle with the stereotype interface and the name of the interface. Relationship is represented by the straight line between the class and the interface



SAMPLE CLASS DIAGRAM- ZEE STORE BOOK MANAGEMENT

ZeeStore

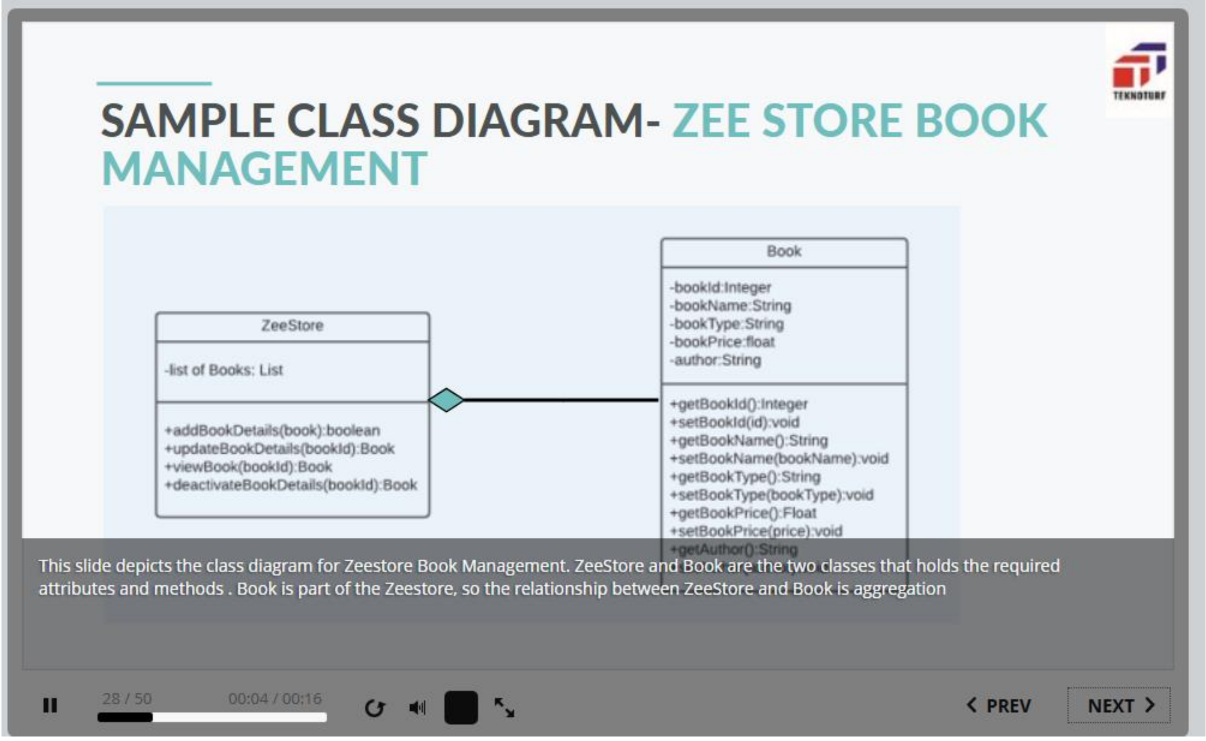
-list of Books: List

+addBookDetails(book):boolean
+updateBookDetails(bookId):Book
+viewBook(bookId):Book
+deactivateBookDetails(bookId):Book

Book

-bookId:Integer
-bookName:String
-bookType:String
-bookPrice:float
-author:String

+getBookId():Integer
+setBookId(id):void
+getBookName():String
+setBookName(bookName):void
+getBookType():String
+setBookType(bookType):void
+getBookPrice():Float
+setBookPrice(price):void
+getAuthor():String
+setAuthor(author):void



SAMPLE CLASS DIAGRAM- ZEE STORE BOOK MANAGEMENT

ZeeStore

-list of Books: List

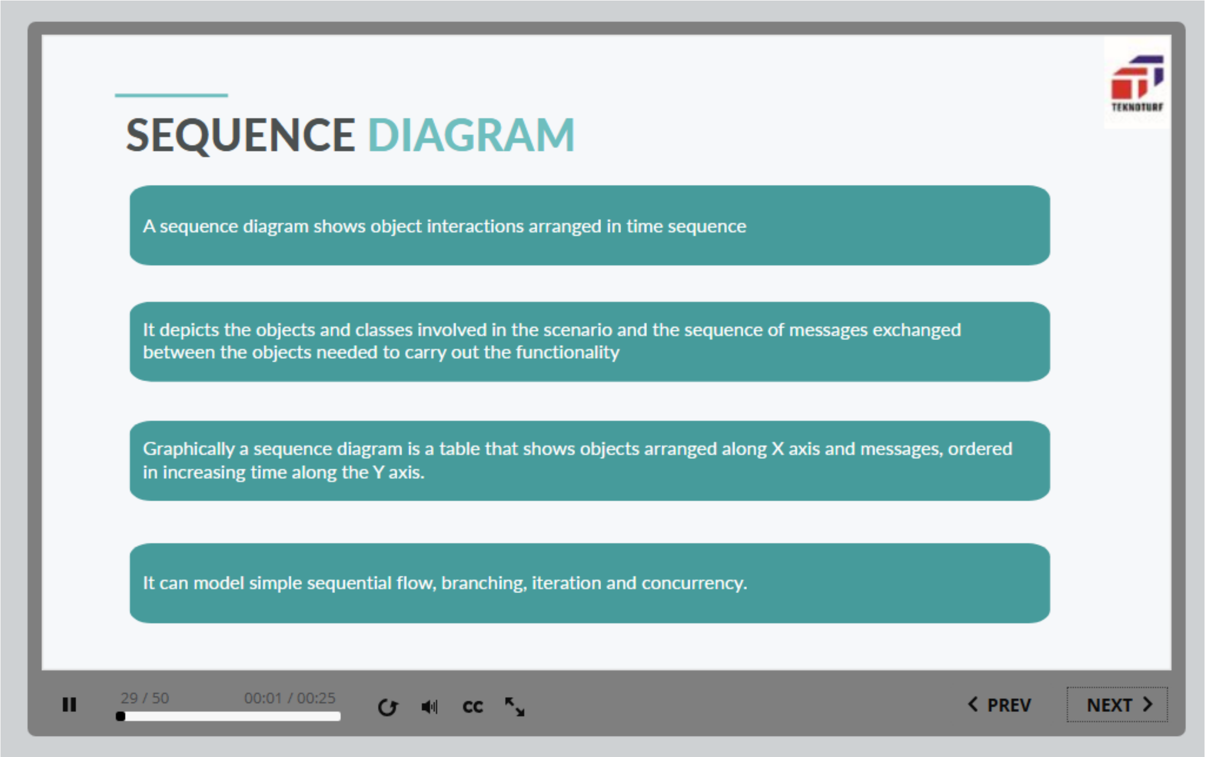
+addBookDetails(book):boolean
+updateBookDetails(bookId):Book
+viewBook(bookId):Book
+deactivateBookDetails(bookId):Book

Book

-bookId:Integer
-bookName:String
-bookType:String
-bookPrice:float
-author:String

+getBookId():Integer
+setBookId(id):void
+getBookName():String
+setBookName(bookName):void
+getBookType():String
+setBookType(bookType):void
+getBookPrice():Float
+setBookPrice(price):void
+getAuthor():String

This slide depicts the class diagram for Zeestore Book Management. ZeeStore and Book are the two classes that holds the required attributes and methods . Book is part of the Zeestore, so the relationship between ZeeStore and Book is aggregation



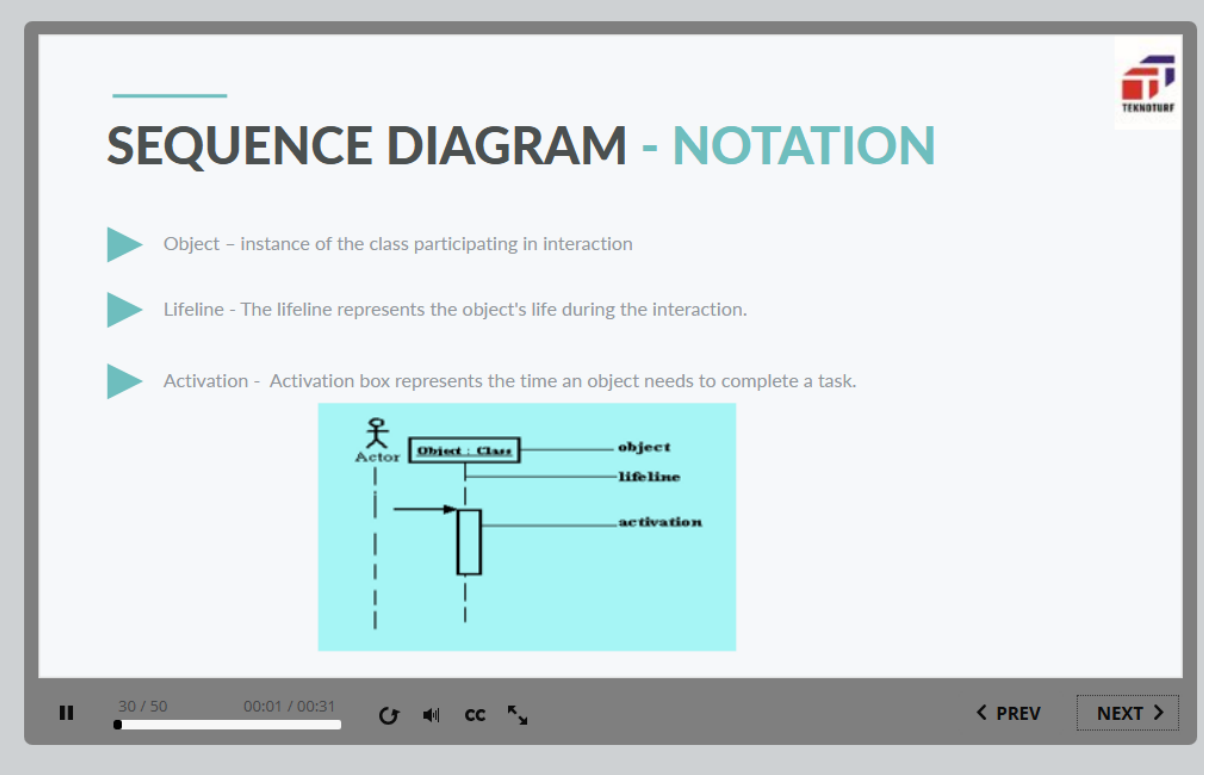
SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality

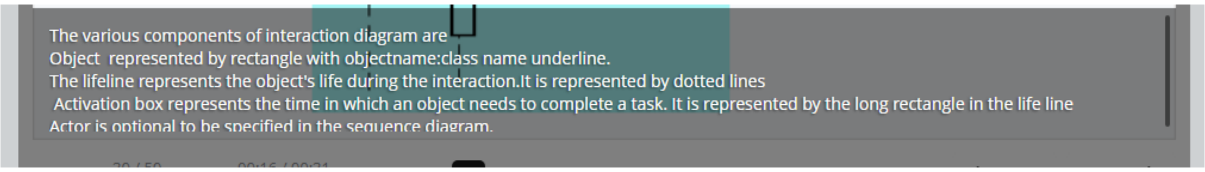
Graphically a sequence diagram is a table that shows objects arranged along X axis and messages, ordered in increasing time along the Y axis.

It can model simple sequential flow, branching, iteration and concurrency.

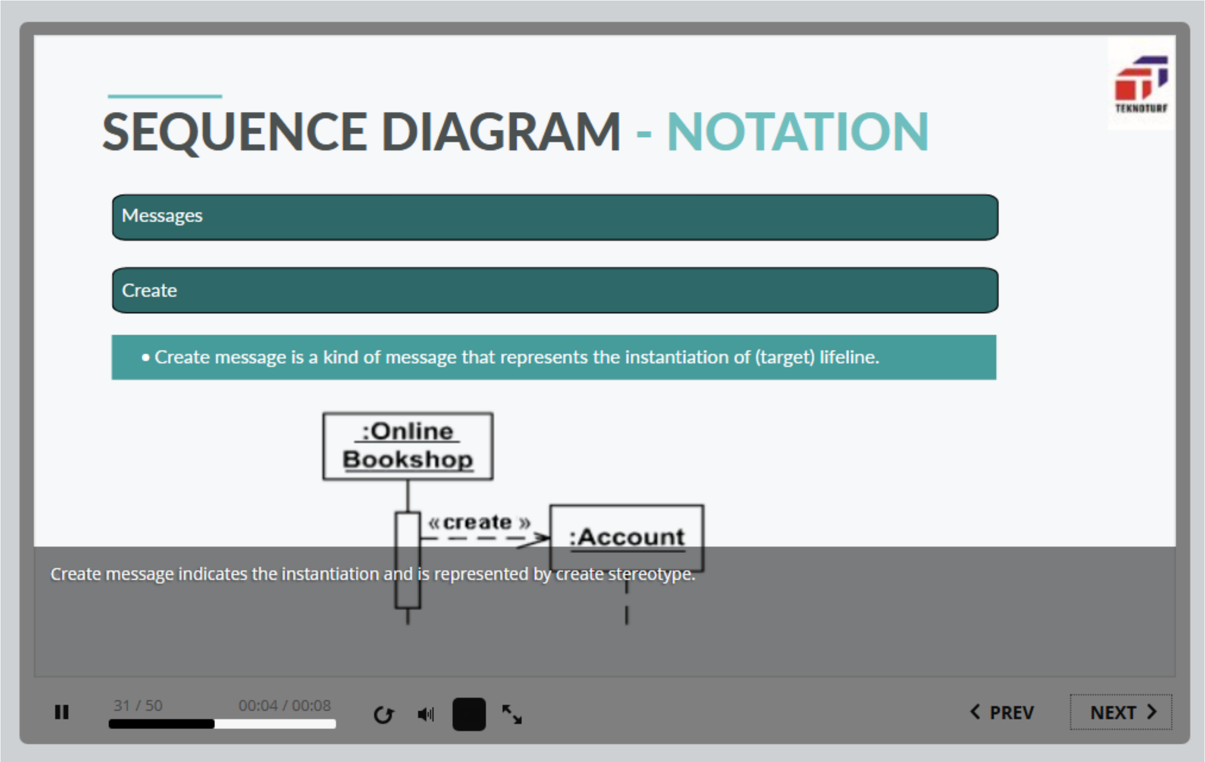


SEQUENCE DIAGRAM - NOTATION

- Object – instance of the class participating in interaction
- Lifeline - The lifeline represents the object's life during the interaction.
- Activation - Activation box represents the time an object needs to complete a task.



The various components of interaction diagram are
Object represented by rectangle with objectname:class name underline.
The lifeline represents the object's life during the interaction.It is represented by dotted lines
Activation box represents the time in which an object needs to complete a task. It is represented by the long rectangle in the life line
Actor is optional to be specified in the sequence diagram.



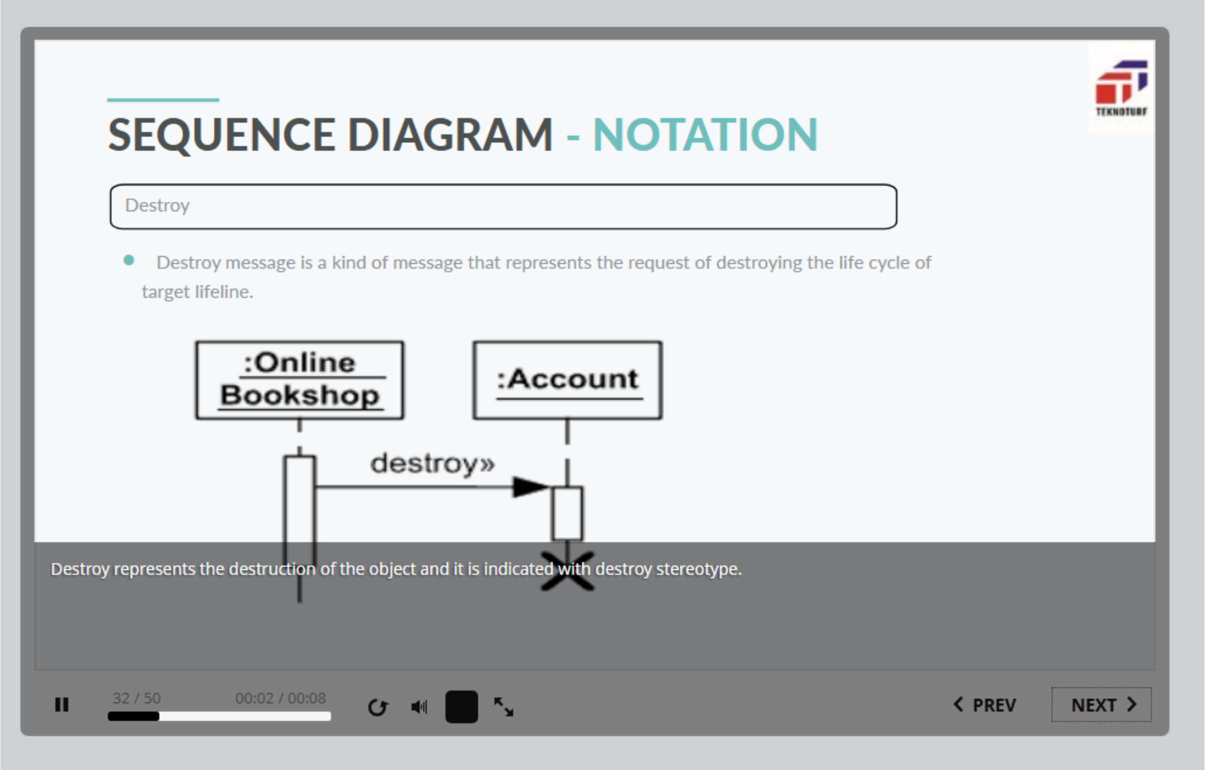
SEQUENCE DIAGRAM - NOTATION

Messages

Create

- Create message is a kind of message that represents the instantiation of (target) lifeline.

Create message indicates the instantiation and is represented by create stereotype.

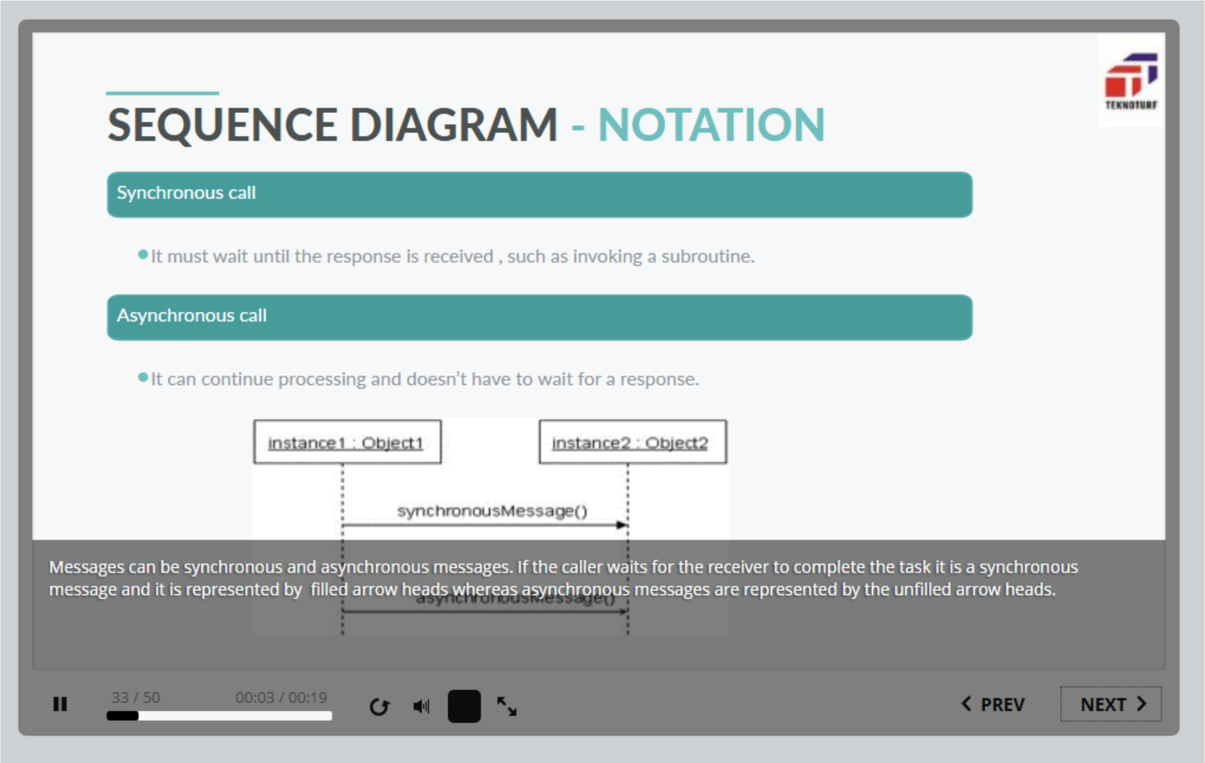


SEQUENCE DIAGRAM - NOTATION

Destroy

- Destroy message is a kind of message that represents the request of destroying the life cycle of target lifeline.

Destroy represents the destruction of the object and it is indicated with destroy stereotype.



SEQUENCE DIAGRAM - NOTATION

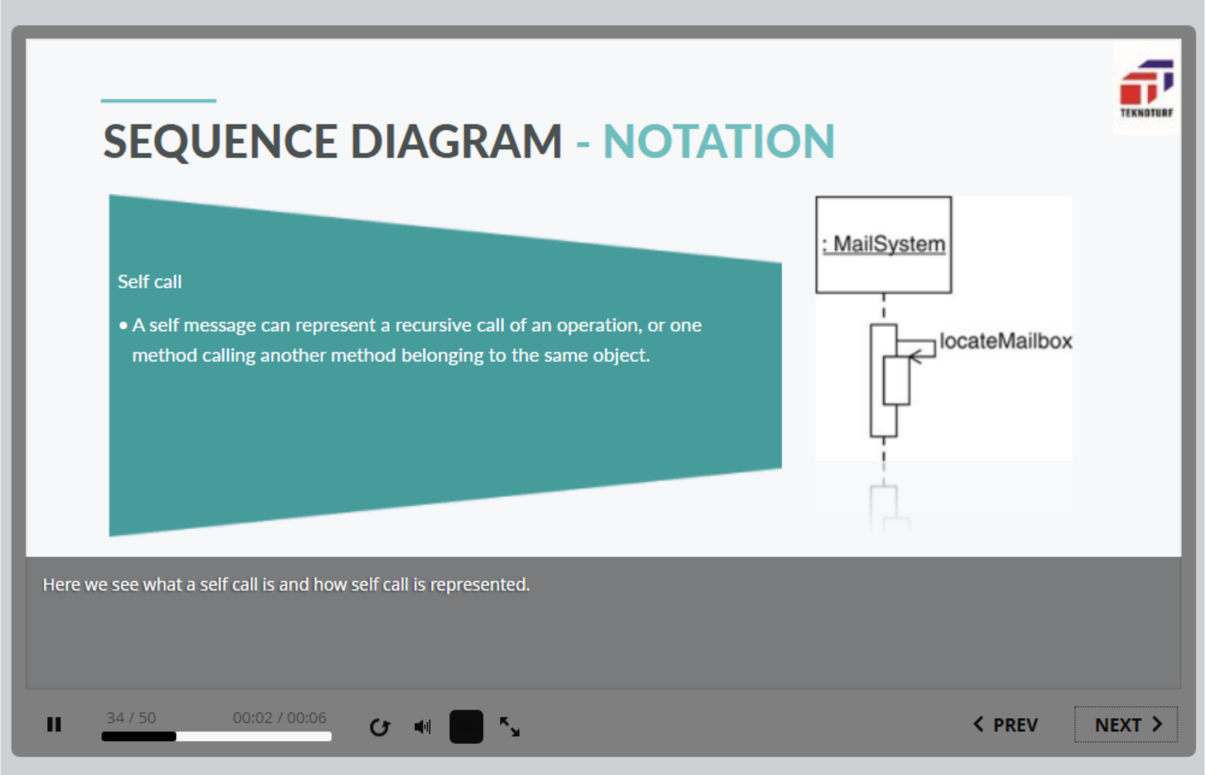
Synchronous call

- It must wait until the response is received , such as invoking a subroutine.

Asynchronous call

- It can continue processing and doesn't have to wait for a response.

Messages can be synchronous and asynchronous messages. If the caller waits for the receiver to complete the task it is a synchronous message and it is represented by filled arrow heads whereas asynchronous messages are represented by the unfilled arrow heads.

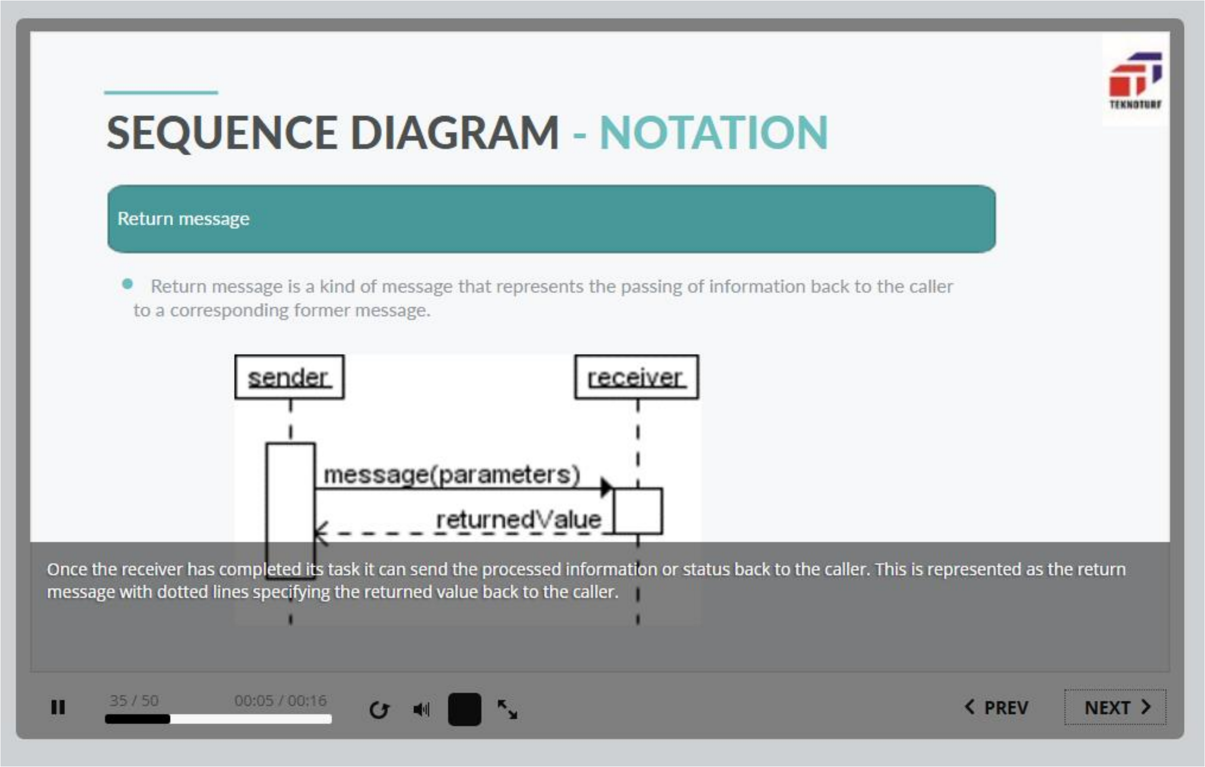


SEQUENCE DIAGRAM - NOTATION

Self call

- A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object.

Here we see what a self call is and how self call is represented.

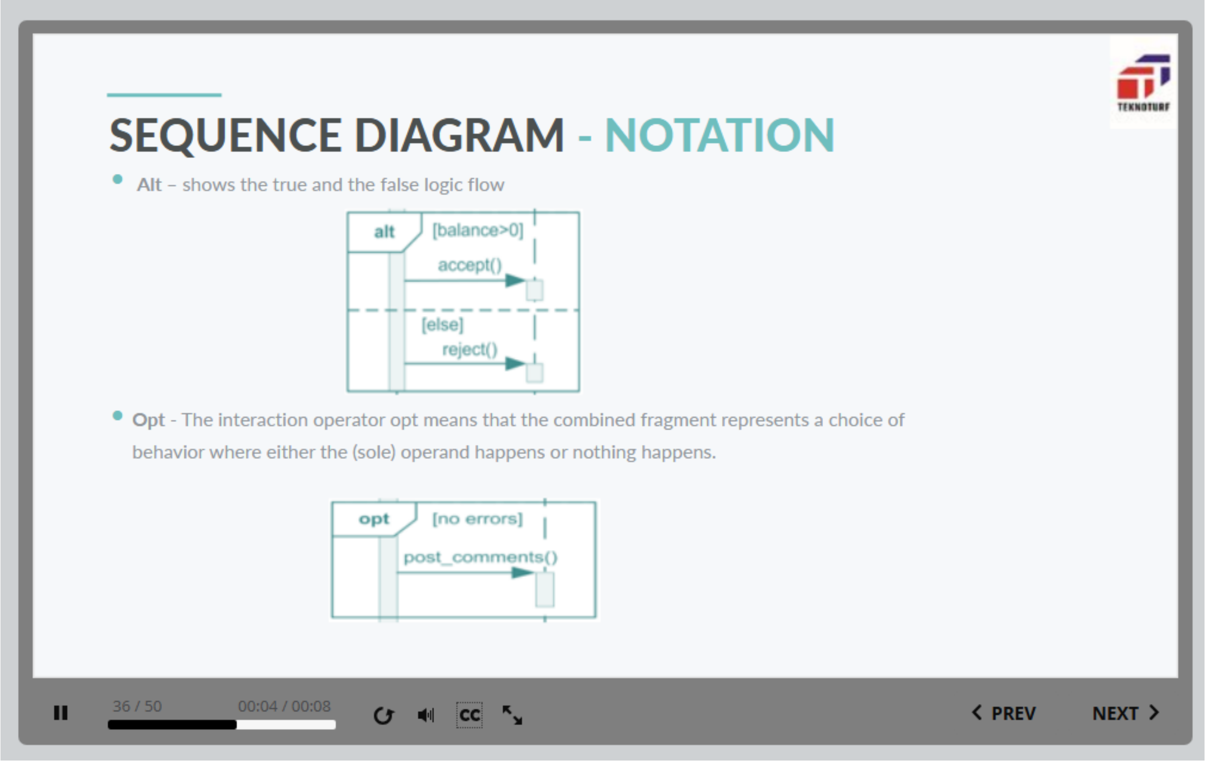


SEQUENCE DIAGRAM - NOTATION

Return message

- Return message is a kind of message that represents the passing of information back to the caller to a corresponding former message.

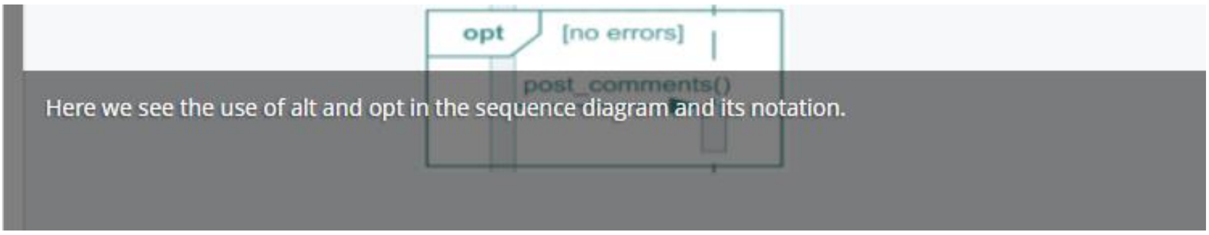
Once the receiver has completed its task it can send the processed information or status back to the caller. This is represented as the return message with dotted lines specifying the returned value back to the caller.



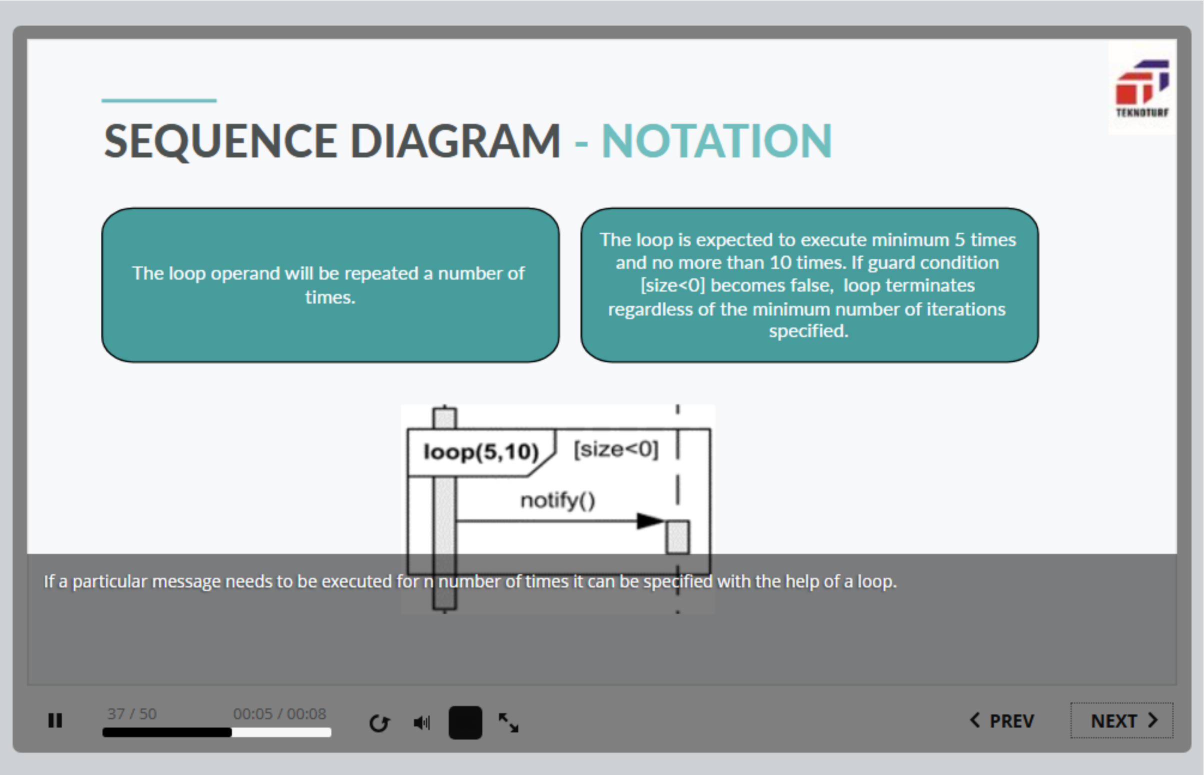
SEQUENCE DIAGRAM - NOTATION

- **Alt** – shows the true and the false logic flow

- **Opt** - The interaction operator opt means that the combined fragment represents a choice of behavior where either the (sole) operand happens or nothing happens.



Here we see the use of alt and opt in the sequence diagram and its notation.

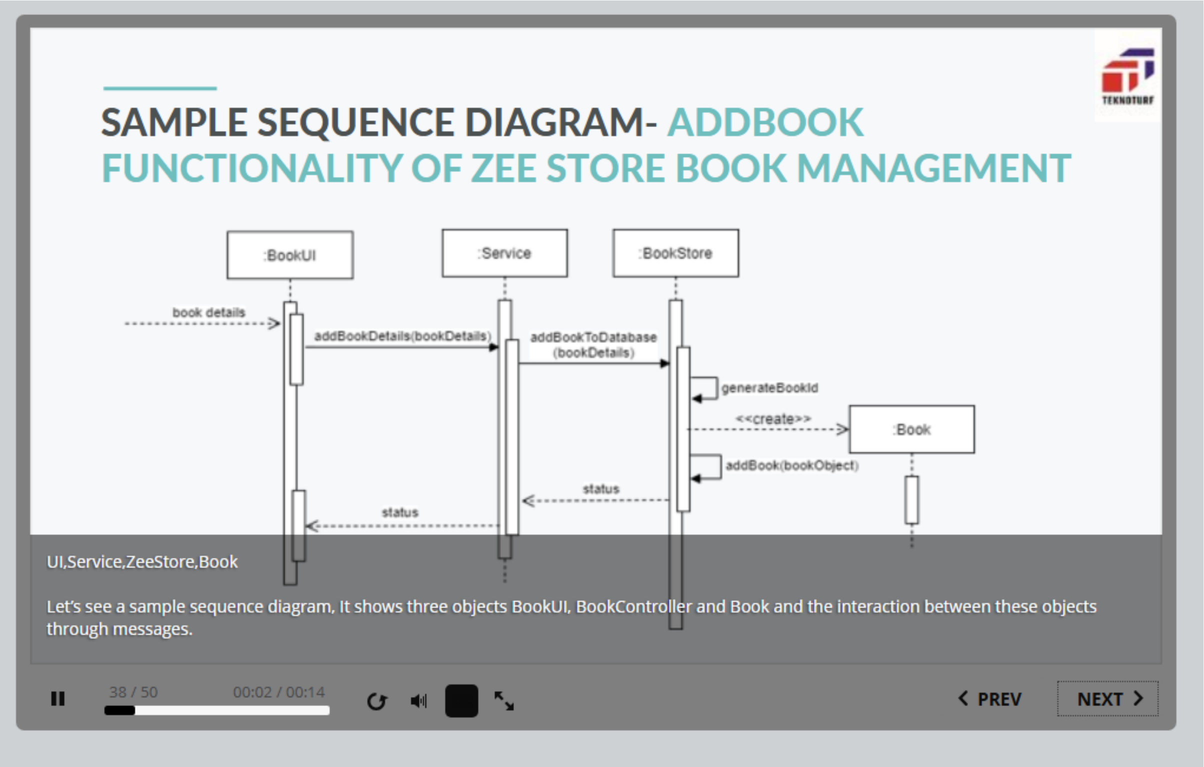


SEQUENCE DIAGRAM - NOTATION

The loop operand will be repeated a number of times.

The loop is expected to execute minimum 5 times and no more than 10 times. If guard condition [size<0] becomes false, loop terminates regardless of the minimum number of iterations specified.

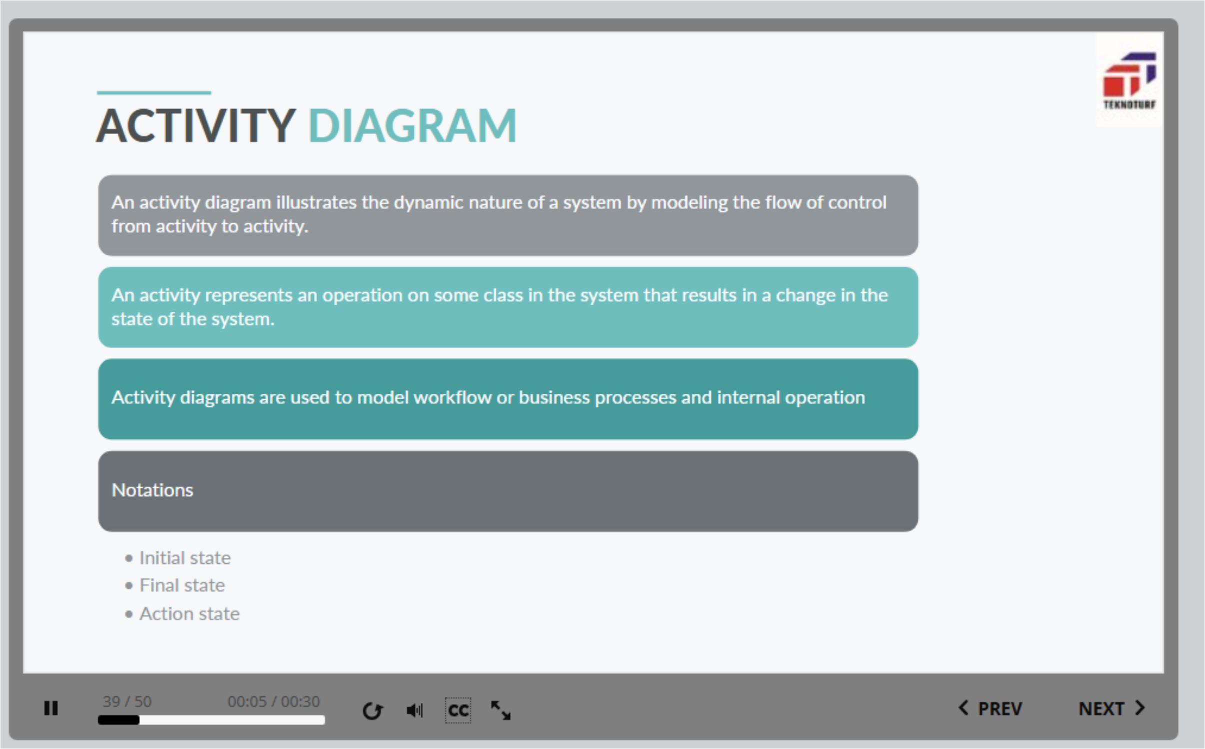
If a particular message needs to be executed for n number of times it can be specified with the help of a loop.



SAMPLE SEQUENCE DIAGRAM- ADDBOOK FUNCTIONALITY OF ZEE STORE BOOK MANAGEMENT

UI,Service,ZeeStore,Book

Let's see a sample sequence diagram, It shows three objects BookUI, BookController and Book and the interaction between these objects through messages.



ACTIVITY DIAGRAM

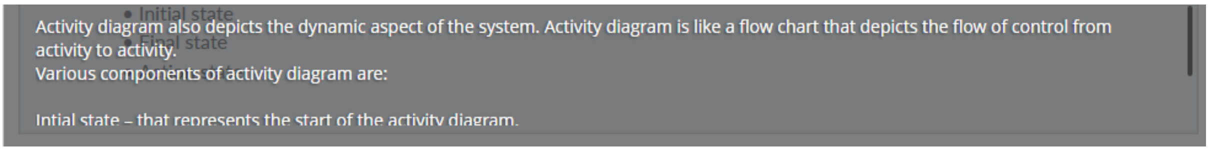
An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity.

An activity represents an operation on some class in the system that results in a change in the state of the system.

Activity diagrams are used to model workflow or business processes and internal operation

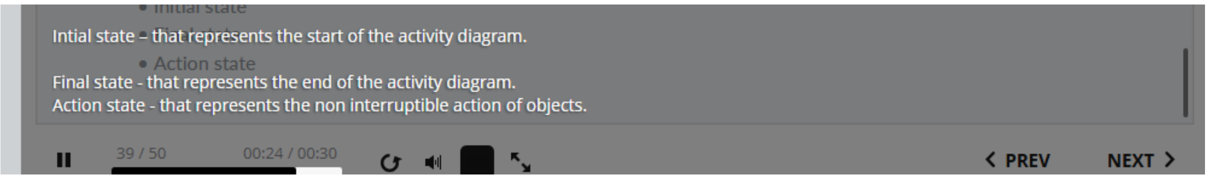
Notations

- Initial state
- Final state
- Action state



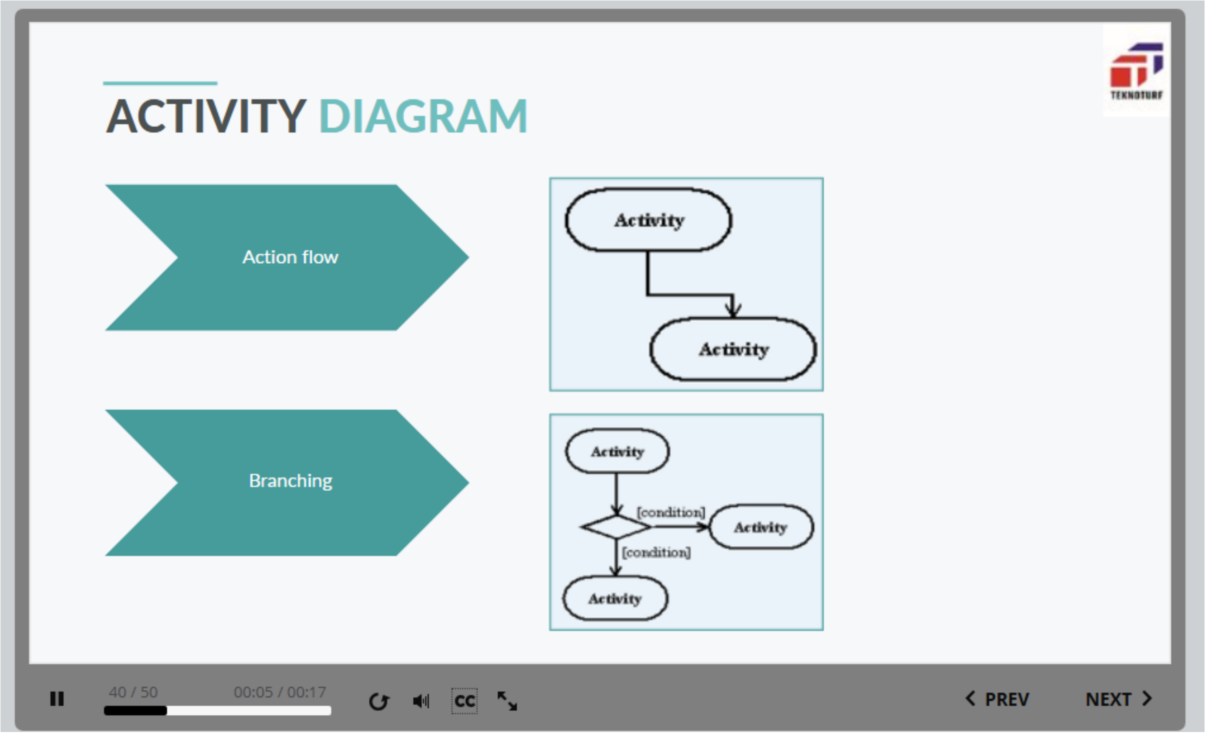
Activity diagram also depicts the dynamic aspect of the system. Activity diagram is like a flow chart that depicts the flow of control from activity to activity.
Various components of activity diagram are:

Intial state – that represents the start of the activity diagram.



Intial state – that represents the start of the activity diagram.

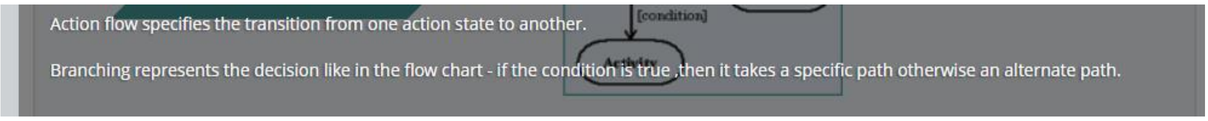
Final state - that represents the end of the activity diagram.
Action state - that represents the non interruptible action of objects.



ACTIVITY DIAGRAM

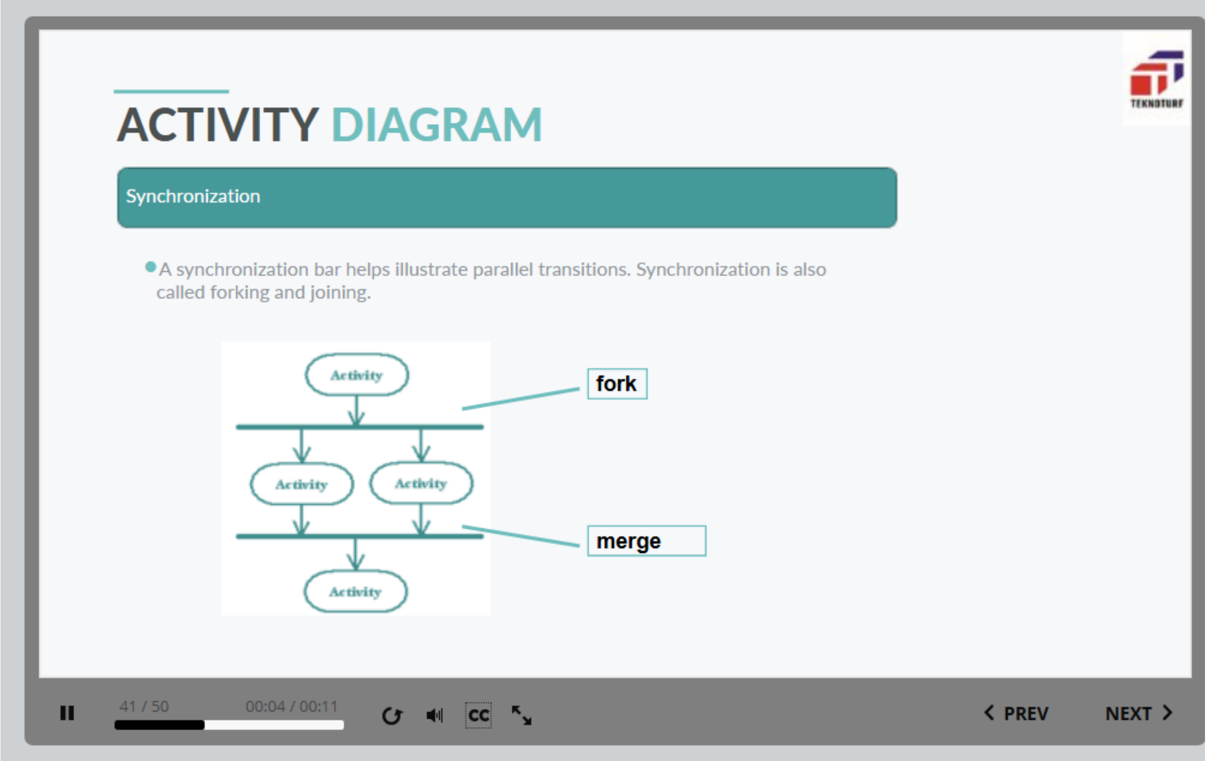
Action flow

Branching



Action flow specifies the transition from one action state to another.

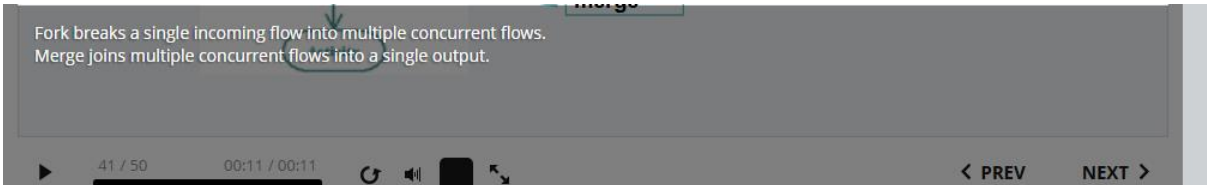
Branching represents the decision like in the flow chart - if the condition is true ,then it takes a specific path otherwise an alternate path.



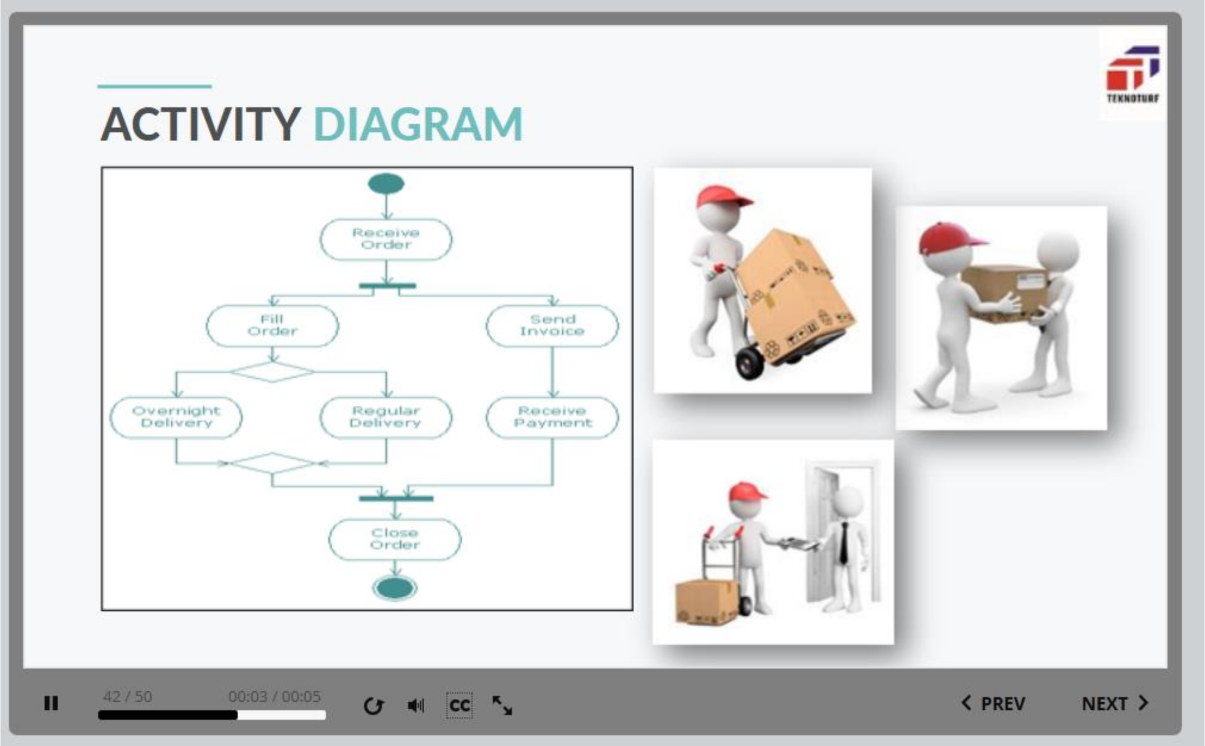
ACTIVITY DIAGRAM

Synchronization

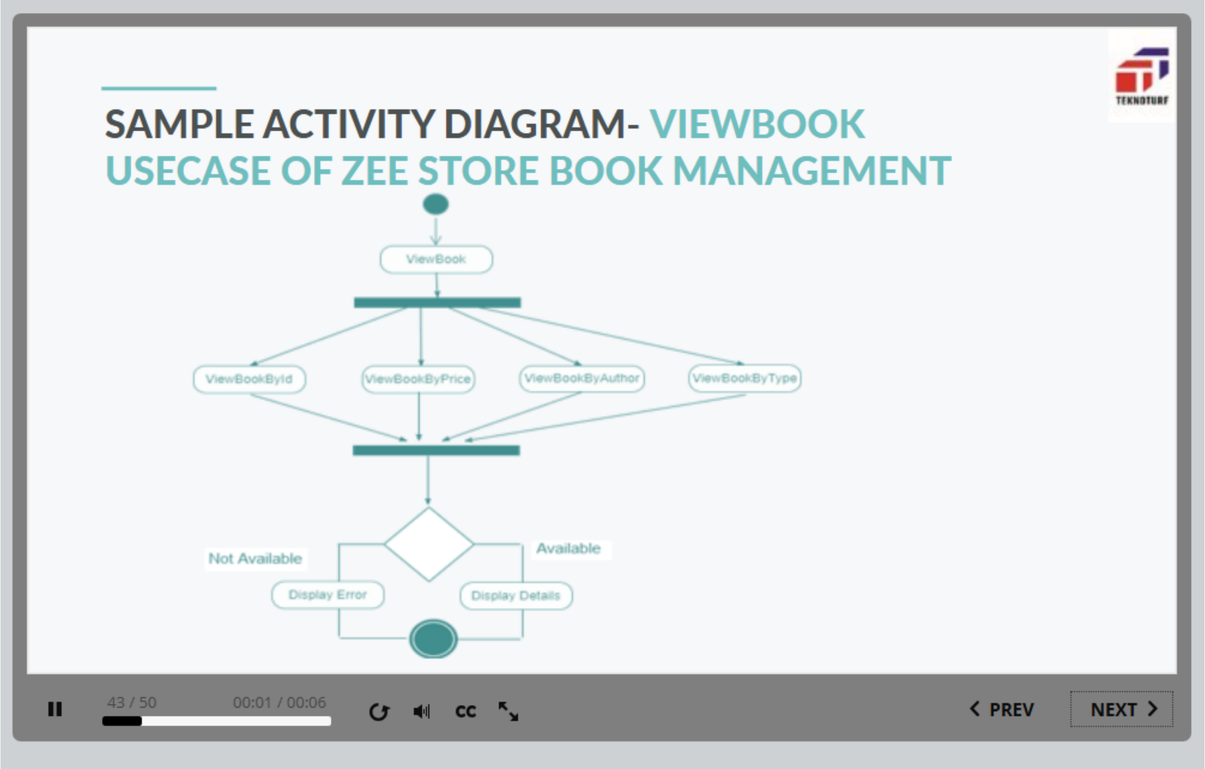
- A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



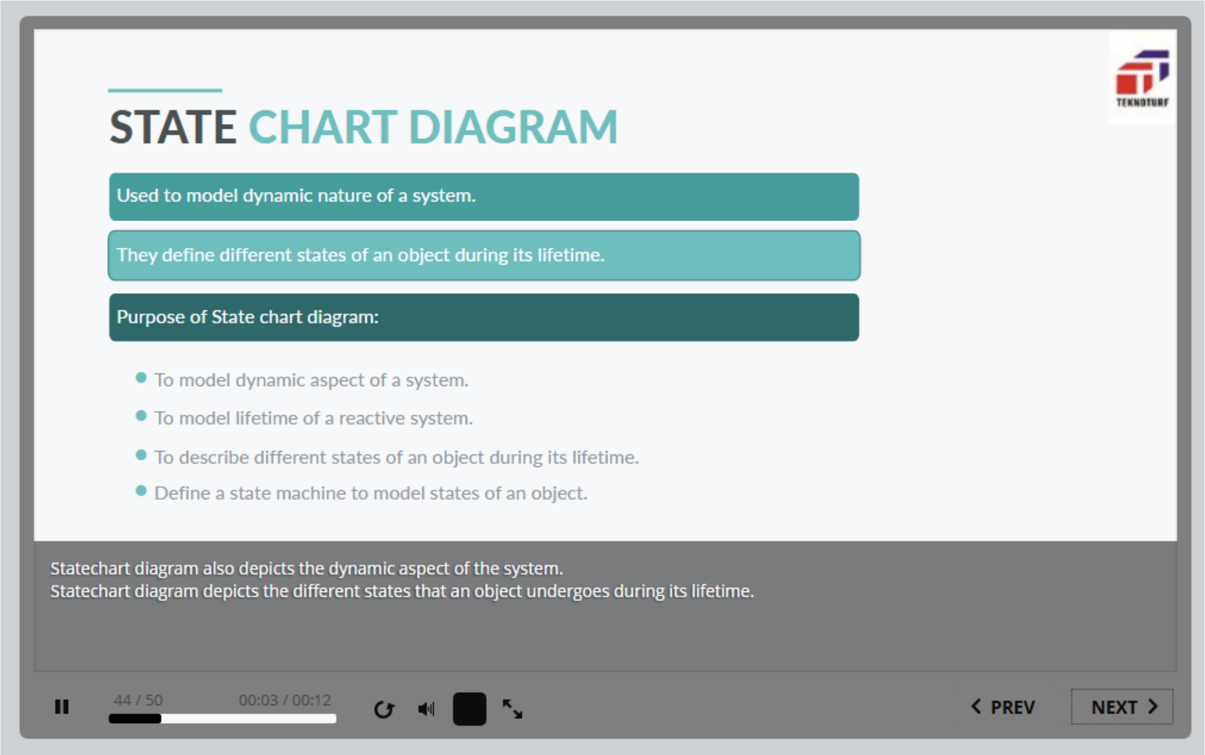
Fork breaks a single incoming flow into multiple concurrent flows.
Merge joins multiple concurrent flows into a single output.



ACTIVITY DIAGRAM



SAMPLE ACTIVITY DIAGRAM- VIEWBOOK USECASE OF ZEE STORE BOOK MANAGEMENT



STATE CHART DIAGRAM

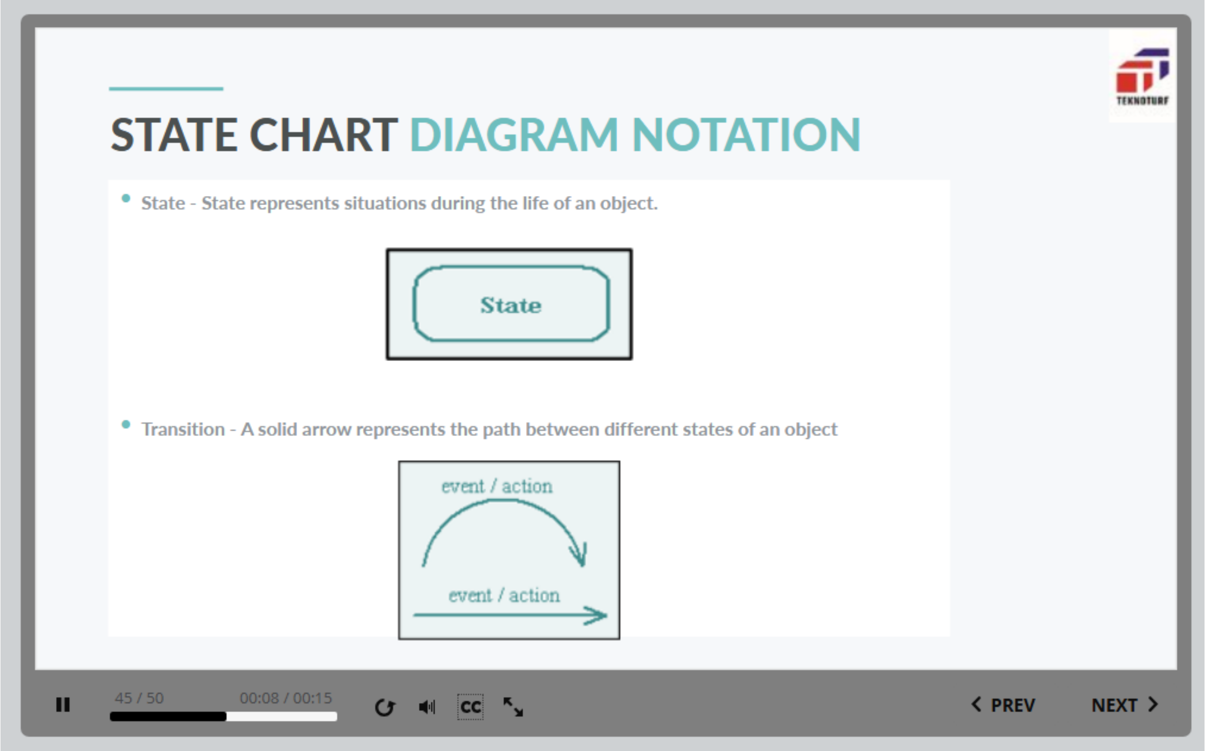
Used to model dynamic nature of a system.

They define different states of an object during its lifetime.

Purpose of State chart diagram:

- To model dynamic aspect of a system.
- To model lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- Define a state machine to model states of an object.

Statechart diagram also depicts the dynamic aspect of the system.
Statechart diagram depicts the different states that an object undergoes during its lifetime.

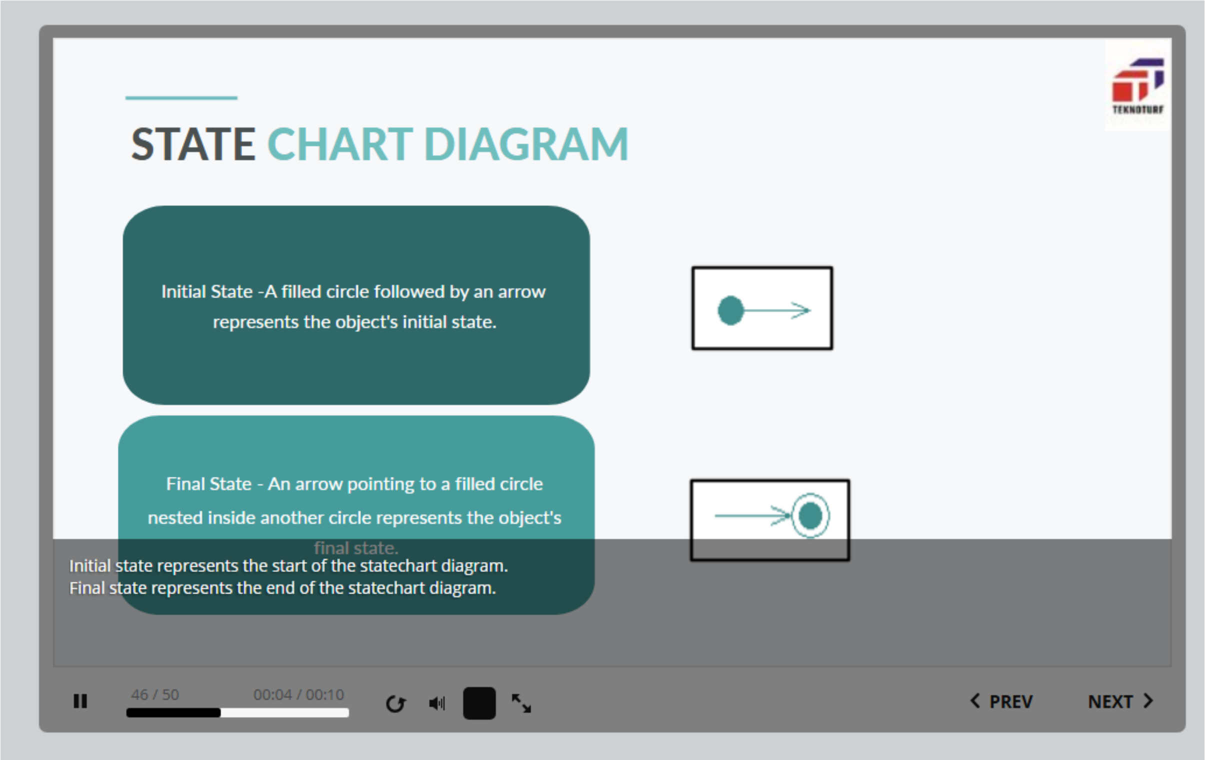


STATE CHART DIAGRAM NOTATION

- State - State represents situations during the life of an object.

- Transition - A solid arrow represents the path between different states of an object

The various notations of state chart diagram are:
State - that represents the situations during the life of an object.
Transition - which is a path between different states of an object.

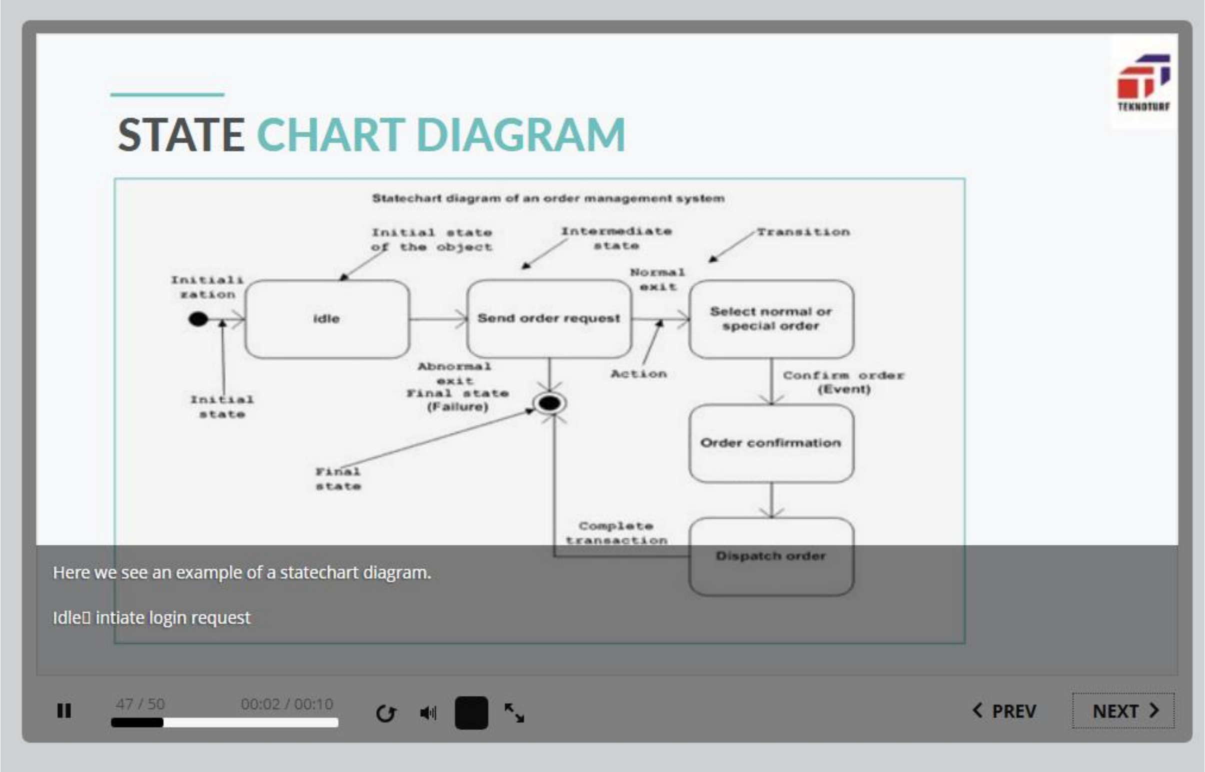


STATE CHART DIAGRAM

Initial State -A filled circle followed by an arrow represents the object's initial state.

Final State - An arrow pointing to a filled circle nested inside another circle represents the object's final state.

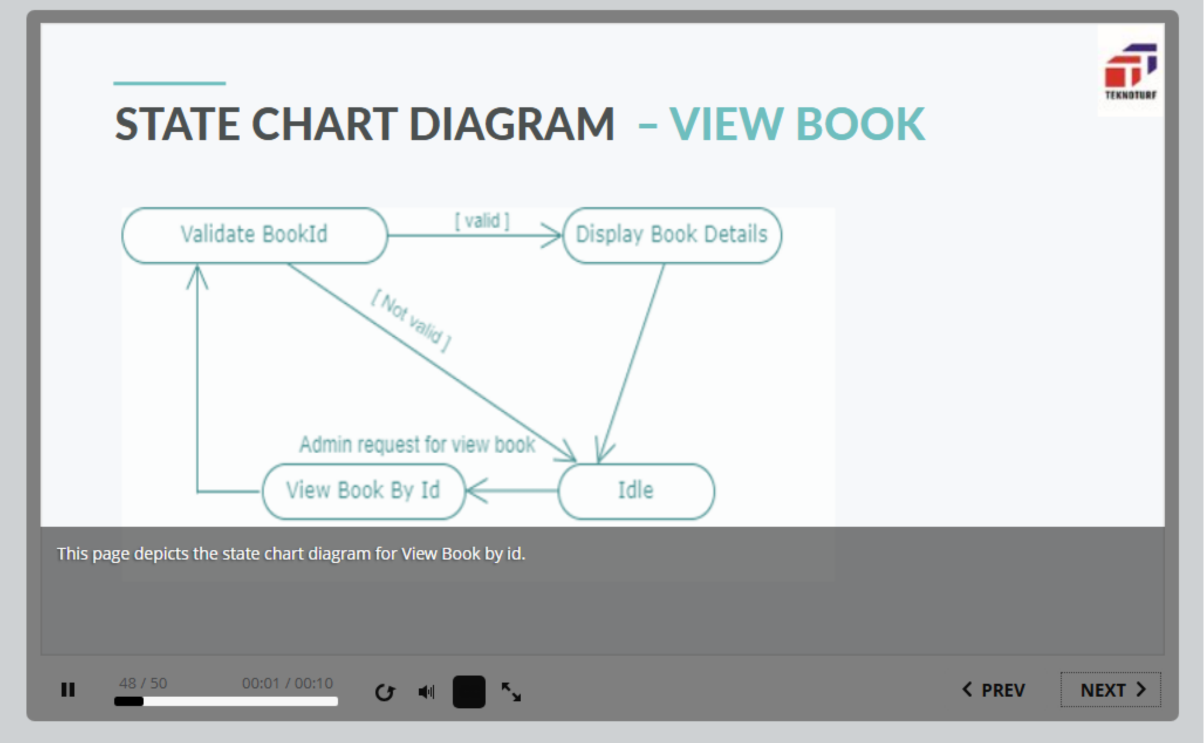
Initial state represents the start of the statechart diagram.
Final state represents the end of the statechart diagram.



STATE CHART DIAGRAM

Here we see an example of a statechart diagram.

Idle□ intiate login request



STATE CHART DIAGRAM – VIEW BOOK

Validate BookId

[valid]

Display Book Details

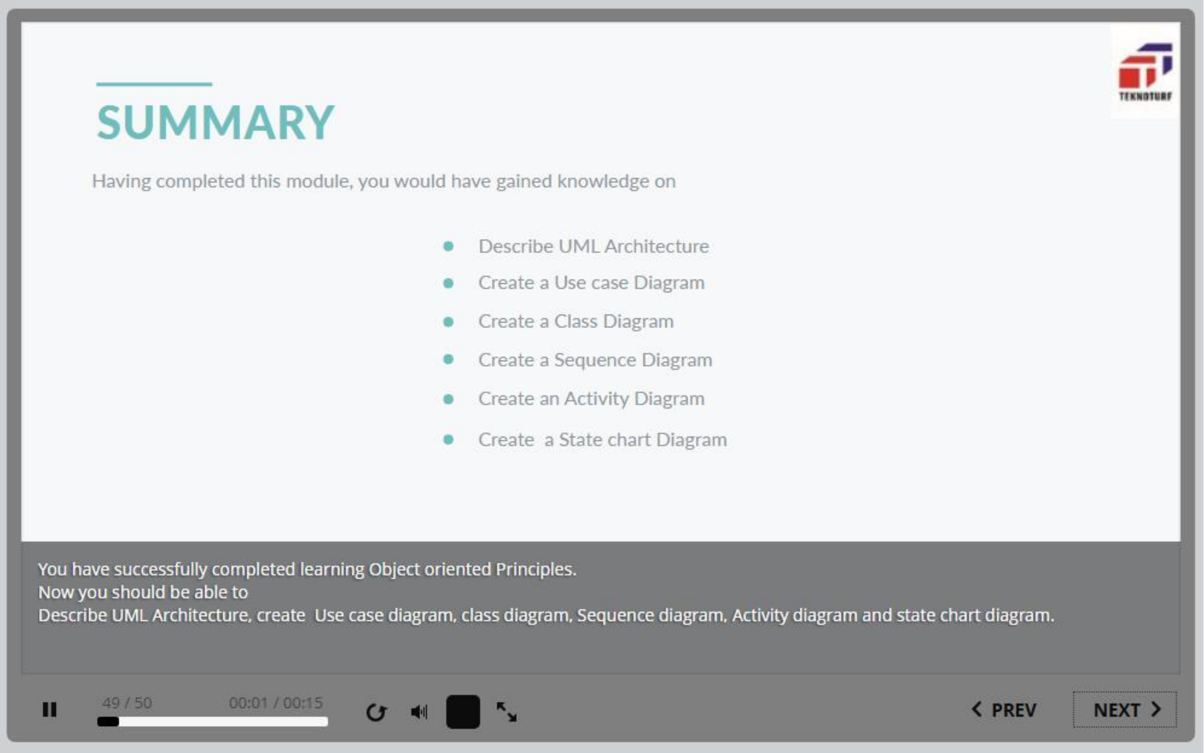
[Not valid]

Admin request for view book

View Book By Id

Idle

This page depicts the state chart diagram for View Book by id.



SUMMARY

Having completed this module, you would have gained knowledge on

- Describe UML Architecture
- Create a Use case Diagram
- Create a Class Diagram
- Create a Sequence Diagram
- Create an Activity Diagram
- Create a State chart Diagram

You have successfully completed learning Object oriented Principles.
Now you should be able to
Describe UML Architecture, create Use case diagram, class diagram, Sequence diagram, Activity diagram and state chart diagram.

THANK YOU

Thank you! Happy learning!!