Introduction

Object Orientation

- **Object Orientation (OO)** is a **paradigm** that views and models **a system** as **a collection of interacting objects**.
- **Abstraction** is a model to **include the most important aspects** while **ignoring less important details**.
- **Encapsulation** is a mechanism for **restricting access** to some **internal components** of an **object**.
- **Polymorphism** is the ability of **different objects to respond to the same request in different ways**.

Objects

- An **object** corresponds to **a single entity** in the **real world**.
- **Objects** may be **tangible or intangible**.
- All **objects** contain **information** and **exhibit behavior**.

Classes

- A **class** is a **uniquely identified abstraction** of a set of **related instances** that share **identical or similar characteristics**.
- An **attribute** is a **named property** of a **class**.
- An **operation** is the **implementation** of a **service**.
- A **class** is an **object-blueprint** and becomes an **object** when **instantiated**.

Object Oriented Software Development

- A **software process** is a **set of activities** that lead to the **production** of **software**.
- A **software process model or paradigm** is an **abstraction** of **software processes**.
- Common software paradigms include:
  1. **Waterfall** - A **linear process** with **distinct phases**.
  2. **Spiral** - **Iterative risk management**.
  3. **Agile** - An **iterative** and **incremental** methodology.

OOA, OOD, OOP, and OOT

- **Object-Oriented Analysis (OOA)** is the process of **analyzing a problem** and **discovering all entities** associated with the problem.
- **Object-Oriented Design (OOD)** is the process of **taking the entities discovered in OOA** and **determining how they interact**.
- **Object-Oriented Programming (OOP)** is the process of **implementing an object-oriented design** in a **programming language**.
- **Object-Oriented Testing (OOT)** is the process of **testing the implemented design**.

The Unified Process

- The **Unified Process** is an **iterative and incremental software development process framework**.
- The **main principals** of the **Unified Process**:
  1. It is **use-case driven**. A **use case** is a **written description** of **interactions** between **a role** and **a system** to **achieve a goal**. It links the **requirements** to the **implementation**.
  2. It is **architecture-centric**. It is a **theme** from the **earliest stages** of a **project**.
  3. It relies on **workflow in iterations**.
  4. It creates **incremental development**. Each **iteration** has **four properties**: the **duration**, the **tasks**, the **outcomes**, and the **usage**.
- The **main benefit** to **iterative development** is the ability to get **feedback, on a regular basis**.

Project Inception

- The **first stage** in a project is the **inception stage**.
- The **inception stage** is a short stage that addresses the following questions:
  1. What are the outcomes?
     a. What is the vision and business case for the project?
     b. Is it feasible to work on this project?
     c. Should the project be purchased or built?
     d. What is the rough cost range of developing the project?
     e. Should the project continue, or stop?
  2. What are the methods for achieving the outcome?
     a. What are the requirements and how will they be achieved?

3. What are the objectives?
    a. What are the most important objectives?
    b. What is the initial plan?