

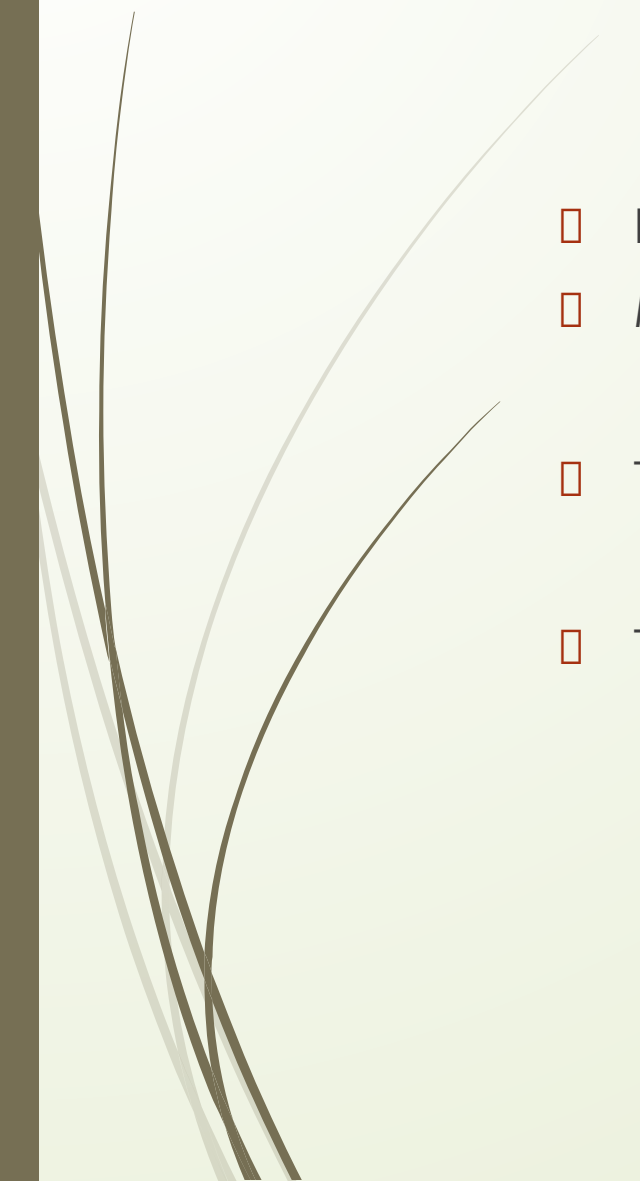


Object Oriented Design

Lecture 01: Introduction



Object Oriented Analysis and Design

- Lecturer: Ms. Sulochana Rupasinghe (sulochana.r@iit.ac.lk)
 - Module Leader: Mr. Saman Hettiarachchi (saman.h@iit.ac.lk)
 - Theory + Practical: 2 Hours
 - Tools: StarUML, Draw.IO, Lucidchart
- 



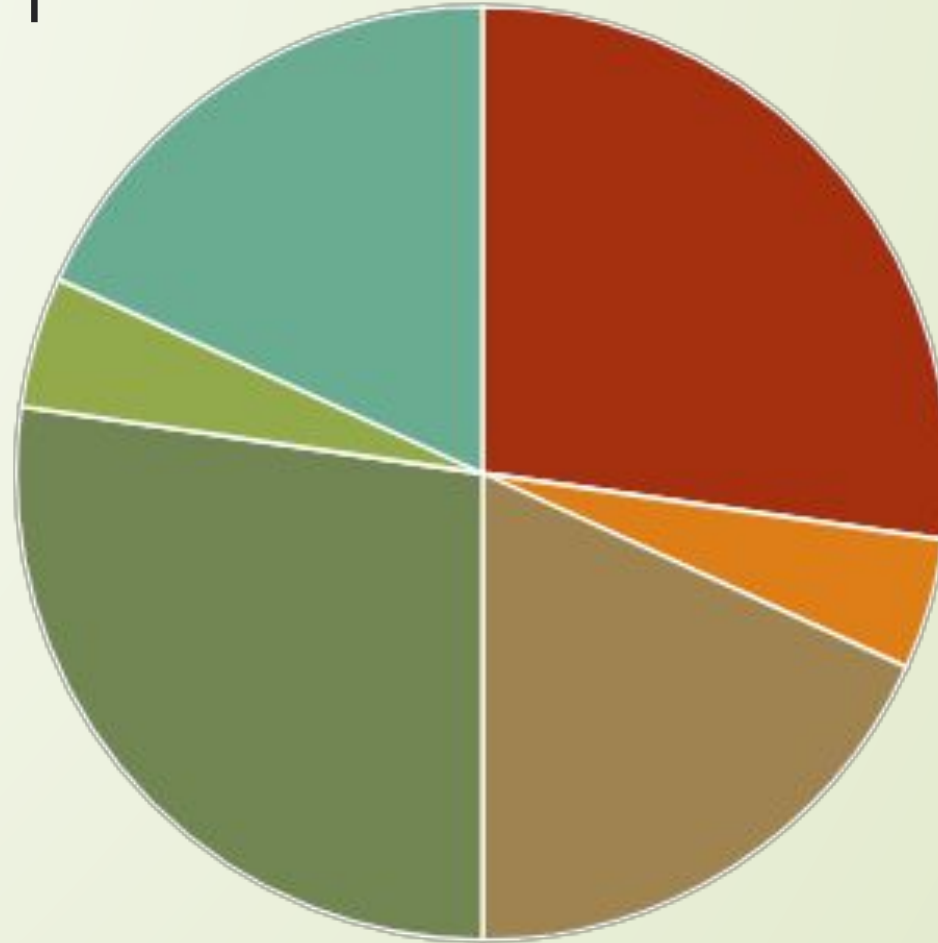
Module Content



- Week 1: Introduction to OOAD
- Week 2: Use Case Diagram
- Week 3: Use Case Descriptions
- Week 4: Activity Diagram
- Week 5: Domain Models
- Week 6: Student Engagement Week
- Week 7: Class Diagrams – Part 1
- Week 8: Class Diagrams – Part 2
- Week 9: Sequence Diagram
- Week 10: State Machine Diagram
- Week 11: SOLID Principles and Design Patterns

Lecture Plan

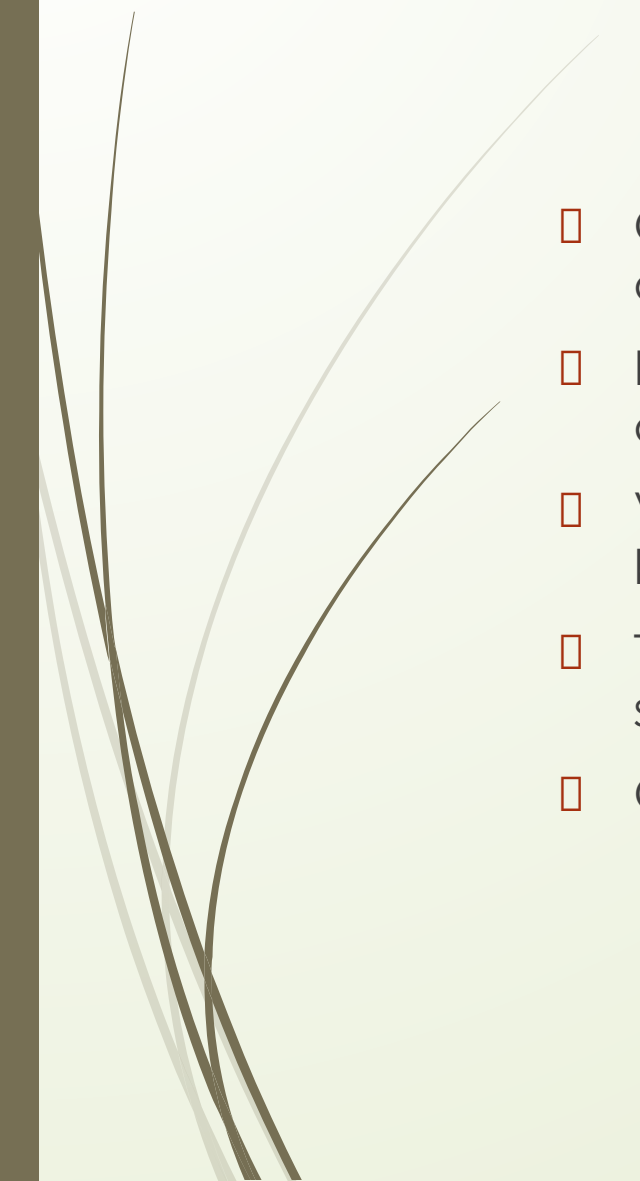
2 Hours




■ Intro + Basic Elements ■ Short Video ■ Activity 1 ■ Advance Features ■ Short Video ■ Activity 2



What is OOAD ??

- ❑ Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects.
 - ❑ Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior.
 - ❑ Various models can be created to show the static structure, dynamic behavior, and run-time deployment of these collaborating objects.
 - ❑ There are a number of different notations for representing these models, such as the Unified Modeling Language (UML).
 - ❑ OOA focuses on what the system does, OOD on how the system does it.
- 



STRUCTURED PROGRAMMING VERSUS OBJECT ORIENTED PROGRAMMING

STRUCTURED PROGRAMMING

A programming paradigm that divides the code into modules or function

Focuses on dividing the program into a set of functions in which each function works as a subprogram

Difficult to modify structured programs

Main method communicates with functions by calling those functions in the main program

There are no access specifiers

Data is not secured

Difficult to reuse code

OBJECT ORIENTED PROGRAMMING

A programming paradigm based on the concept of objects, which contain data in the form of fields known as attributes, and code in the form of procedures known as methods

Focuses on representing a program using a set of objects which encapsulates data and object

Easier to modify Object Oriented programs

Objects communicate with each other by passing messages

There are access specifiers such as private, public and protected

Data is secure

Easy to reuse code

Visit www.PEDIAA.com

Software Development Life Cycle





Technologies in Different Phases

Requirements – CRC cards, use cases, XP stories

Modeling requirements and design –

UML: class diagrams, sequence diagrams,...

UML tools: MS Visio (free to you!), Together ControlCenter, others

Implementation – Java

Development and Testing – Some IDE with debugger, GUI support



Outcomes



Object-oriented analysis and design –

Goals, what to do, what not to do

What to model and how to evaluate it

UML –

Use cases, class diagrams for requirements specification

Class diagrams, sequence diagrams, state diagrams, packages for design

Design patterns – What they are, how they're described, a few common patterns



Outcomes Contd.

Modeling and the SW lifecycle –

Clear understanding of the role of UML models throughout lifecycle

How requirements models are transformed to design

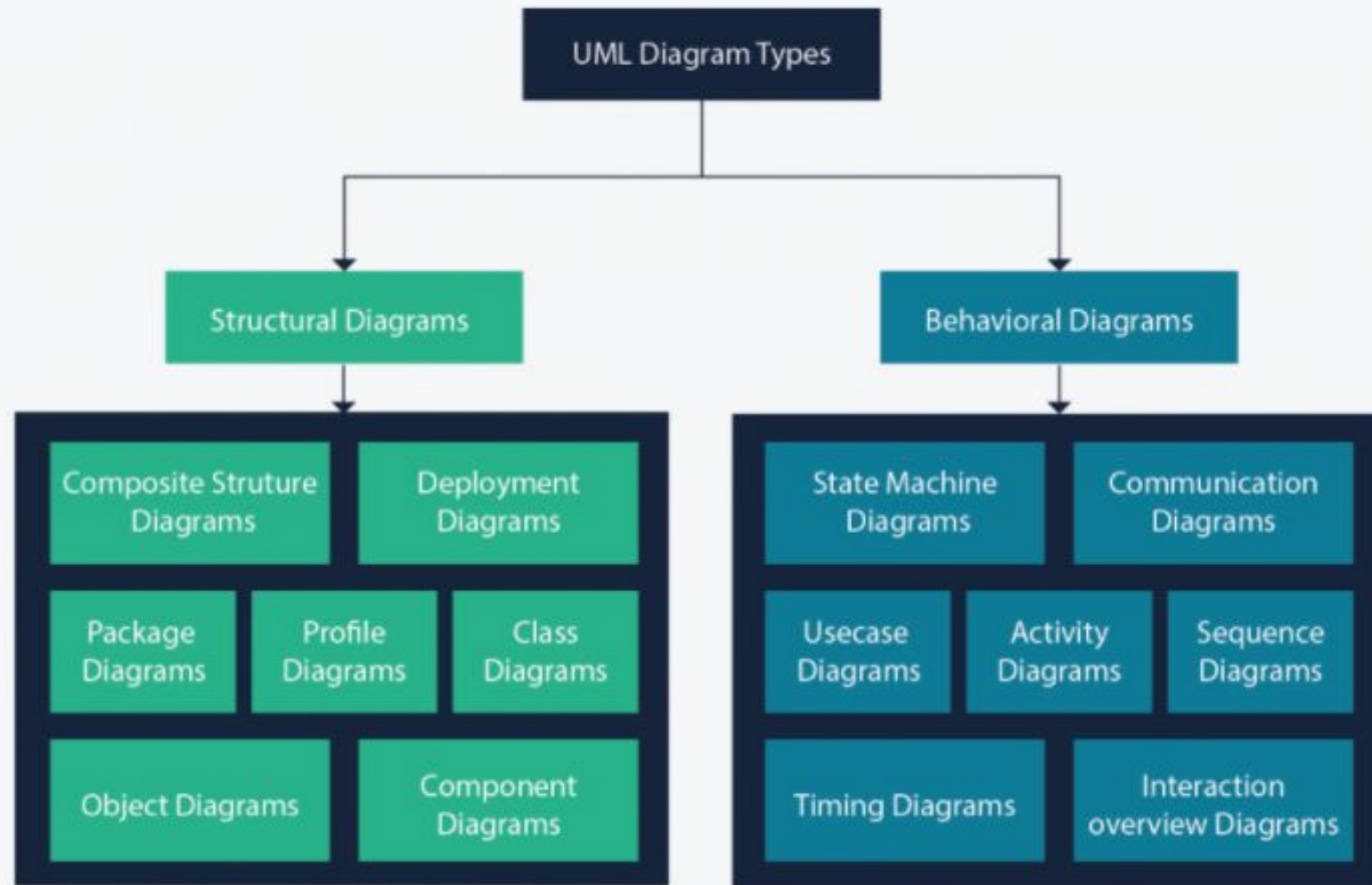
How design models transform to code

Evaluation –

Assessing design quality

On your own and using formal technical reviews

Unified Modeling Language (UML)

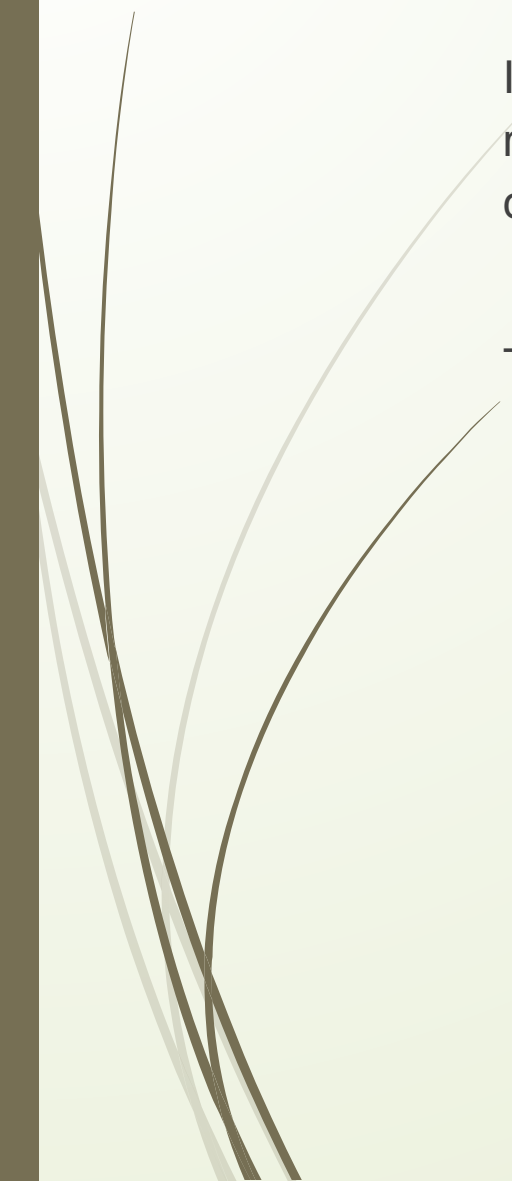




Static Structure

It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.

The process of object modelling can be visualized in the following steps –

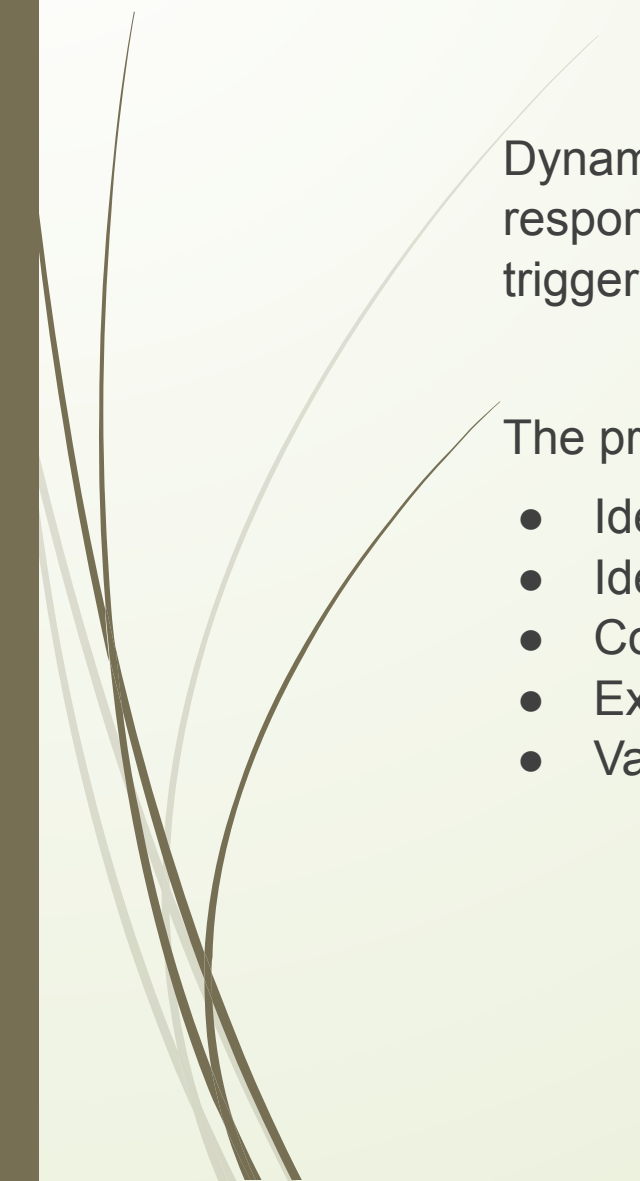
- Identify objects and group into classes
 - Identify the relationships among classes
 - Create user object model diagram
 - Define user object attributes
 - Define the operations that should be performed on the classes
 - Review glossary
- 



Dynamic Behavior

Dynamic Modelling can be defined as “a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world”.

The process of dynamic modelling can be visualized in the following steps –

- Identify states of each object
 - Identify events and analyze the applicability of actions
 - Construct dynamic model diagram, comprising of state transition diagrams
 - Express each state in terms of object attributes
 - Validate the state–transition diagrams drawn
- 



UML Diagrams

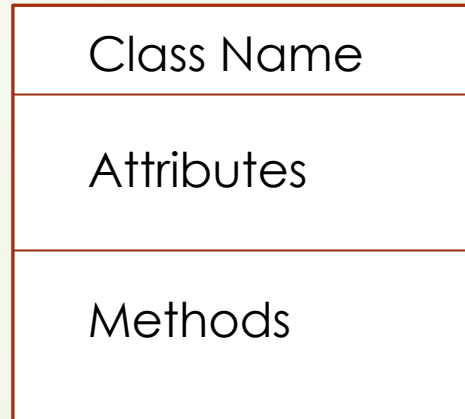


- ❑ **Structure diagrams** show the things in the modeled system. In a more technical term, they show different objects in a system.
- ❑ **Behavioral diagrams** show what should happen in a system. They describe how the objects interact with each other to create a functioning system.



Class Diagram

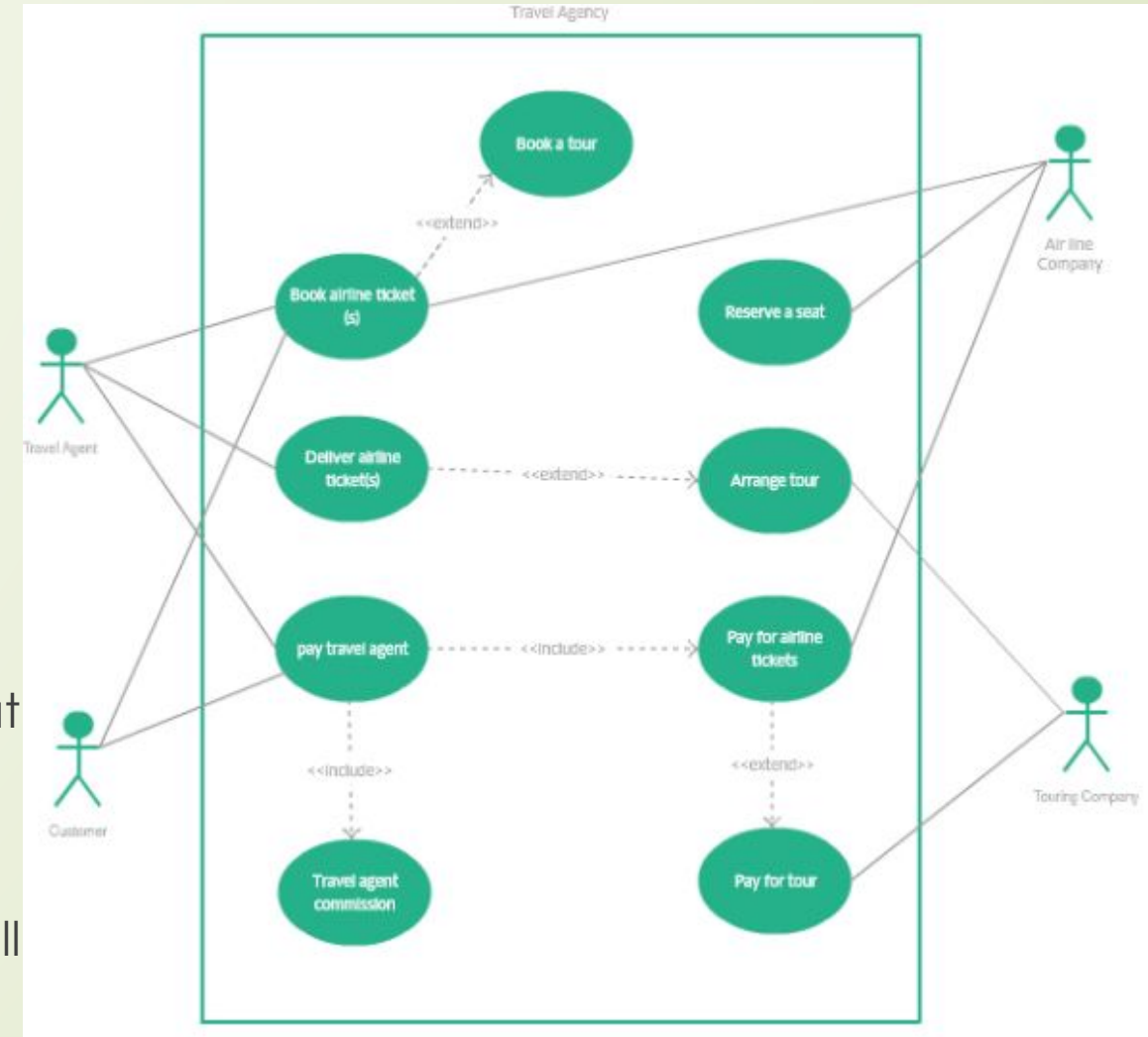
- ❑ The main building block of any object oriented solution.
- ❑ It shows the classes in a system, attributes, and operations of each class and the relationship between each class.



- ❑ In a large system with many related classes, classes are grouped together to create class diagrams.
- ❑ Different relationships between classes are shown by different types of arrows.

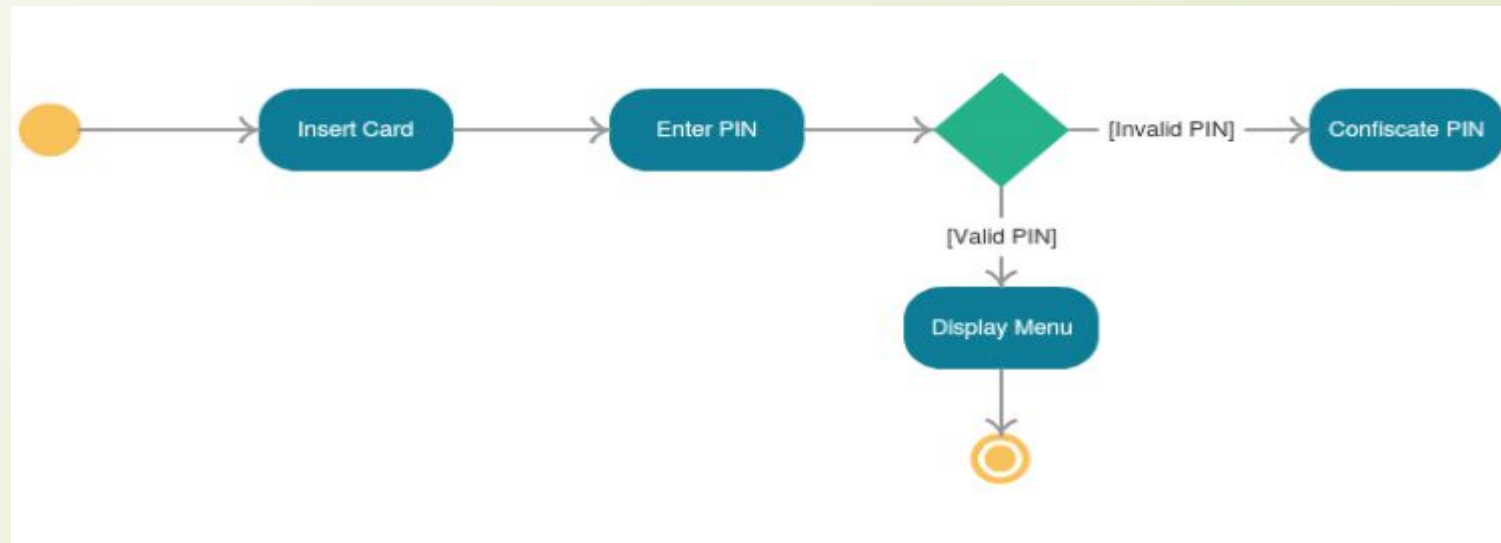
Use Case Diagram

- A use case diagram is the primary form of system/software requirements for a new software program underdeveloped.
- Use cases specify the expected behavior (what), and not the exact method of making it happen (how).
- Use cases once specified can be denoted both textual and visual representation.
- A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.



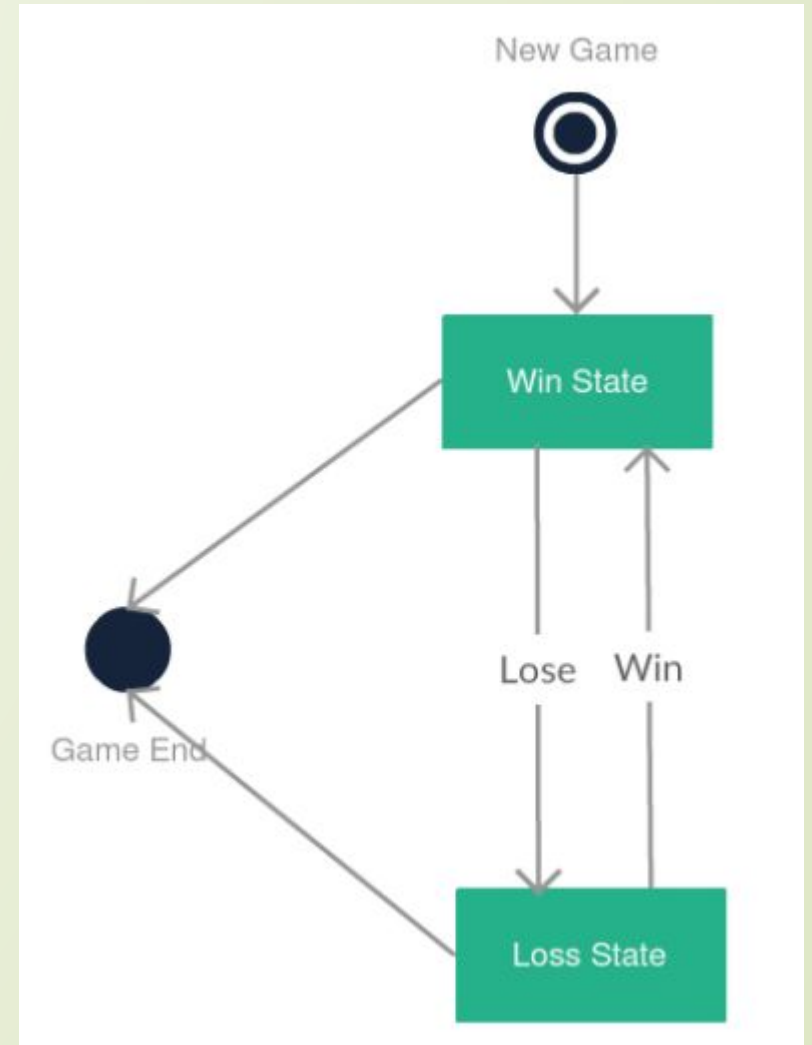
Activity Diagram

- Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.



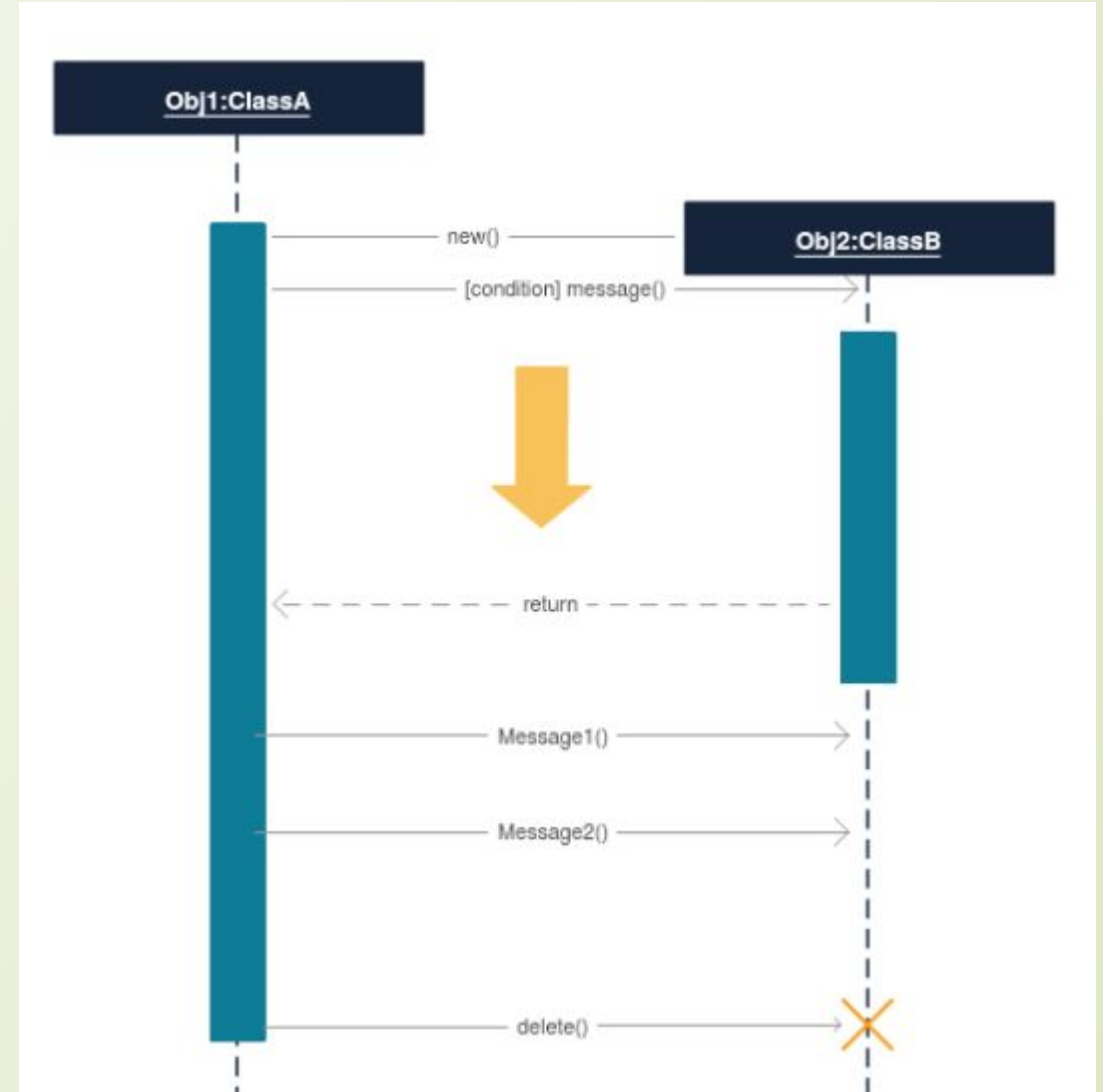
State Machine Diagram

- ❑ It describes different states of a component in a system. The states are specific to a component/object of a system.
- ❑ Statechart diagram describes the flow of control from one state to another state.
- ❑ States are defined as a condition in which an object exists and it changes when some event is triggered.
- ❑ The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.
- ❑ Statechart diagrams are also used for **forward and reverse engineering** of a system.



Sequence Diagram

- Sequence diagram shows how objects interact with each other and the order those interactions occur.
- It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows.





References



- *creately. 2021. UML Diagram Types Guide: Learn About All Types of UML Diagrams with Examples. [online] Available at: <<https://creately.com/blog/diagrams/uml-diagram-types-examples/#ClassDiagram>> [Accessed 26 September 2021].*