

# Day-1

SDLC Overview, Models & Modelling, Agile, Object Oriented Design & Programming,

# Agenda

- SDLC Overview
- Models & Modelling
- Agile
- Object Oriented Design & Programming

# SDLC

- SDLC stands for Software Development Life Cycle.
- SDLC is a process that consists of a series of planned activities to develop or alter the Software Products.
- A life cycle model represents all the methods required to make a software product transit through its life cycle stages.
- **SDLC** is a systematic process for building software that ensures the quality and correctness of the software built.
- SDLC process aims to produce high-quality software that meets customer expectations.
- The system development should be complete in the pre-defined time frame and cost.

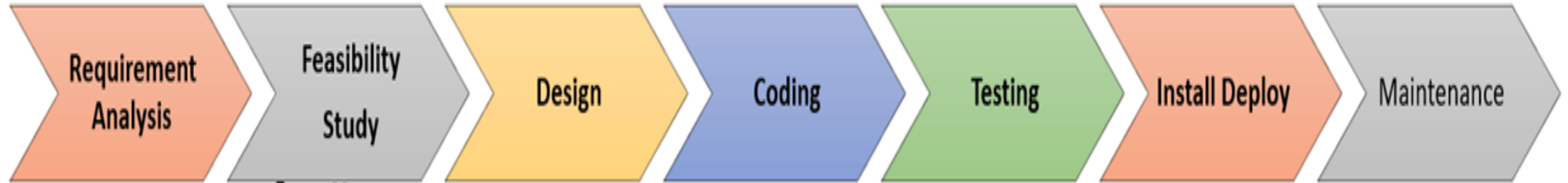
- SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software.
- Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.
- SDLC stands for **Software Development Life Cycle** and is also referred to as the Application Development life-cycle

# Why SDLC

- It offers a basis for project planning, scheduling, and estimating
- Provides a framework for a standard set of activities and deliverables
- It is a mechanism for project tracking and control
- Increases visibility of project planning to all involved stakeholders of the development process
- Increased and enhance development speed
- Improved client relations
- Helps you to decrease project risk and project management plan overhead

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software.
- The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- SDLC is a process followed for a software project, within a software organization.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.

# SDLC Phases



© guru99.com





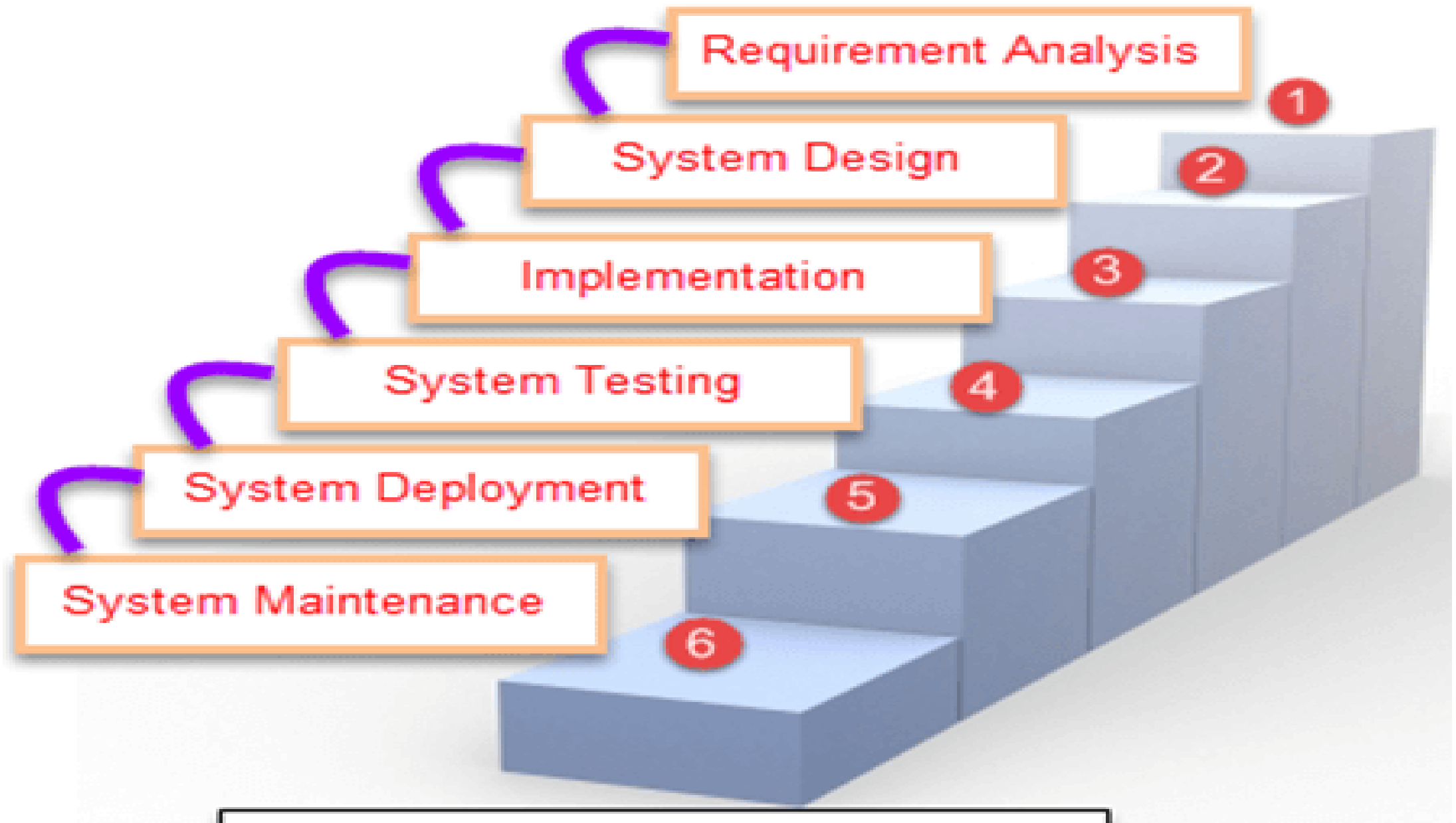
- Phase 1: Requirement collection and analysis
- Phase 2: Feasibility study:
- Phase 3: Design:
- Phase 4: Coding:
- Phase 5: Testing:
- Phase 6: Installation/Deployment:
- Phase 7: Maintenance

# Popular SDLC models

- Waterfall model in SDLC
- Incremental Model in SDLC
- V-Model in SDLC
- Agile Model in SDLC
- Spiral Model
- Big Bang Model
- RAD Model

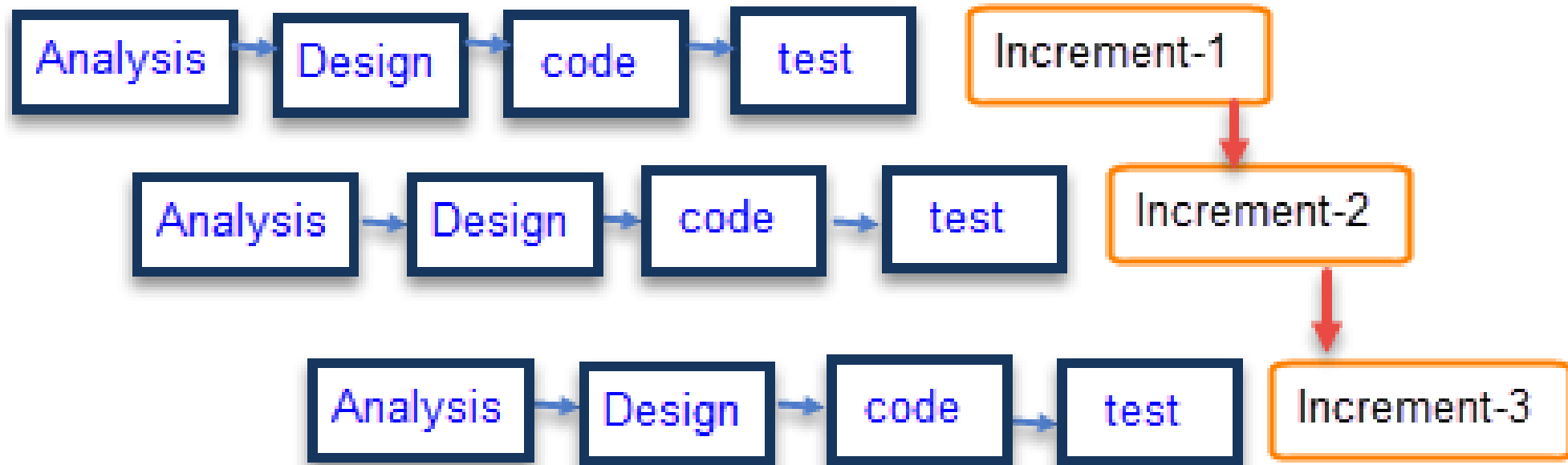
# Waterfall model in SDLC

- **WATERFALL MODEL** is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.
- Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project.
- In "The Waterfall" approach, the whole process of software development is divided into separate phases.



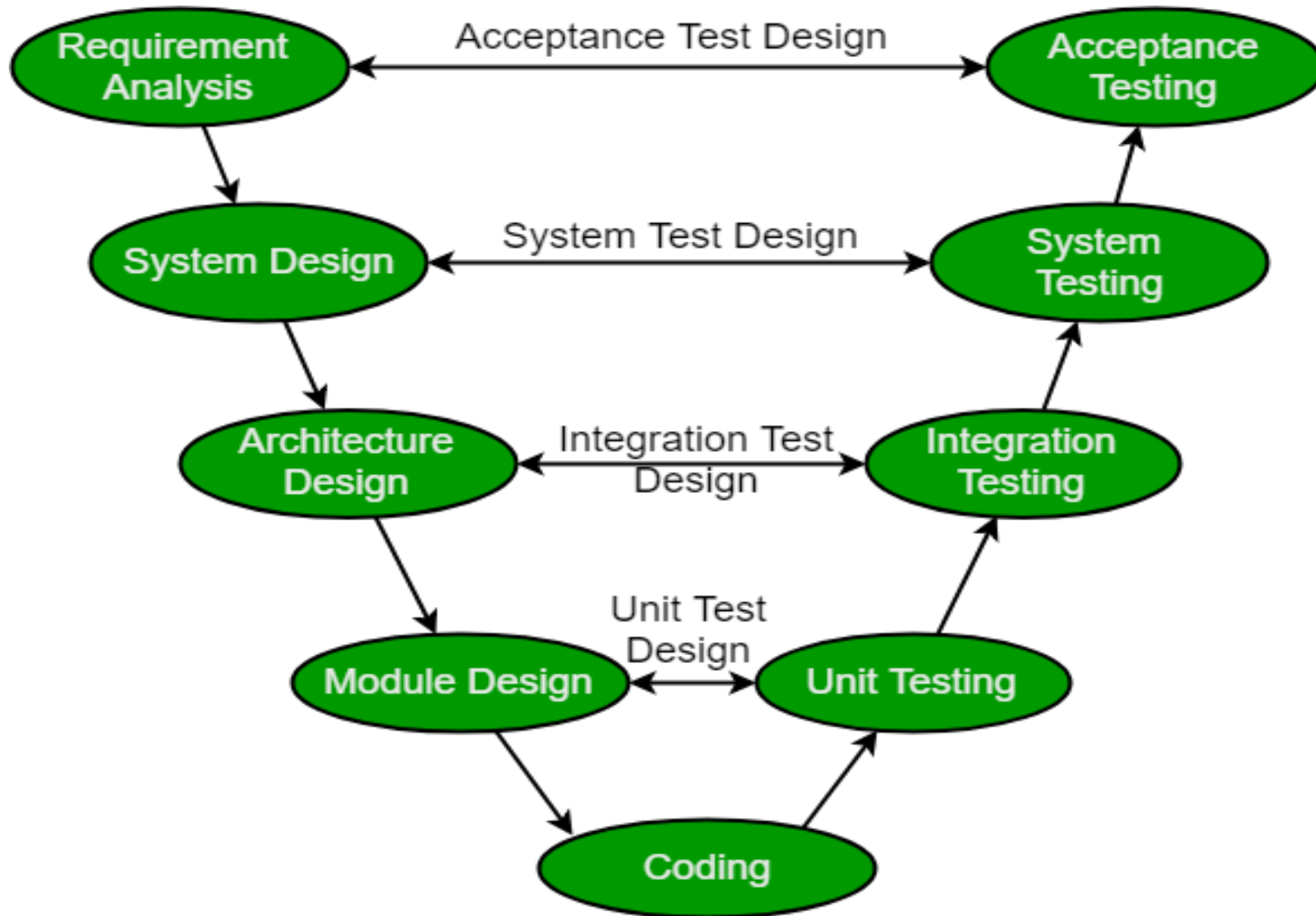
Waterfall Model

# Incremental Model in SDLC



Incremental Model

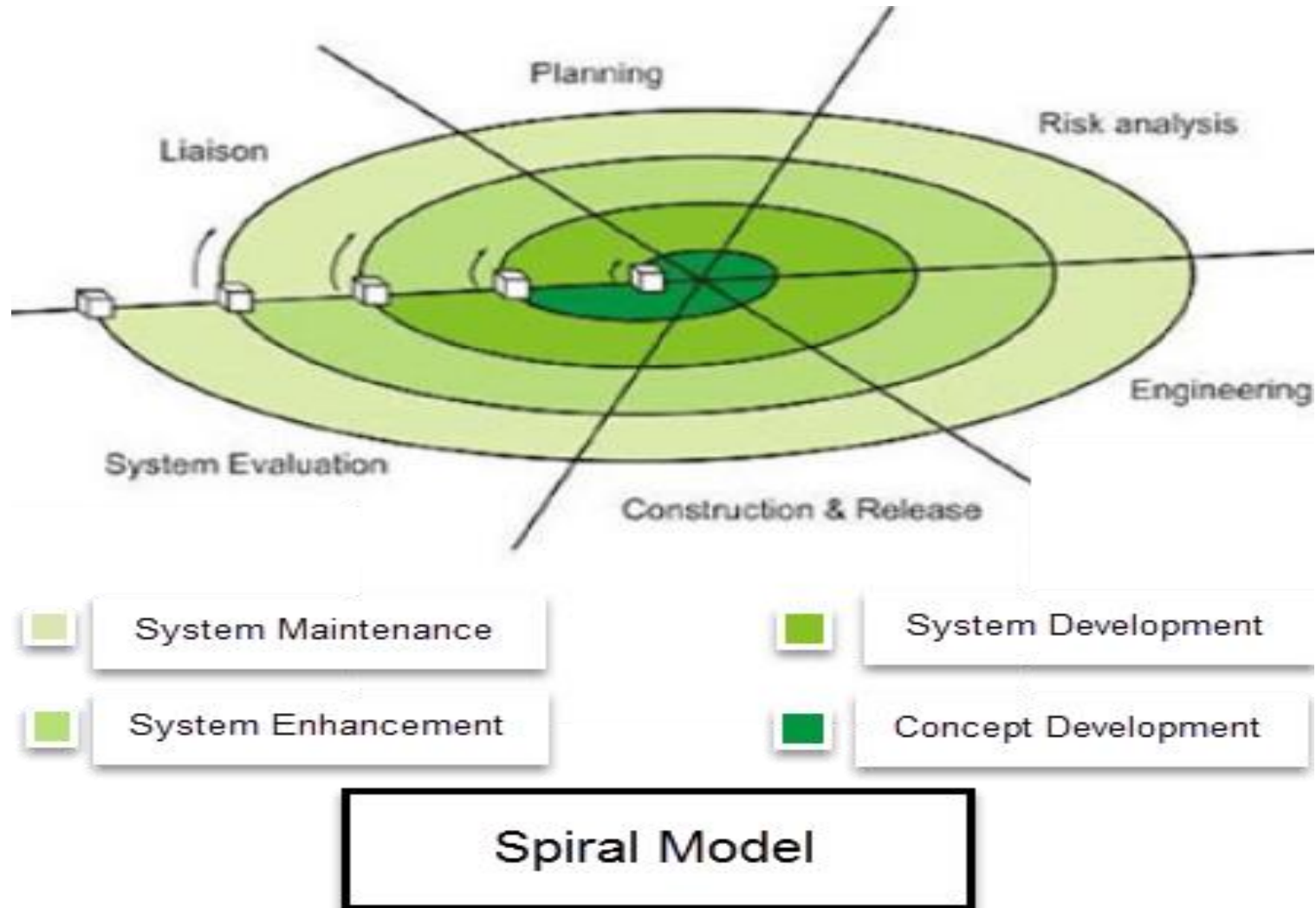
# V-Model in SDLC



# Agile Model in SDLC

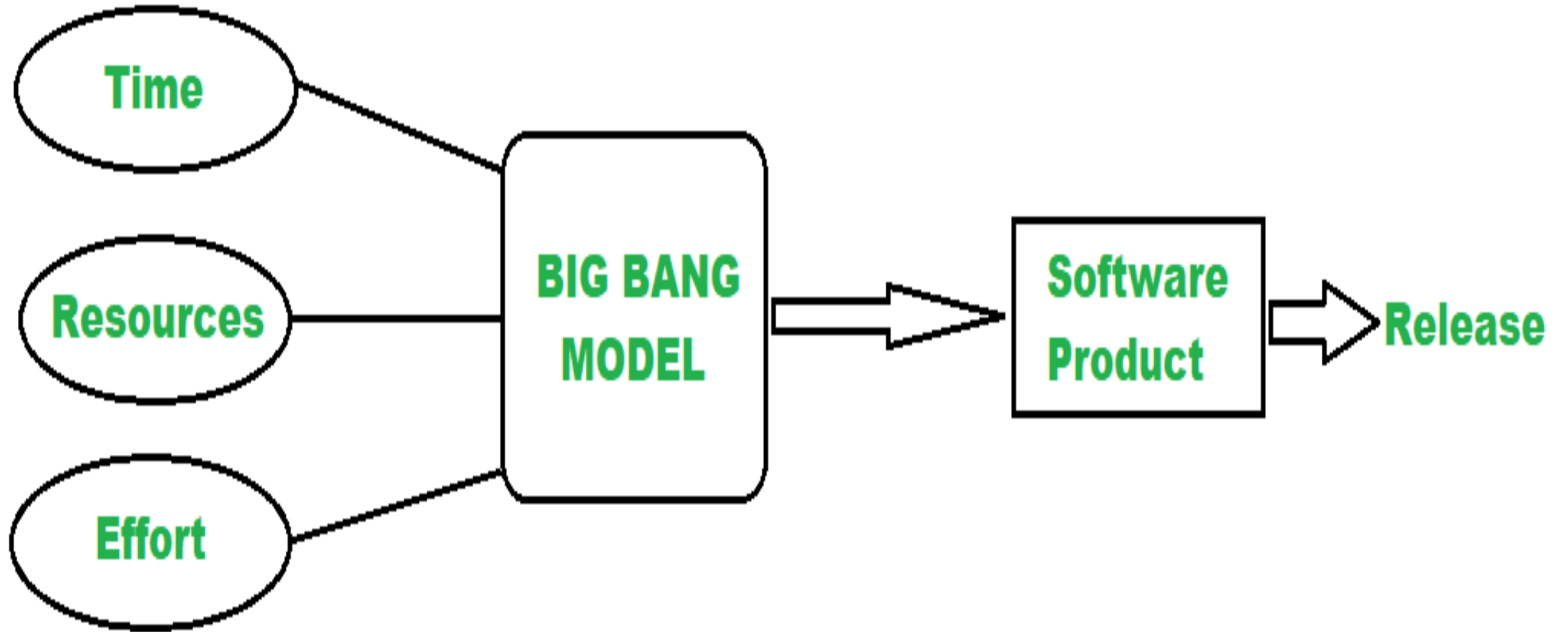


# Spiral Model

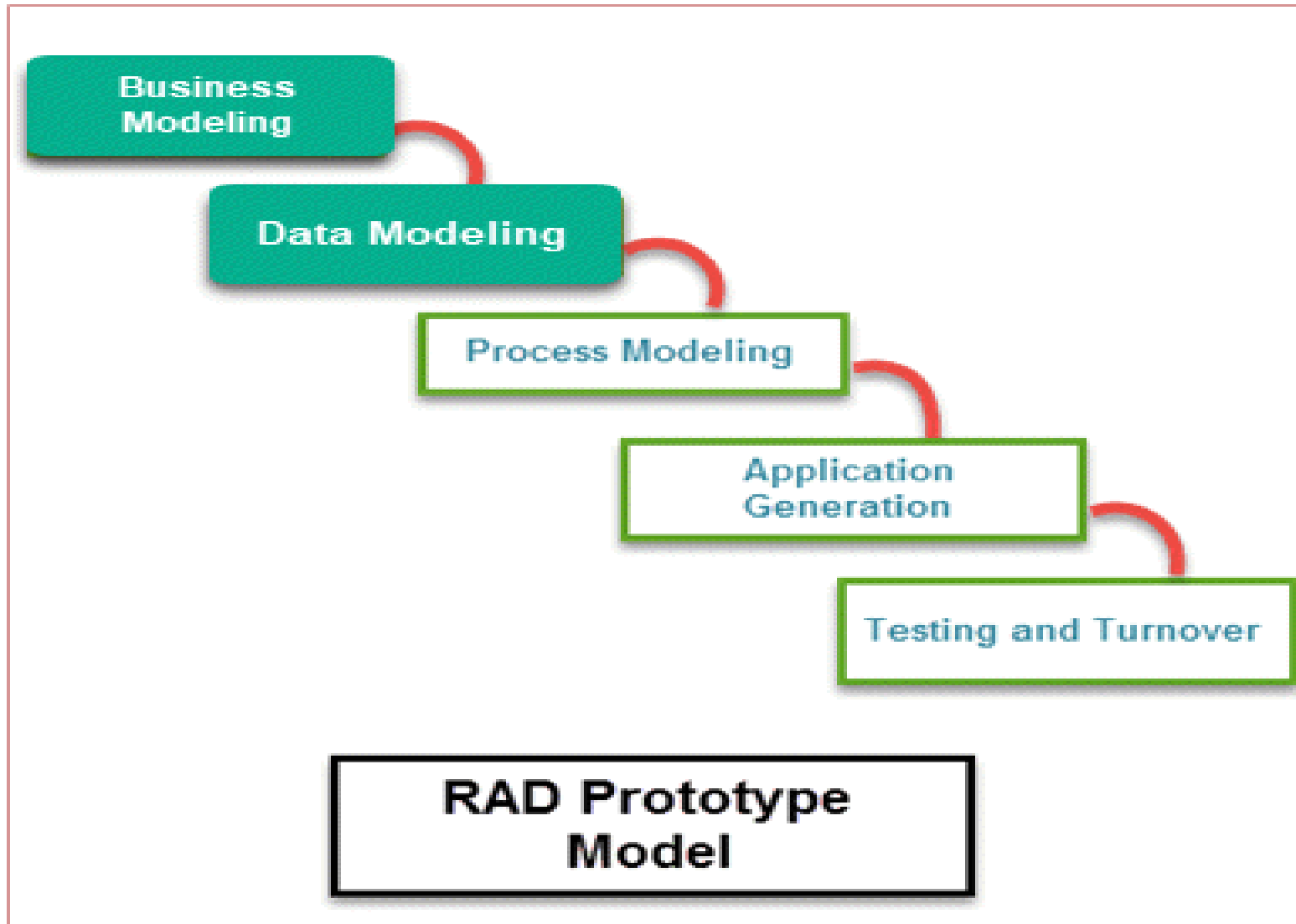




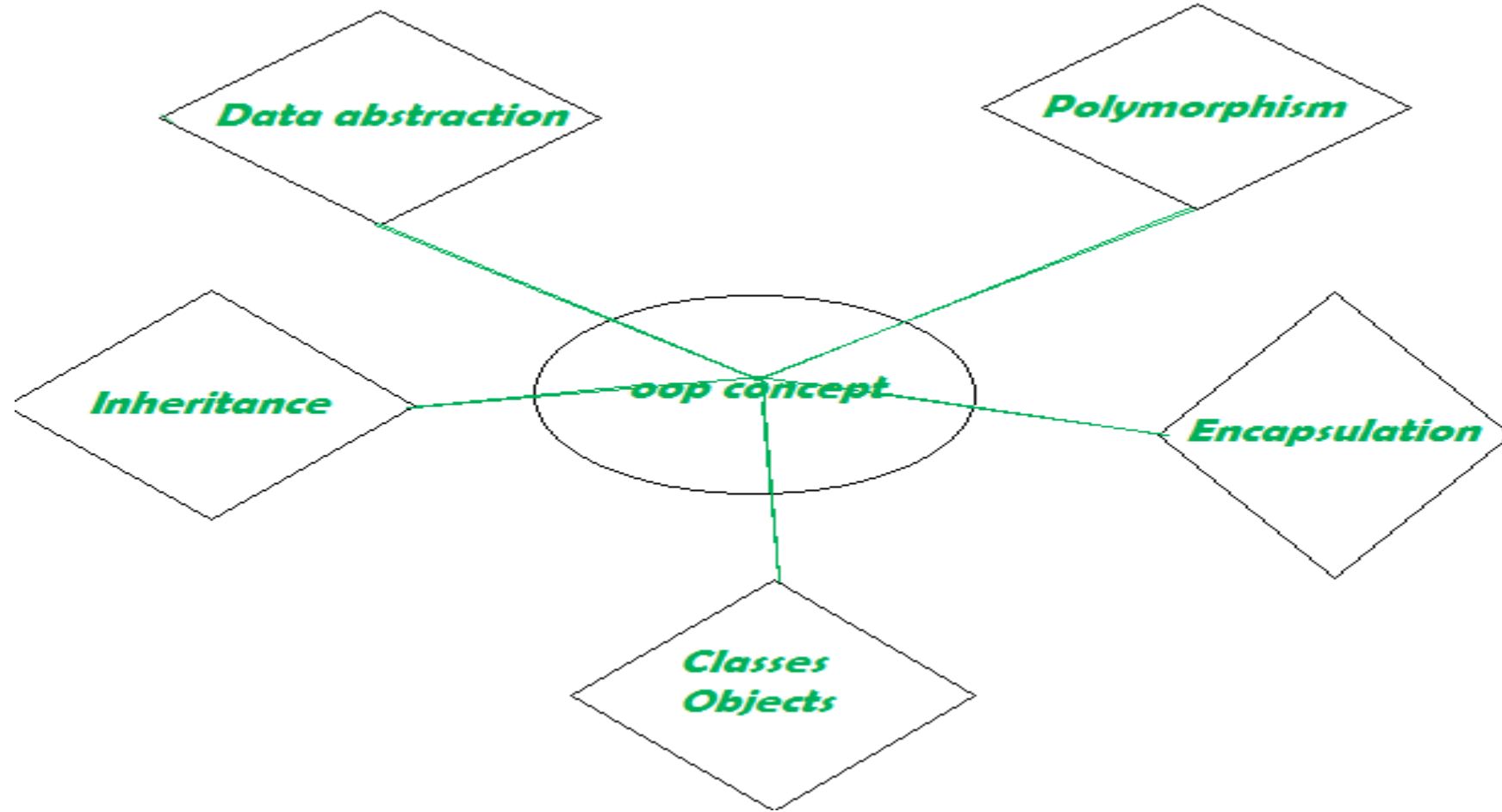
# Big bang model



# RAD Model



# Object Oriented Design & Programming



# UML Diagrams

# UML Diagrams

- A **Class in UML** diagram is a blueprint used to create an object or set of objects. The Class defines what an object can do.
- It is a template to create various objects and implement their behaviour in the system.
- A Class in UML is represented by a rectangle that includes rows with class names, attributes, and operations.
- UML (Unified Modelling Language) is a general-purpose, graphical modelling language in the field of Software Engineering. UML is used to specify, visualize, construct, and document the artifacts (major elements) of the software system.

# What is UML

- The UML stands for Unified modeling language, is a standardized general-purpose visual modeling language in the field of Software Engineering.
- It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system.
- It helps in designing and characterizing, especially those software systems that incorporate the concept of Object orientation. It describes the working of both the software and hardware systems.
- The Object Management Group (OMG) is an association of several companies that controls the open standard UML.
- The OMG was established to build an open standard that mainly supports the interoperability of object-oriented systems.

- It is not restricted within the boundaries, but it can also be utilized for modelling the non-software systems.
- The OMG is best recognized for the Common Object Request Broker Architecture (CORBA) standards.
- UML is a type of diagram/ static structure diagram that describes the structure of system by showing system's classes, their attributes, operations (or methods), and the relationship among objects.

# Goals of UML

- Since it is a general-purpose modeling language, it can be utilized by all the modelers.
- UML came into existence after the introduction of object-oriented concepts to systemize and consolidate the object-oriented development, due to the absence of standard methods at that time.
- The UML diagrams are made for business users, developers, ordinary people, or anyone who is looking forward to understand the system, such that the system can be software or non-software.
- Thus it can be concluded that the UML is a simple modelling approach that is used to model all the practical systems.



# Characteristics of UML

The UML has the following features:

- It is a generalized modeling language.
- It is distinct from other programming languages like C++, Python, etc.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts.

# Conceptual Modelling

- A conceptual model is composed of several interrelated concepts. It makes it easy to understand the objects and how they interact with each other. This is the first step before drawing UML diagrams.
- Following are some object-oriented concepts that are needed to begin with UML:
  - Object
  - Class
  - Abstraction
  - Inheritance
  - Polymorphism
  - Encapsulation

# Basic Notations

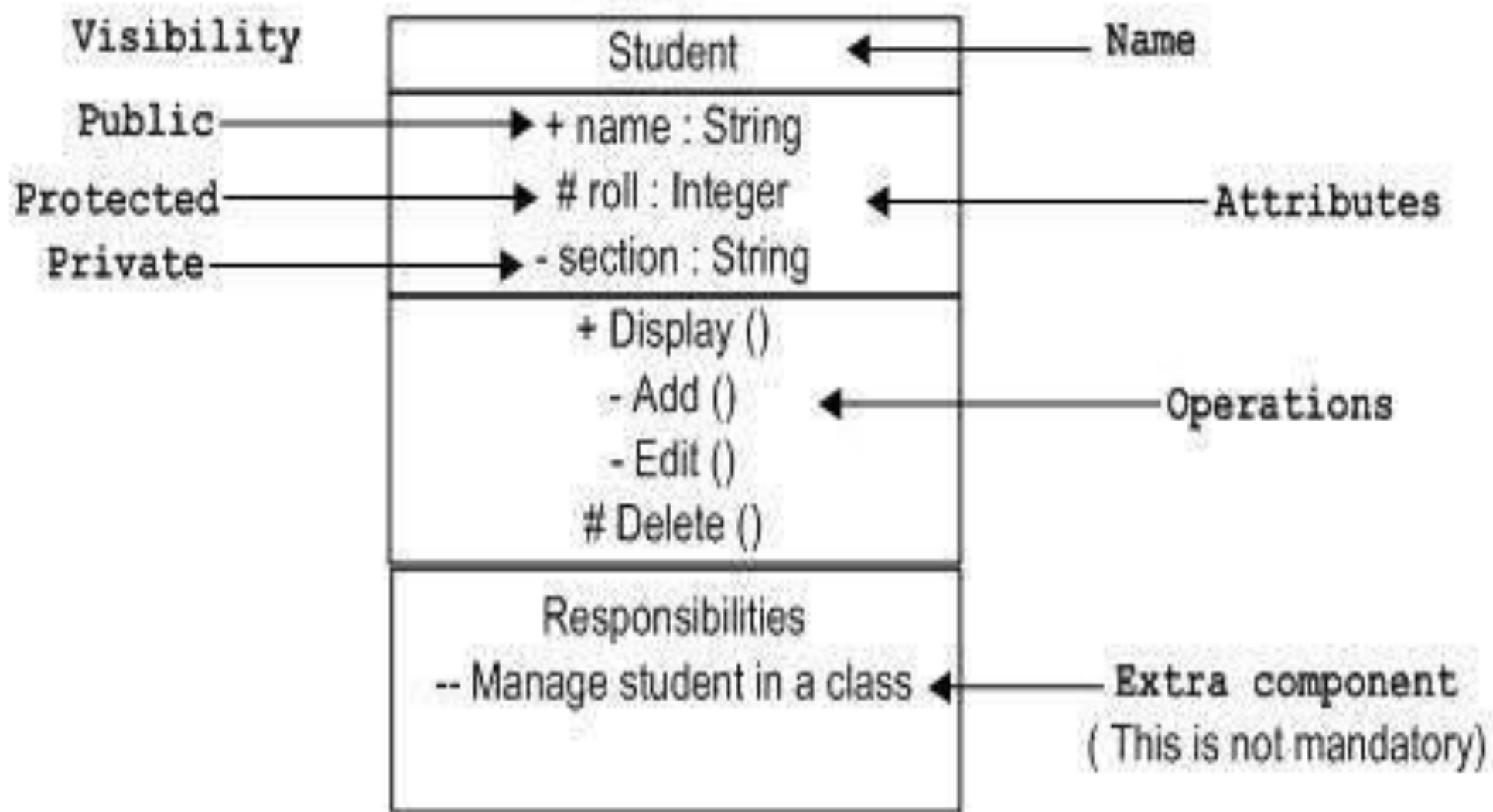
- Classes
- Object
- Interface
- Collaboration
- Use case
- Active classes
- Components
- Nodes

# Class Notation

UML *class* is represented by the following figure. The diagram is divided into four parts.

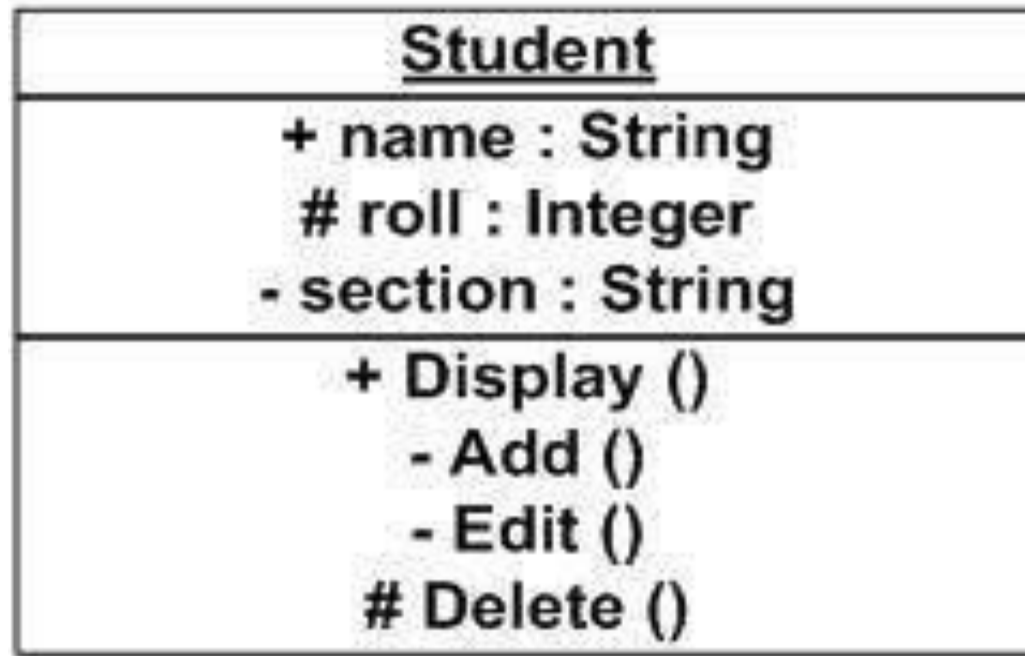
- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.

## Class



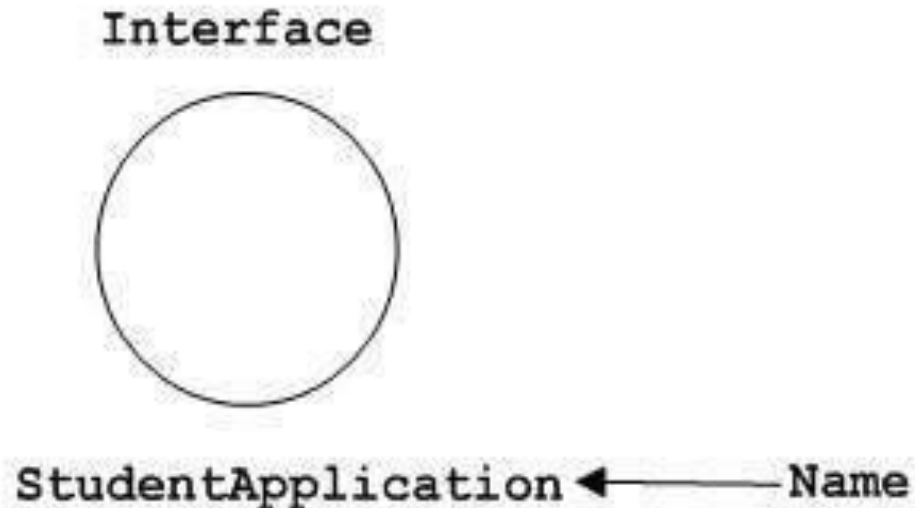
# Object Notation

- The *object* is represented in the same way as the class.
- As the object is an actual implementation of a class, which is known as the instance of a class. Hence, it has the same usage as the class.



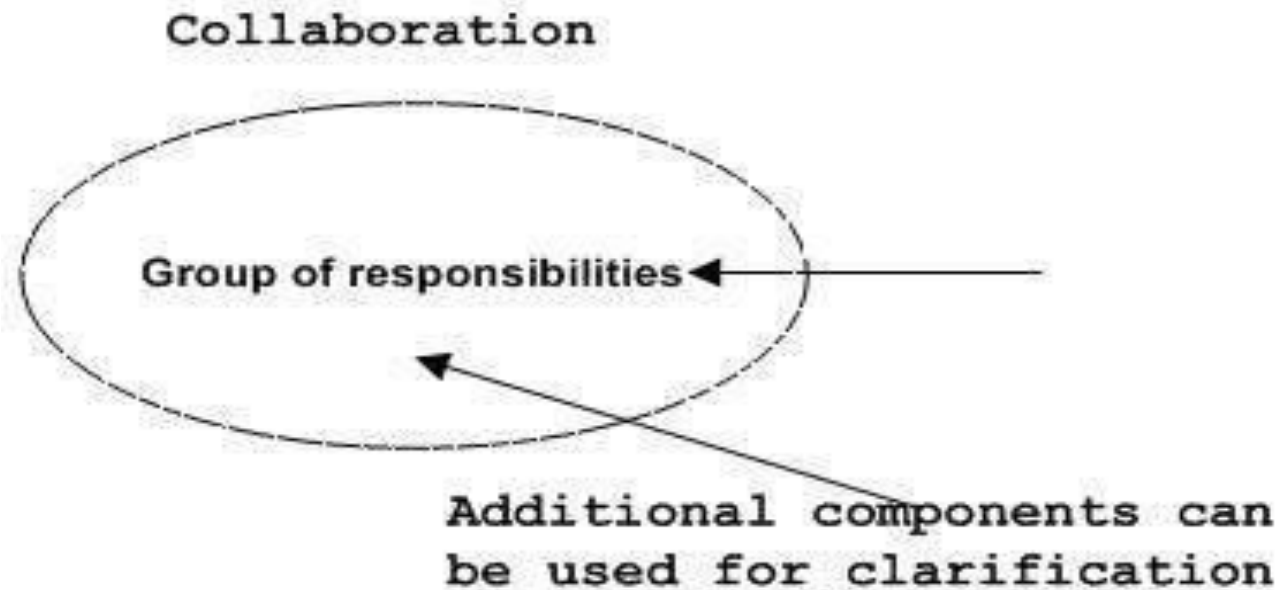
# Interface Notation

- Interface is represented by a circle as shown in the following figure. It has a name which is generally written below the circle.
- Interface is used to describe the functionality without implementation. Interface is just like a template where you define different functions, not the implementation. When a class implements the interface, it also implements the functionality as per requirement.



# Collaboration Notation

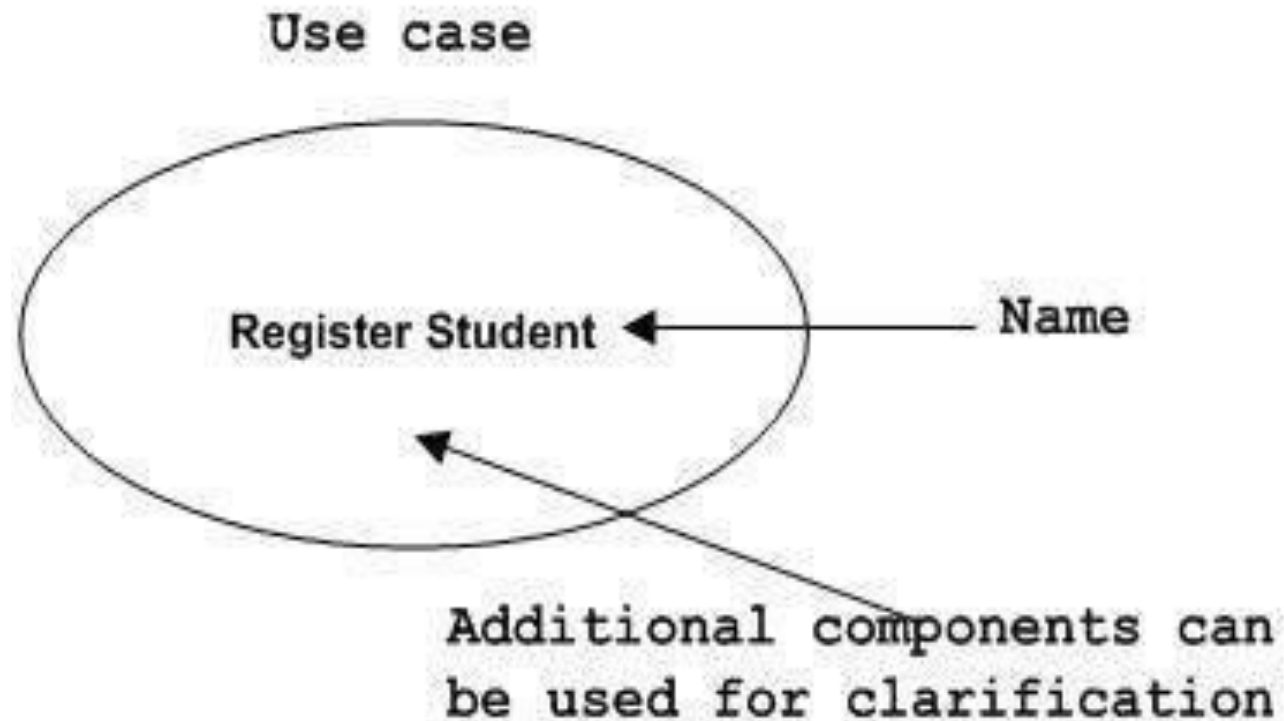
- Collaboration is represented by a dotted ellipse as shown in the following figure. It has a name written inside the ellipse.
- Collaboration represents responsibilities. Generally, responsibilities are in a group.





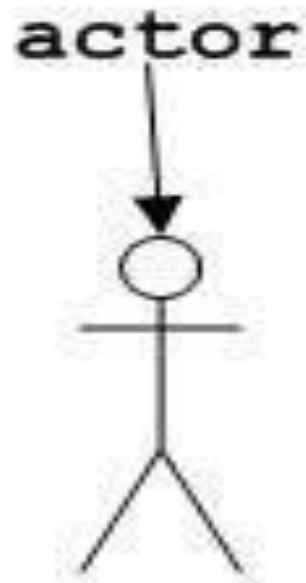
# Use Case Notation

- Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.
- Use case is used to capture high level functionalities of a system.



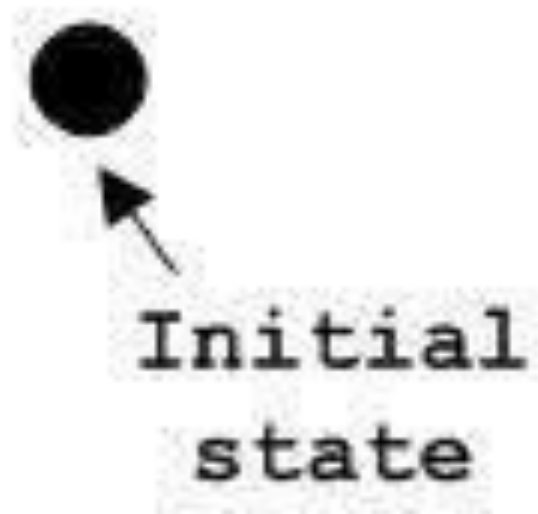
# Actor Notation

- An actor can be defined as some internal or external entity that interacts with the system.
- An actor is used in a use case diagram to describe the internal or external entities.



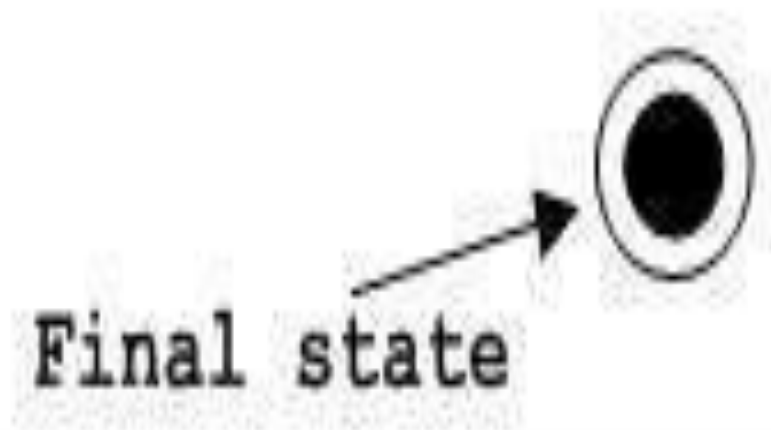
# Initial State Notation

- Initial state is defined to show the start of a process. This notation is used in almost all diagrams.
- The usage of Initial State Notation is to show the starting point of a process.



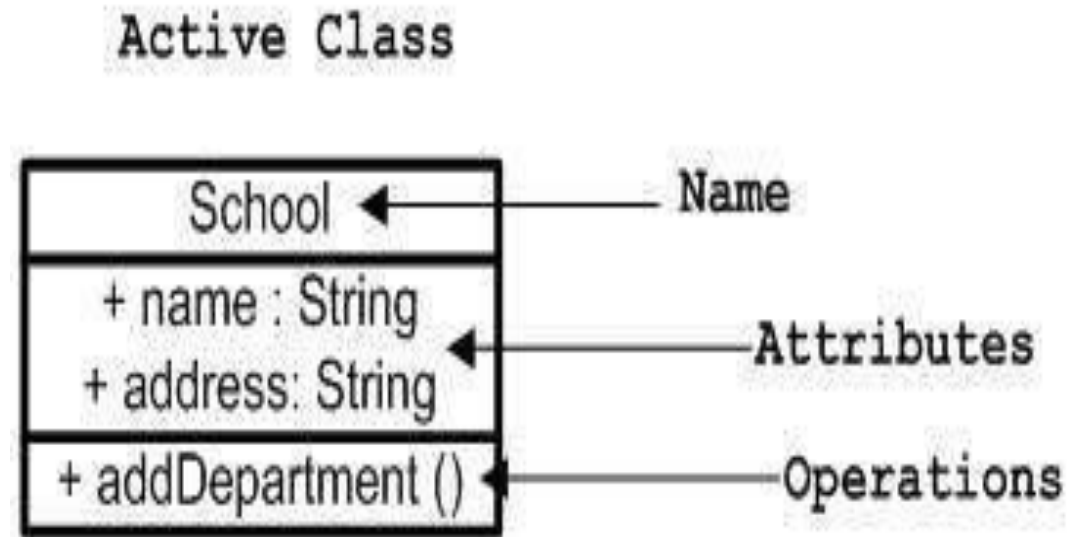
# Final State Notation

- Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.
- The usage of Final State Notation is to show the termination point of a process.



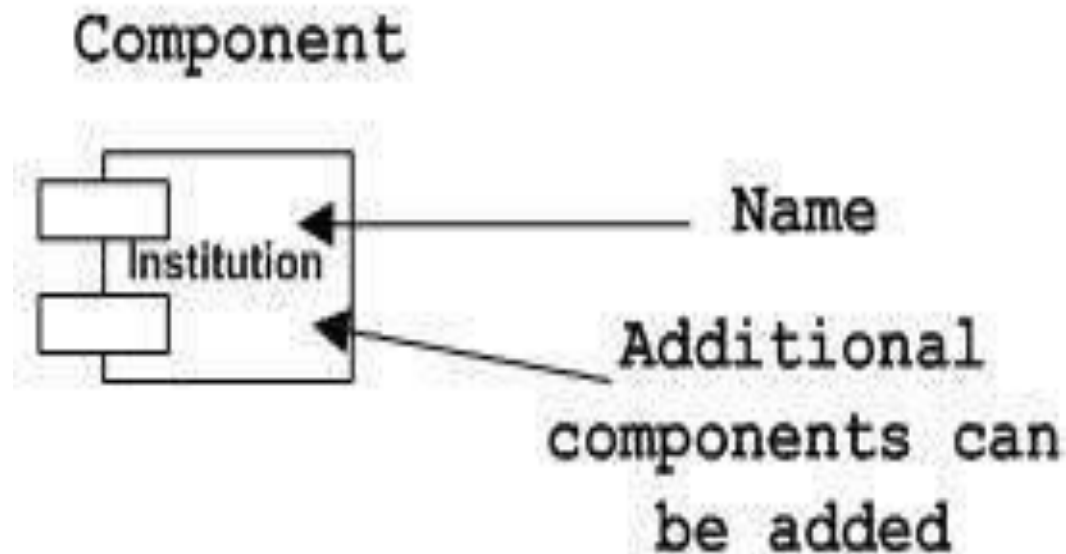
# Active Class Notation

- Active class looks similar to a class with a solid border. Active class is generally used to describe the concurrent behavior of a system.
- Active class is used to represent the concurrency in a system.



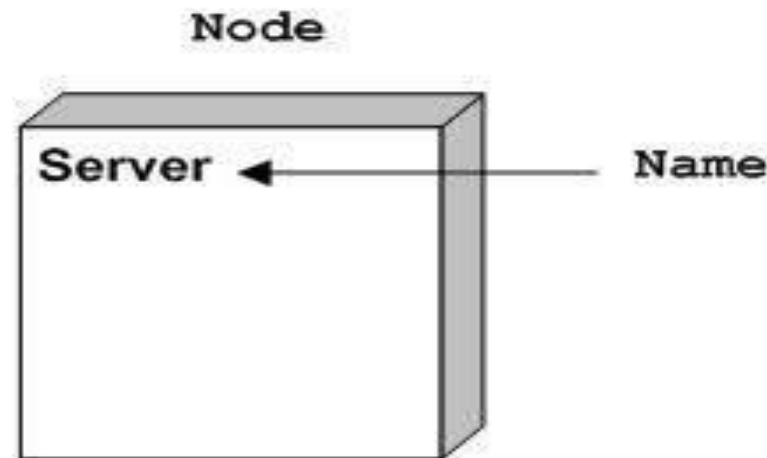
# Component Notation

- A component in UML is shown in the following figure with a name inside. Additional elements can be added wherever required.
- Component is used to represent any part of a system for which UML diagrams are made.

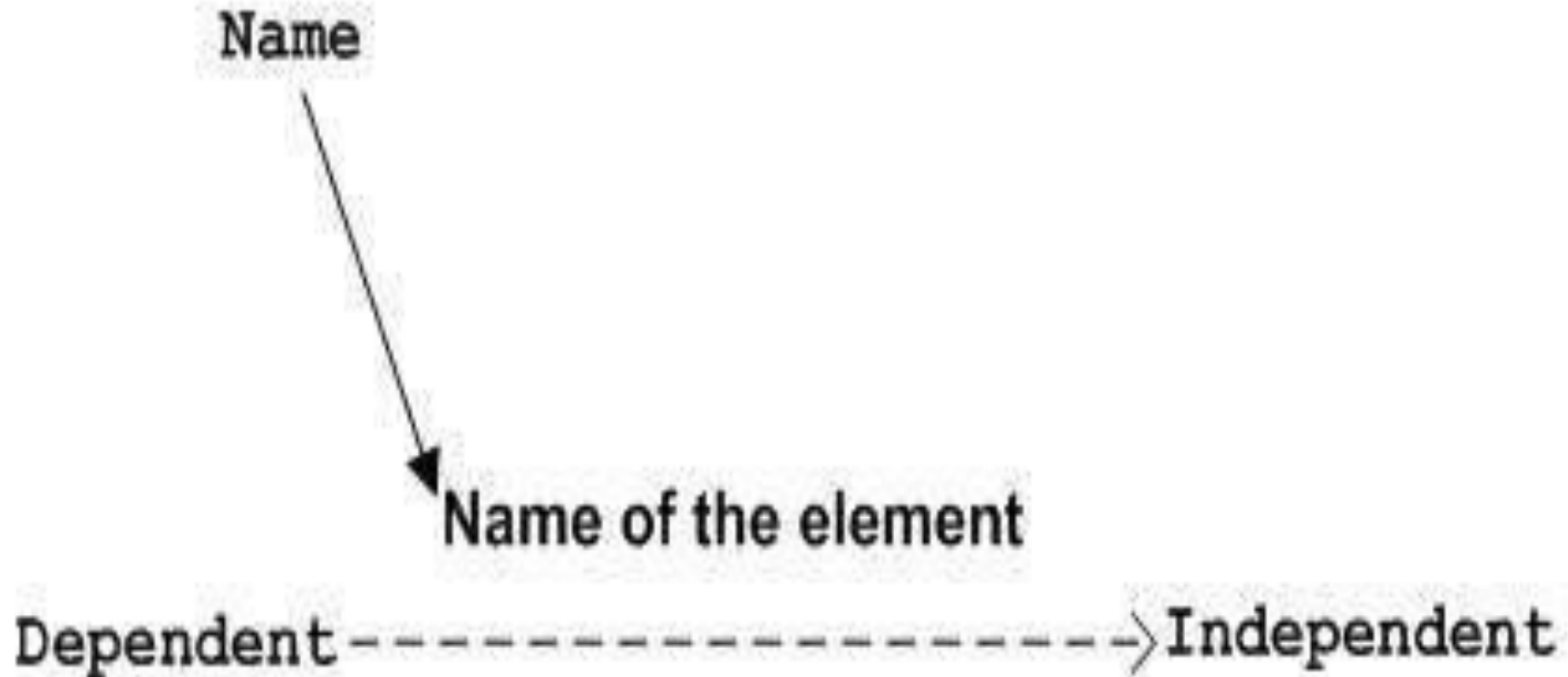


# Node Notation






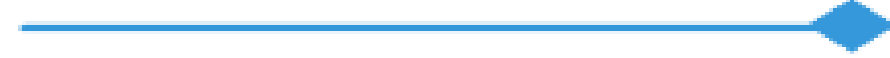
- A node in UML is represented by a square box as shown in the following figure with a name. A node represents the physical component of the system.
- Node is used to represent the physical part of a system such as the server, network, etc.



# Dependency Notation





Class Diagram Relationship Type	Notation
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	

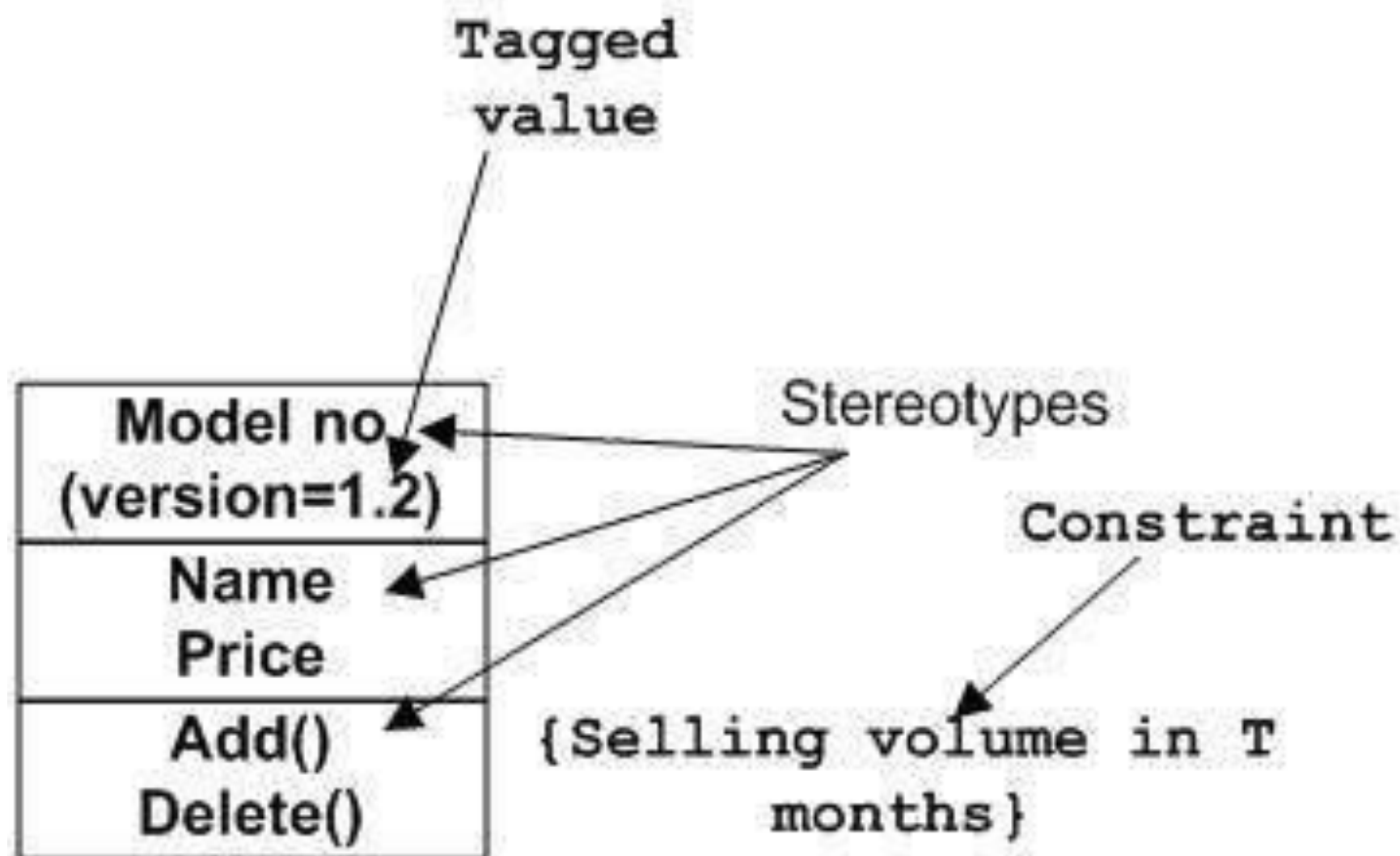
# Generalization Notation

- Generalization describes the inheritance relationship of the object-oriented world. It is a parent and child relationship.
- Generalization is represented by an arrow with a hollow arrow head as shown in the following figure. One end represents the parent element and the other end represents the child element.



# Extensibility Notation

- All the languages (programming or modeling) have some mechanism to extend its capabilities such as syntax, semantics, etc. UML also has the following mechanisms to provide extensibility features.
  - Stereotypes (Represents new elements)
  - Tagged values (Represents new attributes)
  - Constraints (Represents the boundaries)
  
- Extensibility notations are used to enhance the power of the language. It is basically additional elements used to represent some extra behavior of the system. These extra behaviors are not covered by the standard available notations.



# **Types of UML Diagrams**

# Types of UML Diagrams

## Structure Diagrams

- **Class Diagram**
- Component Diagram
- **Deployment Diagram**
- Object Diagram
- Package Diagram
- Profile Diagram
- Composite Structure Diagram

## Behavioural Diagrams

- **Use Case Diagram**
- Activity Diagram
- **State Machine Diagram**
- **Sequence Diagram**
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram

# Use Case Diagrams

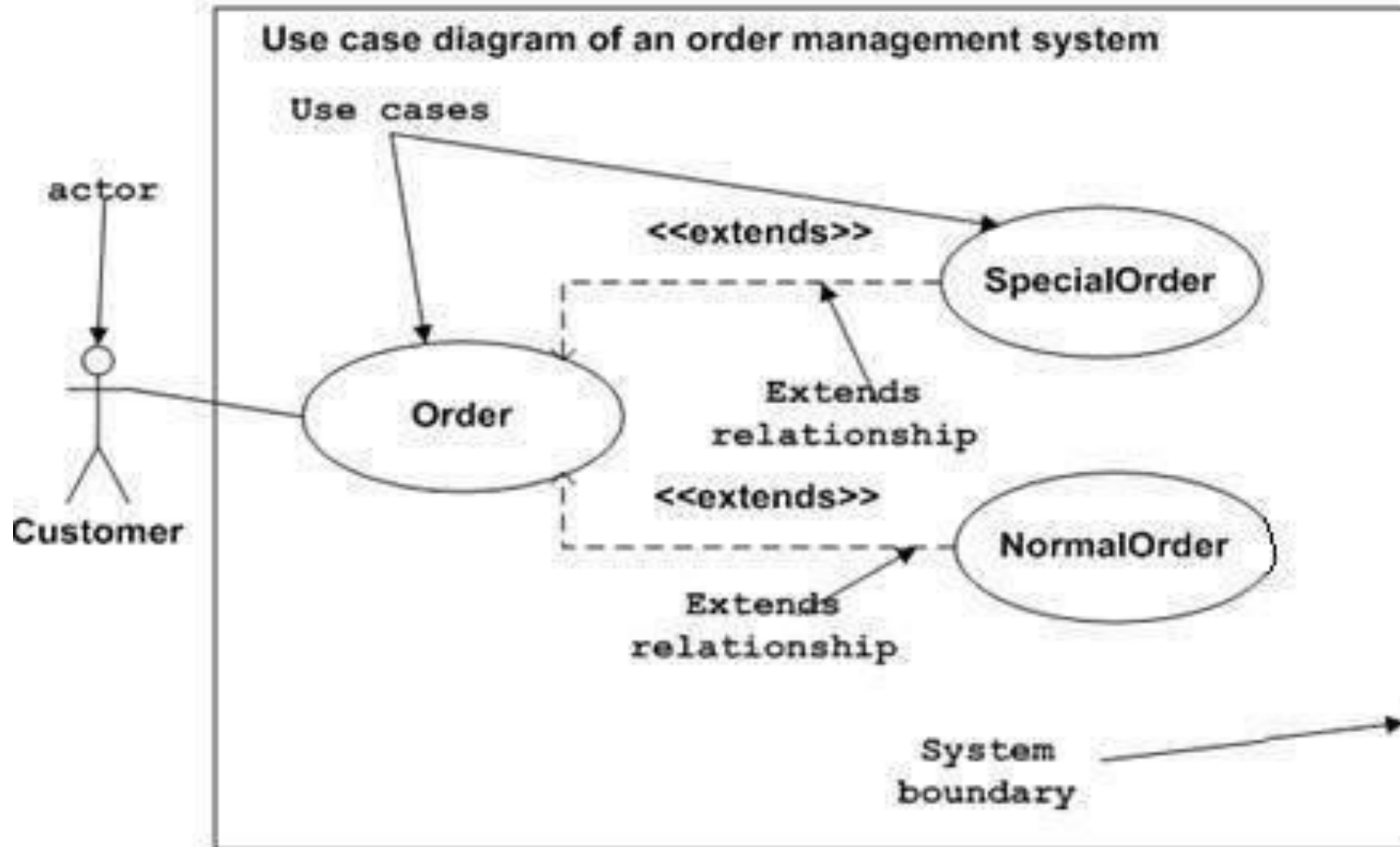
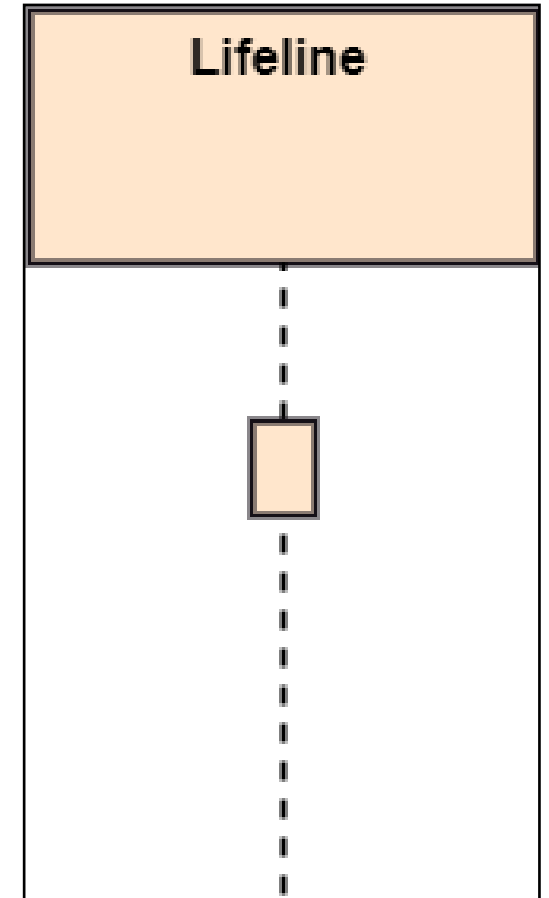
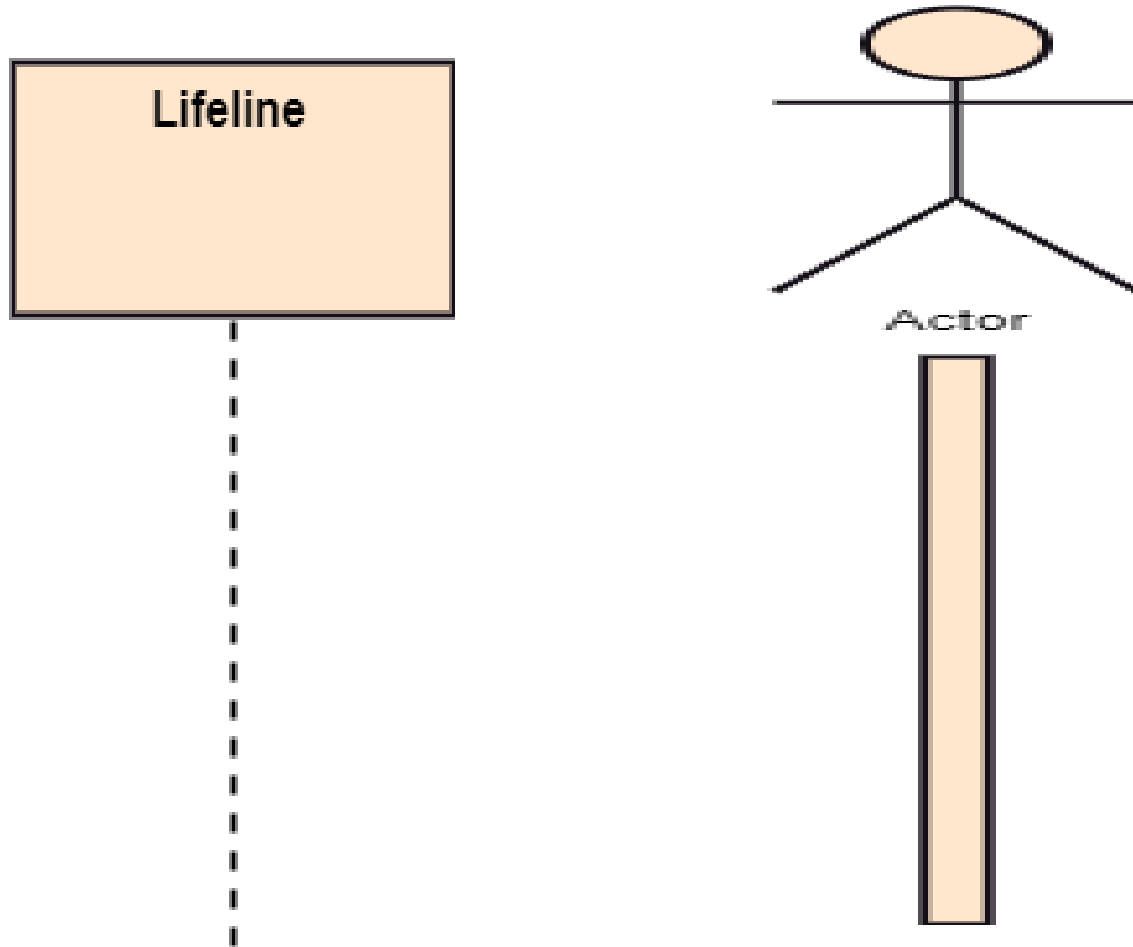


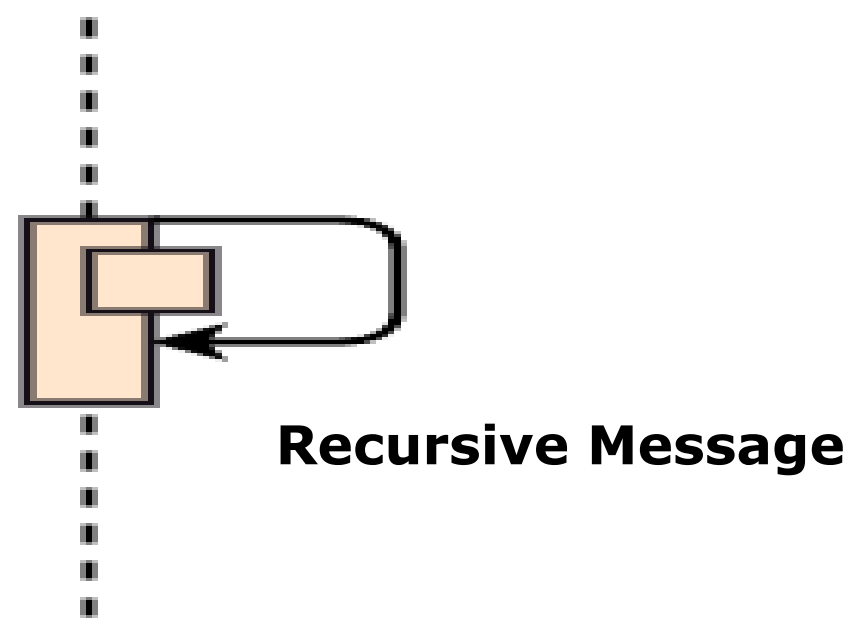
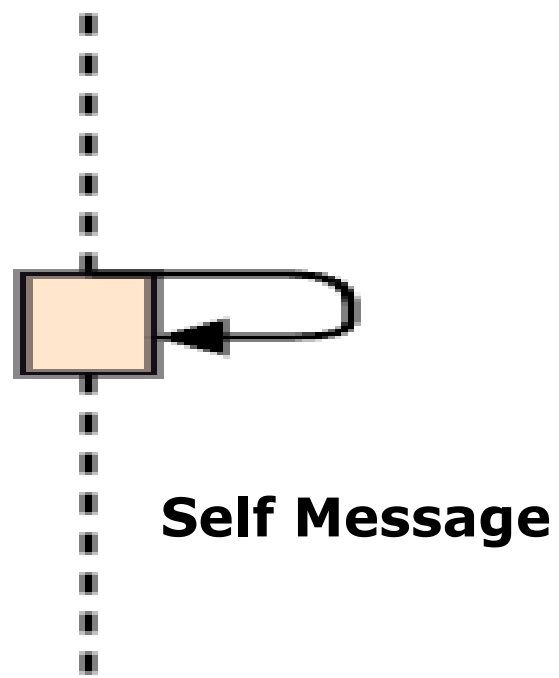
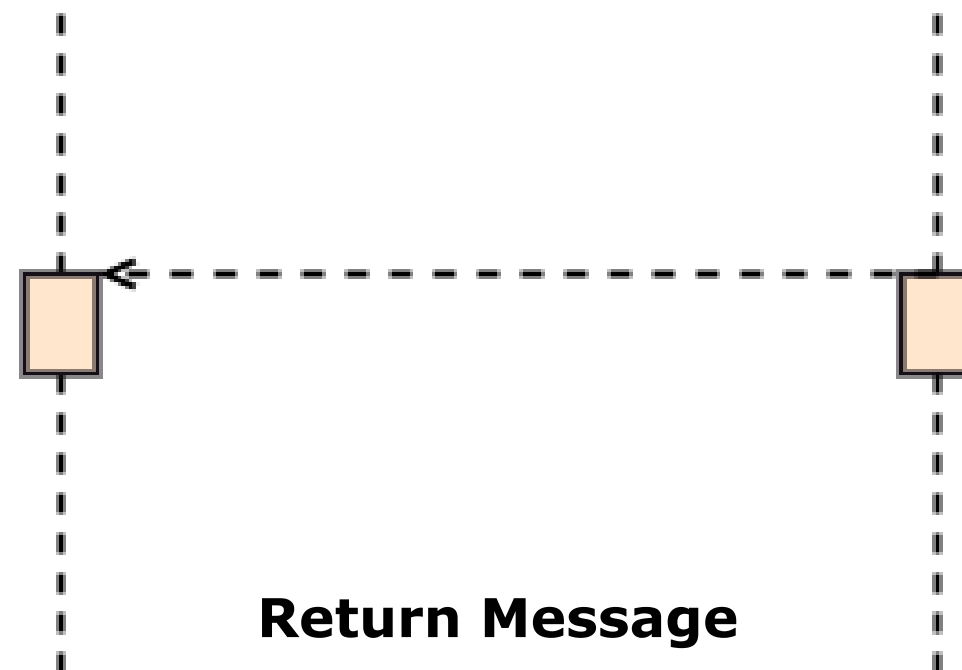
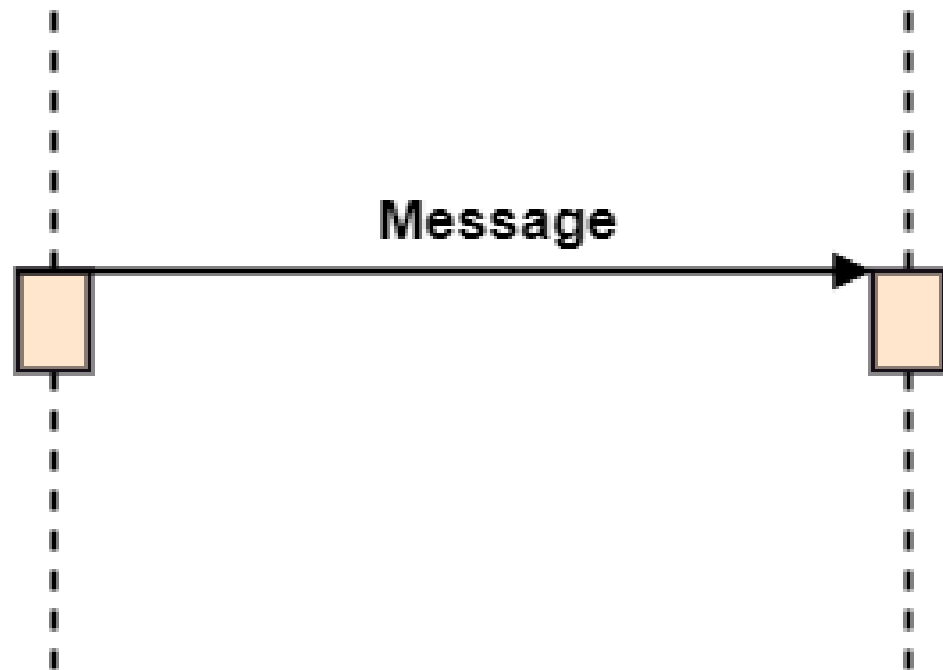
Figure: Sample Use Case diagram

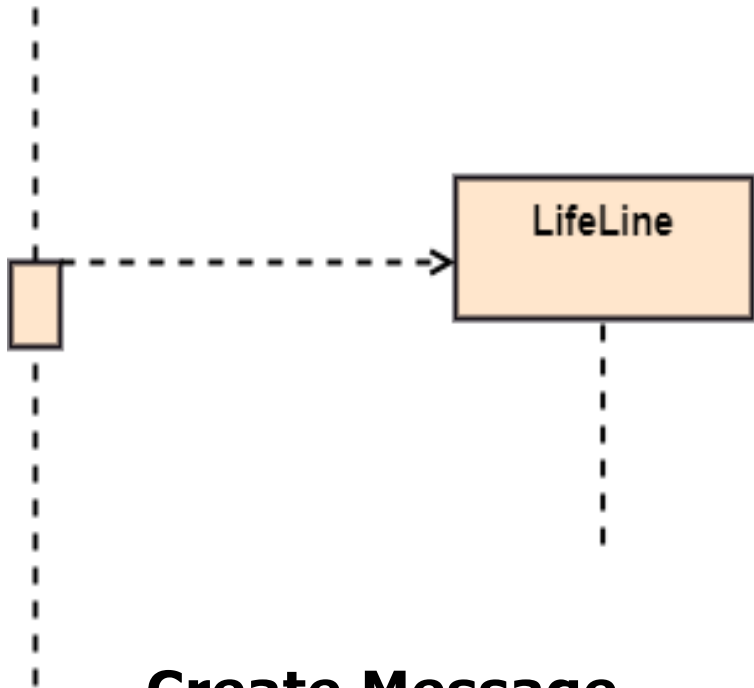
# Sequence Diagram



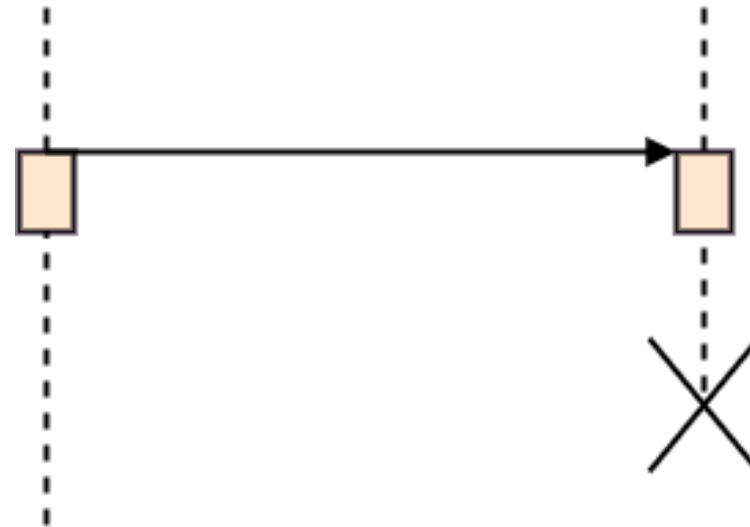
**Activation**



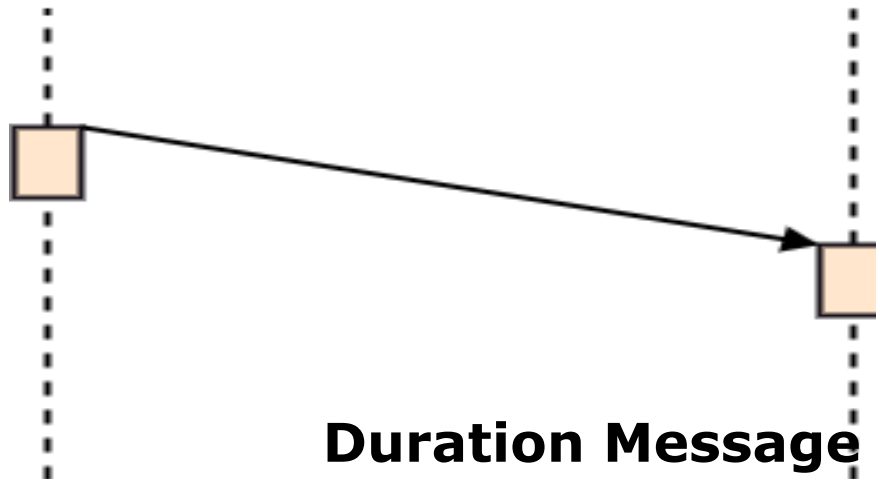




**Create Message**

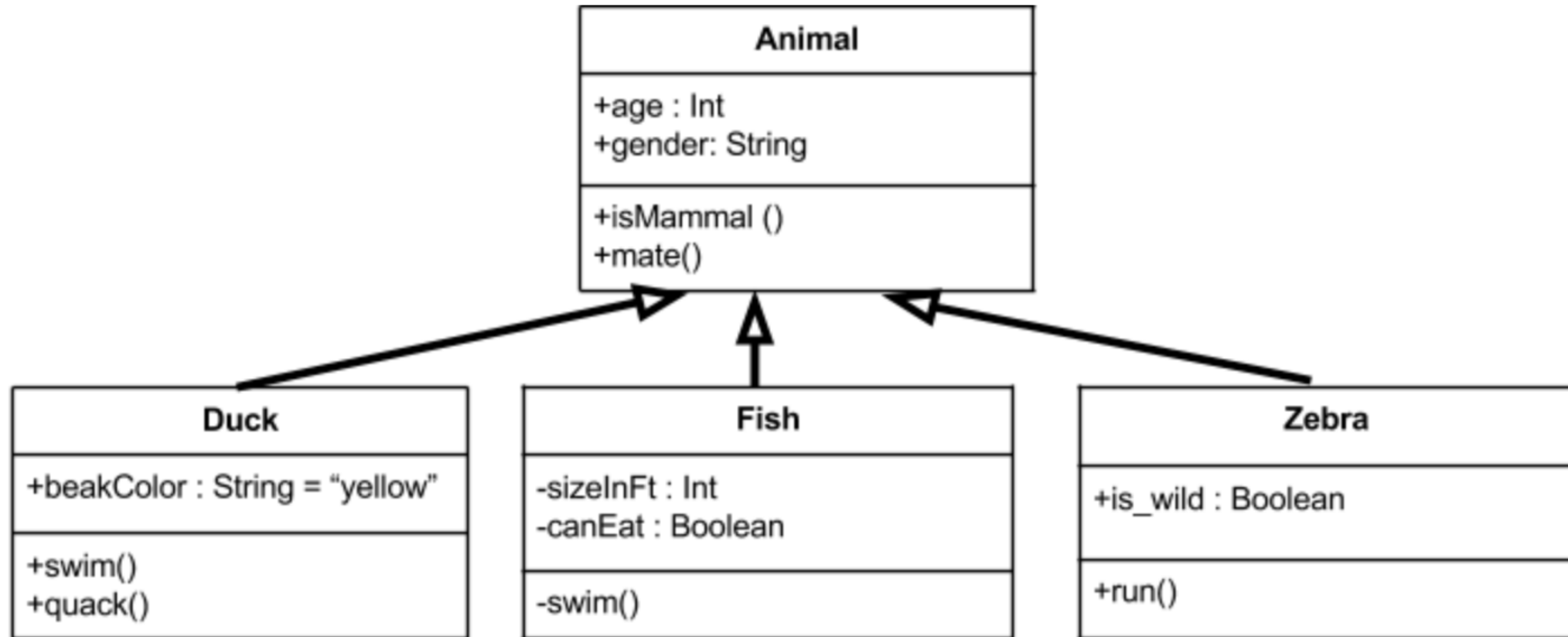


**Destroy Message:**

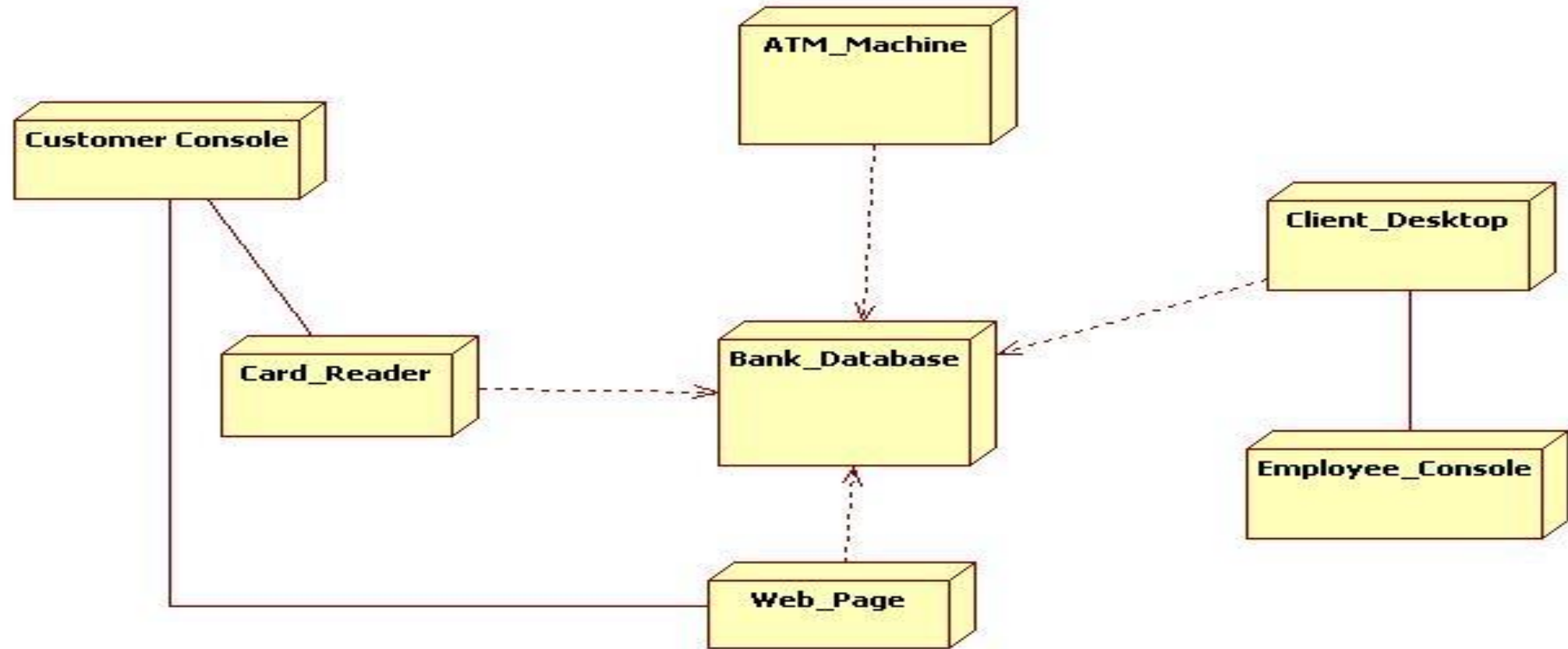


**Duration Message**

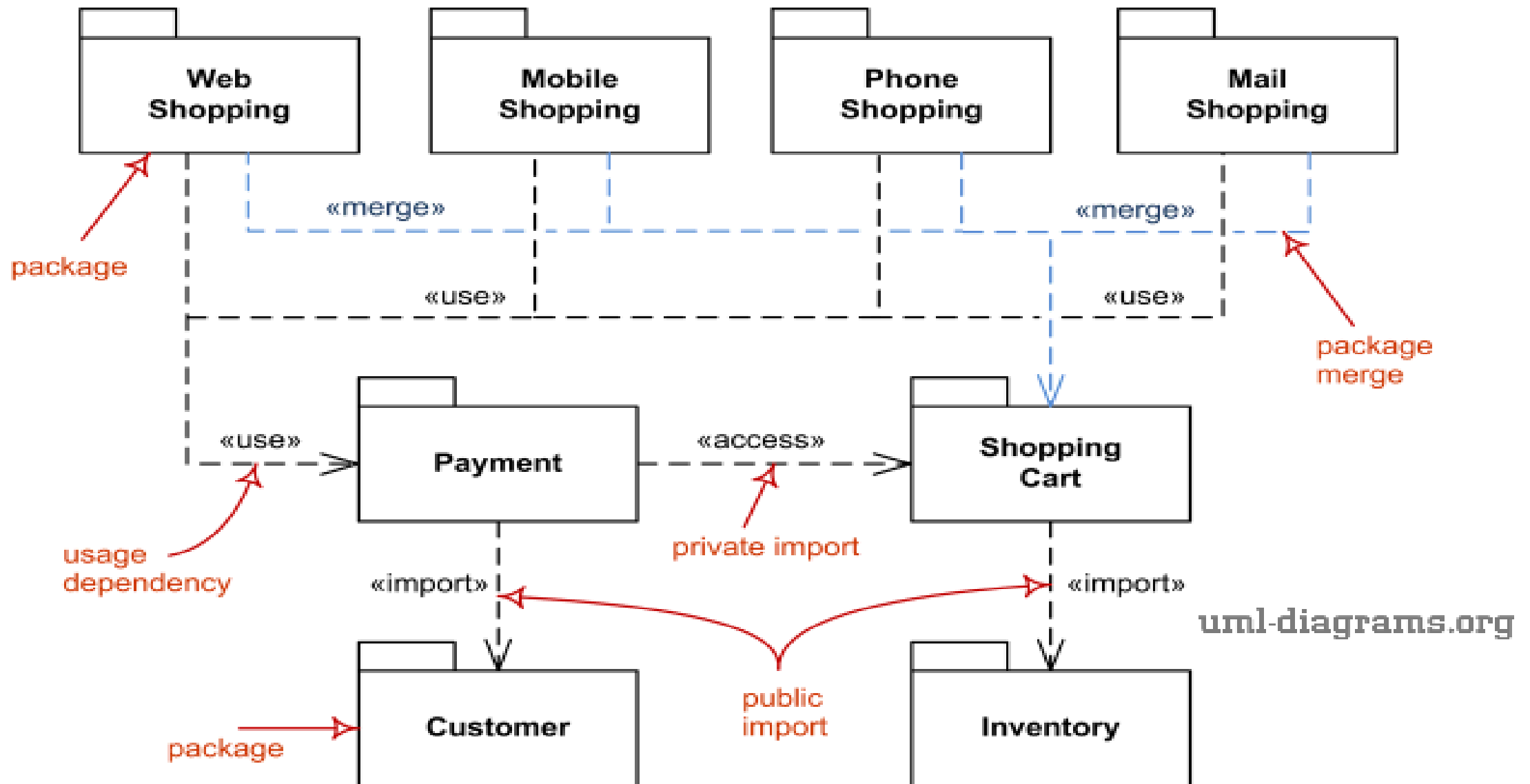
# Class Diagram



# Deployment Diagram

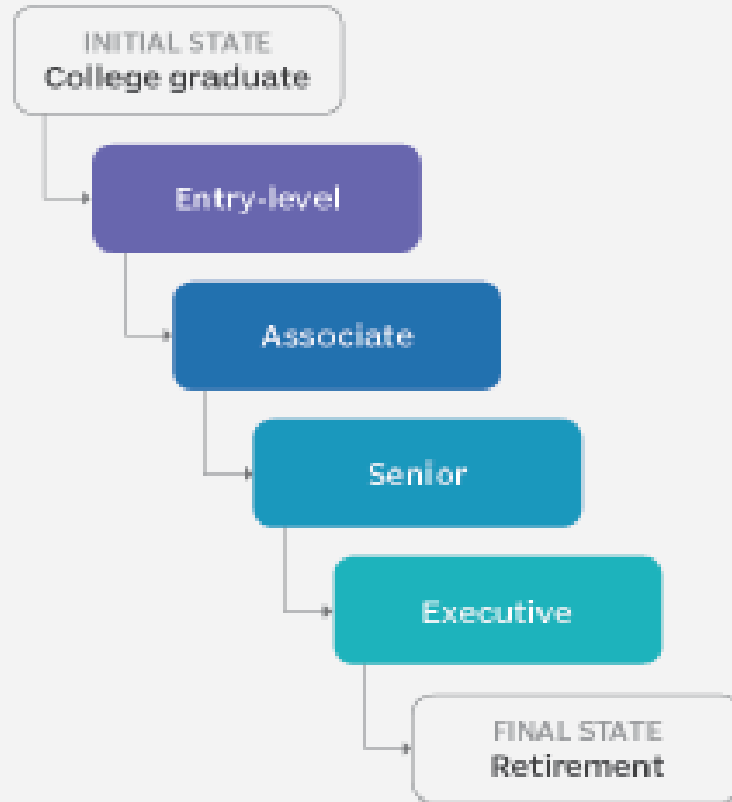


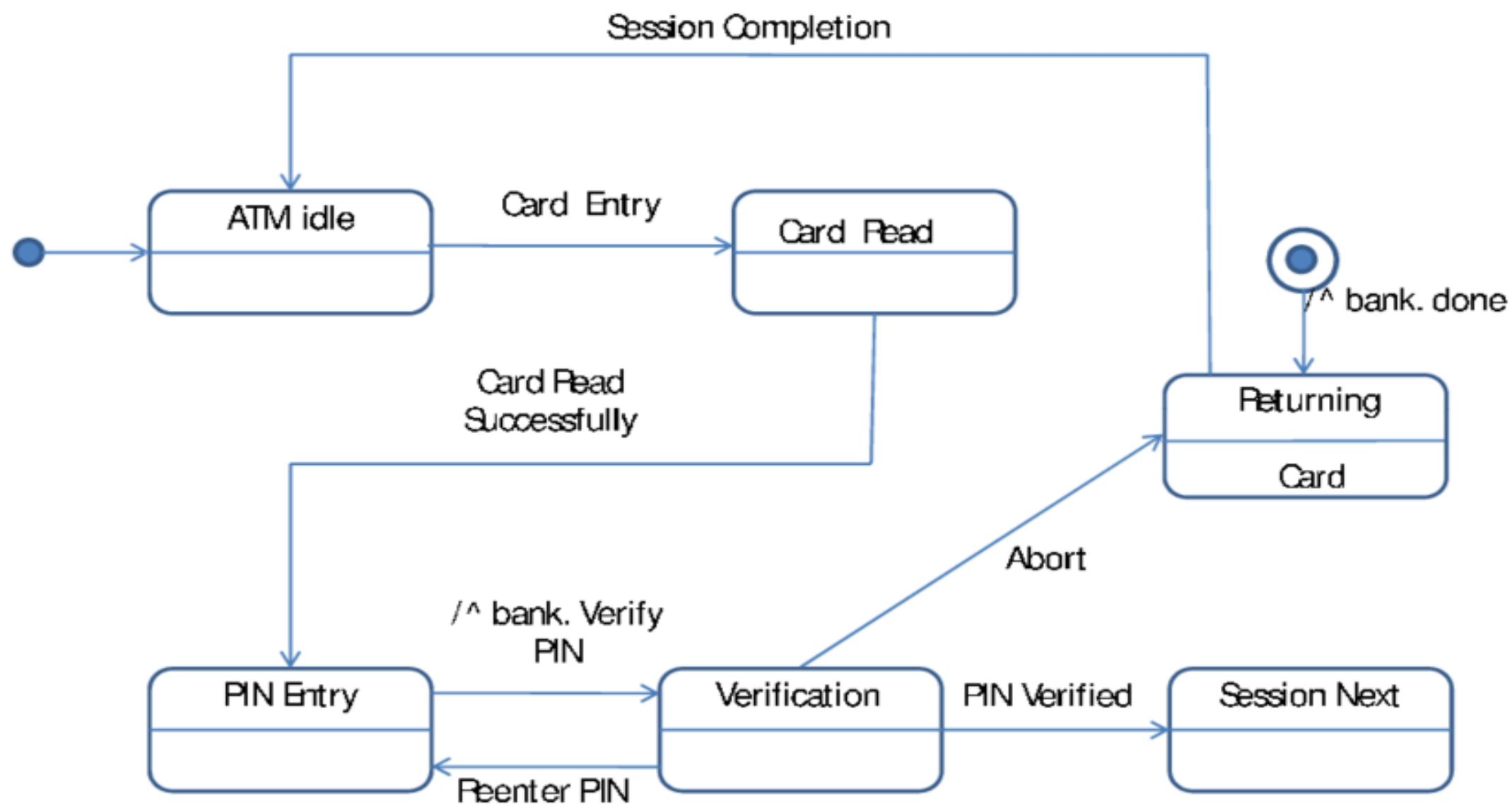
# Package Diagram



# State Chart Diagram

## Example of a state diagram





*Thanks.....*