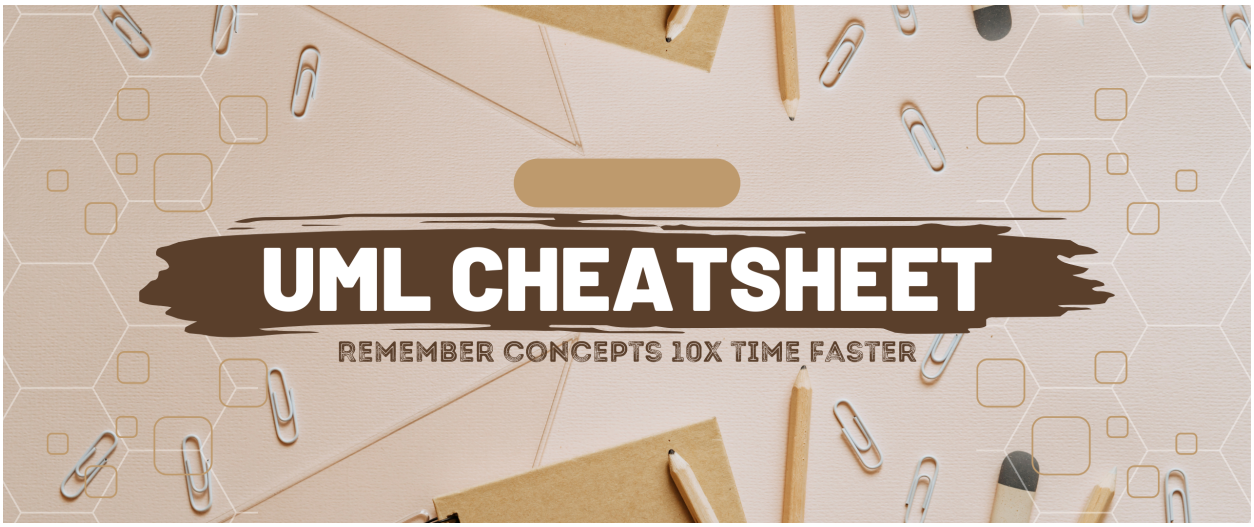
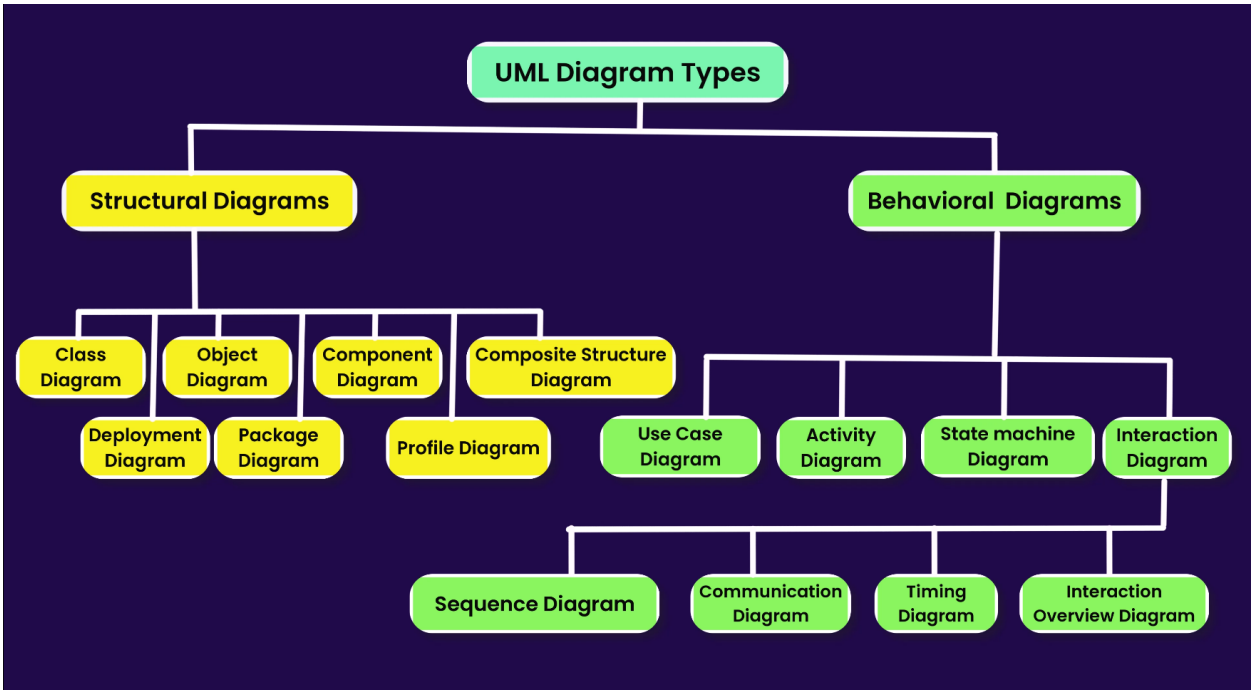


UML Cheat sheet

UML : Unified Modeling Language

Types of UML Diagrams :



A. Structural Diagrams

Represent the static aspects of the system and emphasizes the things must be present in the system being modeled.

1. Class Diagram :

Purpose : The Class Diagram describes the classes of the application being modeled, Blueprint of the code

Essentials :

1. **Classes :** Entities in the System
2. **Attributes :** properties or characteristics of a class
3. **Methods :** actions that a class can perform
4. **Visibility :** Where the Attributes and Methods are accessible

Visibility	Representation	Where Accessible ?
Public	+	Anywhere in the program
Private	-	in the class that defines it
Protected	#	Class that defines it and subclasses of that class
Package	~	Instances of other classes within same package

5. **Relationships :**

Relationship Type	Purpose
Associations	show how objects of one class interact with objects of another class
Inheritance	hierarchical relationship between classes
Aggregation	represents a whole-part relationship between two classes. indicates that an object of one class (whole) contains objects of another class (part). part can exist without whole
Composition	special type of aggregation that represents a strong whole-part relationship between two classes. part class can not exist without whole class

6. **Multiplicity** : Specify the number of instances of one class that are linked with the instances of another class
    a. One to One
    b. One to Many
    c. Many to One
    d. Many to Many

2. Object Diagram

Purpose : Provide a snapshot of a class diagram at a particular moment in time shows the Instances of the classes active at the given moment.

Essentials :

1. **Objects** : Instance of the class with Data

3. Component Diagram:

Purpose : Represent High level view of the system

Essentials :

1. **Components :** Modular and Independent part of the software system, that can be replaced or modified without affecting other components.
2. **Interfaces :** Represents a set of operations or services that a component offers to its clients.

    Provided interface

    Required Interface
3. **Port :** Point of interaction between a component and its environment
4. **Connectors :** specify how component interact

4. Package Diagram:

Purpose : Help to represent complex system , SHows us how our code is packaged

Essentials :

1. **Package :**container or grouping mechanism or namespace used to organize related elements such as classes, interfaces, other packages
2. **Subsystems :**group of related packages and elements that are organized together to perform a specific function or set of functions within a larger system
3. **Dependencies :**

   <<import>> — One package imports the functionality of another package.

   <<access>> — One package requires assistance from functions of another package.

5. Deployment Diagram :

Purpose : Represent the view of the Physical hardware or cloud component on which our software components are installed

Essentials :

1. **A node :**a physical or virtual element that is capable of executing hardware or software components.
2. **A component :** represents a modular, self-contained unit of software that can be deployed and executed on a node.
3. **An artifact** : represent any type of physical or digital resource, such as code files, configuration files, libraries, executables, scripts, database tables or documentation

6. Composite Structure diagram :

Purpose : illustrates the internal structure of a system or a component.

Essentials :

1. **Component** :internal structure of a class or component
2. **Part :** Internal structure of component

3. **Connector :** Use to associate the parts

7. Profile Diagram :

Purpose : Allow you to define, customize your own modeling language that is tailored to your specific needs.

Essentials :

1. **Stereotypes** : special annotation that can be applied to a UML element to modify its behavior or appearance.
2. **Tagged values** : a tagged value is a customizable attribute that can be associated with a UML element to provide additional information about the element
3. **Constraint :** used to specify requirements, business rules, or other criteria that must be satisfied by the model element.

B. Behavioral Diagrams

How the system interacts with external entities and users, how it responds to input or event and what constraints it operates under.

8. Use case Diagram

Purpose : Representation of a system's behavior from the perspective of the users.

Essentials :

1. **Actor :** an actor represents a person, group, or external system that interacts with the system being modeled
2. **Primary Actors :** main actor who initiates a use case and has a goal to achieve by interacting with the system.
3. **Secondary Actors :** is an actor who assists the primary actor in achieving their goal.
4. **Use case :** goal or task performed by the end user, use case represent the action performed

a. **Business Use case :**some functionality or service that is offered to the customers, business partners or other business systems .

b. **System Use case :**functionality that exists within a business system, which is neither visible nor accessible
to outsiders, represents an internal activity, meaning an internal business process.

5. **System boundary :**A system boundary in a use case diagram represents the boundary or scope of the system being modeled.
6. **Relationships**
   a. **Association :**represents a communication between an actor and a use case.
   b. **Generalization** : represents an inheritance relationship between two use cases.
   c. **Include :** represents an invocation of one use case by another use case.
   d. **Extend :**represents an optional or alternative functionality of a use case.
   e. **Dependency :**represents a dependency between two use cases or between a use case and an actor.

9. Activity Diagram :

Purpose : used to model the flow of actions, activities, and decisions in a system or process.

Essentials

1. **Activities :** represent the steps or actions that are taken in the system or process.
2. **Initial State :**beginning of the set of actions
3. **Final state :** End of set of actions
4. **Control Flow :** show the order in which activities are executed in the system or process
5. **Decision Nodes :** show conditional logic in the system or process.
6. **Fork Nodes :**used to split a single control flow into multiple flows.
7. **Merge Nodes :**used to combine two or more incoming control flows into a single outgoing flow

8. **Join Nodes:** used to synchronize two or more incoming control flows into a single outgoing flow.
9. **Swim lanes :**divide the activities or actions into categories or groups.

10. State Machine Diagram :

Purpose : used to model the different states of the object in a system

Essentials :

1. **States :**represents a moment of an object or system at a given time in its lifespan.
2. **Transitions :** represent a change in the state of an object or system in response to an event or condition
3. **Event :**Trigger or a stimulus to perform an action.
4. **Guards** :conditions or constraints that must be met in order for a transition to occur.

11. Sequence Diagram :

Purpose :shows how objects interact in a step-by-step process over time or in sequential order,

Essentials :

1. **Lifeline :**represents an object or participant in the system
2. **Activation:** describes that time period in which an operation is performed by an element,
3. **Messages :**messages are basically a method calls ,The messages depict the interaction between the objects
    - **Call Message** one object is invoking an operation on another object.
    - **Return Message** represent the response of an object to a previously received message.
    - **Self Message** represent the invocation of a method or operation by an object on itself.
    - **Recursive Message :** A self message sent for recursive purpose

- **Create Message** message is used to represent the creation of a new object
- **Destroy/Delete Message :**used to represent the destruction or deletion of an object or life
- **Synchronous message** A message that requires a reply from the receiver, and the sender must wait for the reply before continuing
- **Asynchronous message** A message that requires the reply from the receiver but the there is not need of sender to wait for the reply

4. **Message constraint**
    a. **Guard conditions:** Express conditions that must be met before a message can be sent.
    b. **Timing constraints:** Specify time limits for sending or processing messages.
    c. **Ordering constraints:** Specify the order in which messages must be sent or processed.
    d. **Frequency constraints:** Specify how many times a message can be sent.
    e. **Iteration constraints:** Specify the number of times a message can be sent in a loop.

12. Communication Diagram :

Purpose : same information as shown by the sequence diagram but pay more emphasis on interaction messages between the objects or parts of the system.
Essentials :

1. **Objects/ lifelines :** components or entities that participate in the communication.
2. **Connector :** line which connects the two objects together.
3. **Messages:** information packets that are passed between the objects.

13. Timing Diagram :

Purpose: model the behavior of objects or components over a certain period of time and to depict the order in which events occur in a system.

Essentials :

1. **Lifeline:** depicts a single element within the interaction
2. **State/condition timeline :**represents the set of valid states the object goes through.
3. **Message :**represents the communication or interaction between two objects or components state
4. **Duration Constraint :**constraint of an interval
5. **Time Constraint** constraint of time

14 Interaction Overview Diagram

Purpose :form of activity diagram in which the nodes represent interaction diagrams

Essentials :

1. **Interaction :** represents a behavior that is performed by the system or application.
2. **Interaction use :**represent the reuse of an existing interaction within an Interaction Overview Diagram.