Lecture 03

Noun-Verb analysis and CRC Cards

Outline

1. SDLC – Software Development Life Cycle
    a. Requirements gathering
        i. What?
        ii. How?
            1. Use case diagrams
            2. Use case scenarios as user stories
    b. Object Oriented Analysis
        i. What?
        ii. How?
            1. Identifying Classes
            2. Defining Classes
            3. Describing Classes
        iii. Analysis procedure
            1. Noun Verb Analysis
                a. Identify objects in the problem statement by looking for **nouns** and phrases.
                b. Each of these could be an object in the solution.
                c. Eliminate unnecessary nouns by applying these five rules.
                    i. Redundant
                    ii. An event or operation
                    iii. Outside scope of system
                    iv. Meta Language
                    v. An attribute

            2. Analysis Class
                a. Entity Classes
                b. Boundary Classes
                c. Control Classes

            3. CRC Cards
                a. Fields
                    i. Name of the class
                    ii. Responsibilities
                    iii. Collaborations
    c. Design

        i. What?
        ii. How?
            1. Class diagrams
            2. Object Diagrams
            3. Sequence Diagrams
            4. State Diagrams
            5. Component Diagrams
            6. Deployment Diagrams
        iii. Key principles
            1. Encapsulation
            2. Inheritance
            3. Polymorphism
            4. Abstraction
    d. Implementation
    e. Testing
    f. Deployment
    g. Maintenance

SDLC

Introduction

Software Development Life Cycle is a structured process used for **developing software applications efficiently and systematically**.

There are 7 steps in SDLC when it comes to OOP context.

1. Requirement Gathering
2. Object Oriented Analysis
3. Design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

1. Requirements Gathering

What?

- This is the first step of the SDLC. In this step you are supposed to **understand what clients need and want the software to do**.

How?

1. **Use Case Diagrams**
    - Use Case Diagrams are **visual representations** that demonstrate how users **(actors)** interact with the system to achieve specific goals (**use cases**).
    - Read More…

2. **Use Case Scenarios as User Stories**
    - **Use Case Scenarios:** Detailed textual descriptions of a specific sequence of actions a user performs with the system to achieve a particular goal.
        - They outline:
            - The steps involved
            - Potential issues
            - Alternative paths

- **User Stories:** Short, simple description of a feature told from the perspective of the end-user.

Ex:

User Story: As a ***cashier*** I want ***the computer to automatically apply loyalty discounts*** at the checkout, so I ***do not have to calculate it manually***.

Use Case Scenario:

>>Steps Involved

1. Get the registered number from the user.
2. If there are any discounts calculate it for the current bill.
3. Apply the discount and complete the checkout.

>>Potential Issues

1. Users do not need it for the current bill.
2. Users forgot their registered phone numbers.

>>Alternative paths

1. Give loyalty members a loyalty card and also let them claim discounts via phone number.
2. If they do not prefer a discount on the current bill, add it as store credits to their account.

2. Object Oriented Analysis

What?

- In this step you should understand the problem in terms of ***objects***.
- You are required to identify the
    - Key entities
    - Concepts
    - Abstractions

  that are relevant to the problem.

How?

- This has three main steps
    1. Identifying Classes
       Determine the different categories or blueprints for the objects in the system.

       $$\left.\begin{array}{l}Lecturer\\ Chairperson\\ Security\end{array}\right\} \quad Employee$$

       $$\left.\begin{array}{l}OOP\\ DM\\ TR\end{array}\right\} \quad Modules$$

    2. Defining Classes
       Specify the attributes and methods that each class will have.

       Class: *Employee*
       Attributes: *Salary, Name, Position*
       Methods: *calculateBonus(), promoteTo()*

    3. Describing Classes
       Describes how objects interact with each other.

       Ex: The ***manager*** can promote the ***clerk***.

Analysis Procedure

1. **Noun/ Verb Analysis**

- Noun verb analysis is a technique to understand potential **classes** and their **responsibilities**.
- It basically means examining the textual description of the problem (System requirements).
- After extracting the nouns and verbs from the description, we use some rules to eliminate some possible classes.
- There are two types of nouns
  - **Common Nouns:** Mostly stand for a class.
    Ex:
    → User
    →Book
    →Employee

  - **Proper Nouns:** Stands for an object.
    Ex:
    → IT0024695
    →Object Oriented Programming with Java – 3rd Edition
    →Mr. Rajitha Dias

<u>Identifying Classes</u>

How?

1. Identify objects in the statement by looking for nouns/ phrases.
2. Underline/ Write down these nouns
3. Eliminate the unnecessary nouns according to the five rules.
   a. **Redundant**
   The two nouns that stand for the same object.
      Ex: *Customer* and *Buyer* stand for the same object. Keep one and eliminate the other.

   b. **An event or an operation**
   A method/ behavior.
      Ex: *Calculate Discount* and *Buy Item*. They are both methods (behaviors/ Actions). Eliminate both.

c. **Outside scope of system**
A noun that does not affect the system that we are making.
Ex: *Supermarket* itself has nothing to do with the inner system. So, eliminate it

d. **Meta-Language**
Terms used to describe a category rather than actual objects in the system.
Ex: *A person who buys groceries at the supermarket* can call as a *customer.*

e. **An attribute**
A quality/ data of an object.

Ex: The *price* of an item is an attribute. So, you can eliminate it.

Identifying Responsibilities

How?

1. Look into the verbs in the problem description.
2. These verbs describe actions, tasks or behaviors that occur within the system.
3. Then we should carefully examine how these verbs connected to the nouns (Classes).
4. These verbs are possible responsibilities of the respective class (connected noun).

Note

- The responsibilities of the class do not mean the *methods* of the class.
- Some responsibilities might be achieved using several methods.

Ex:

➔ Responsibility: Borrow a book from the library.
  - Methods used to achieve it:
    - searchInventory()
    - getUserId()
    - checkBorrowLimit()
    - updateUserBorrowedBook()
    - updateInventory()

Identifying Collaborators

How?

1. Examine the responsibilities that you found out.
2. Sometimes to achieve these responsibilities classes should collaborate with other classes.
3. When a class collaborates with other classes in order to fulfill the responsibilities, the other classes that it collaborates with are called **collaborators**.
   Ex:
   ➔ Responsibility: Borrow a book
      - Collaborators:
         - Inventory

2. Analysis Classes

- An analysis class is a representation of a
  - key concept
    or
  - entity

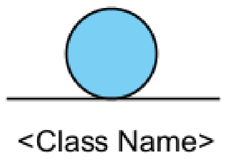
  within the system that we are trying to model.

- It's a way to capture and describe the essential
  - Characteristics
  - Behaviors
    of these important elements of the problem.

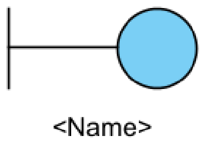
- Analysis classes are **not** about how they get implemented.
- It's about **what entities do.**

Types of analysis classes

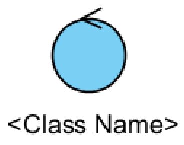
1. **Entity Classes**
   - Represents the core data or information.
   - They usually have few attributes and methods.
   - Usually, the purpose is storing data.
   - Represented using the following symbol.



2. **Boundary Classes**
   - Handles the interaction between the system and the outside world.
   - Might interact with,
     - The system itself
     - Users
     - Other Systems
     - Devices
   - Acts as an interface.
   - Represented using the following symbol.



3. **Control Classes**
   - These classes manage the flow of activity within the system.
   - Used to control interactions between classes.
   - Often implement the logic of
     - Use cases
     - Business rules
   - Represented using the following symbol.



3. CRC Cards

- CRC (Class, Responsibility, Collaboration) is a simple way to represent the:
  - Responsibilities
  - Collaborators

  of a class.

- It has three fields.
  - **Class**: For class name
    - On top of the card
  - **Responsibilities**: To list down the expected functionalities of the class.
    - On the left of the card
  - **Collaborators**: Other classes that the class needs to interact with.
    - On the right of the card

- Format of the CRC card:
  - 4 x 6 inches card
  - It has three sections
    - Name of the class
    - Responsibilities of the class
    - Collaborations of the class
  - The format of a CRC card looks like this.

<Class Name>	
<Responsibility>	<Collaborators>
<Responsibility>	<Collaborators>
<Responsibility>	<Collaborators>
<Responsibility>	<Collaborators>
<Responsibility>	<Collaborators>

- You just have to fill out the responsibilities and respective collaborators for each responsibility in CRC table.

  Example Scenario

Member	
Update Password	
Borrow Book	Inventory
Return Book	Inventory, PenaltyDatabase
Check Available Books	Inventory

Check Pending Penalty	PenaltyDatabase

3. Design

- The designing phase consists of:
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - State diagrams
  - Component diagrams
  - Deployment diagrams

- We use the following key principles in the design phase.
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction

- We should try to create a,
  - Modular
  - Reusable
  - Maintainable system.

4. Implementation

- Once we designed the system, we need to translate it to an Object-Oriented programming language.
- We will
  - Implement the classes
  - Create the objects
  - Define the methods
    in this step

5. Testing

- Focuses on objects and their interactions.
- We use:
  - **Unit Tests**: To test individual classes and methods
  - **Integration Tests**: To test those objects, work together correctly.
  - **Polymorphism Testing**
  - **Inheritance Testing**
  - **TDD – Test-Driven Development:** Tests are written (Planned) before the code.

6. **Deployment**
   - The process of establishing the program in client's place/ platform.
   - May be involve:
     - Configuring the runtime environment
     - Setting up the necessary infrastructure
   - Deployment Diagrams that we made in design phase might be helpful to this process.

7. **Maintenance**
   - The process of maintaining and updating the system.
   - The **modularity** and **reusability** in OOP make maintenance easier.

Car Manufacturer Analogy for SDLC

Assume a company working on manufacturing a modern car.

In SDLC, there are steps as we remember.

1. **Requirements Gathering**
    - This is where we get to understand what we should make.
    - Let's say the car should be power efficient.
        - Steps Involved:
            - Use a battery instead of fossil fuel
        - Potential Issues:
            - Lag of the performance
        - Alternative Paths:
            - Use the hybrid architecture
    - The car should be modern looking.
        - Steps Involved.
            - Apply a futuristic looking body.
            - Use shiny and glossy metals to create the body.
        - Potential issues:
            - The shape could reduce power efficiency by increasing air resistance.
            - Metals can increase the dead weight and reduce the speed.
        - Alternative paths:
            - Use aero-dynamic body while keeping a futuristic look.
            - Use alternative materials such as carbon fiber for the body.
2. **Object-Oriented Analysis**
    - In this step, we identify each required element (Objects) in the system.
    - **Car Analogy**:
        - Identify key components of the car as objects:
            - Engine (or Battery)
            - Chassis

            - Wheels
            - Steering System
            - Interior (Seats, Dashboard)
            - Electronics (Navigation, Entertainment)

3. **Design**
    - This is where we create the blueprint for the car.
    - **Car Analogy**:
        - Create detailed designs for each object identified in the analysis phase.
            - **Class Diagrams**: Design the structure of each component (e.g., the "Engine" class has attributes like power output, fuel consumption, and methods like "start," "accelerate," "decelerate")
            - **Object Diagrams**: Show how specific instances of the components will look (e.g., a diagram of a particular model of the electric motor)
            - **Sequence Diagrams**: Map out how the different systems interact (e.g., how the accelerator pedal interacts with the engine and transmission to increase speed)
            - **State Diagrams**: Define the different states of a component and how it transitions between them (e.g., the engine's states: "off," "idle," "running")
4. **Implementation**
    - This is where we build the car based on design. We do this inside the factory. Not at the client's place.
    - **Car Analogy**:
        - Manufacturing each component according to the design specifications.
            - Fabricate the chassis from the chosen materials (steel, carbon fiber)
            - Assemble the engine, transmission, and other mechanical parts
            - Develop the software for the car's computer systems (engine control, navigation)
            - Wire the electrical systems
            - Assemble the interior

5. **Testing**
    - This is where we rigorously test the car to ensure it meets quality and performance standards.
    - **Car Analogy**:
        - Test individual components (engine, brakes, electronics)
        - Test the assembled car as a whole
            - Performance tests (speed, acceleration, fuel efficiency)
            - Safety tests (crash tests, brake tests)
            - Reliability tests (endurance testing in various conditions)
            - Software testing (for the car's computer systems)
6. **Deployment**
    - This is where we release the car into the market.
    - **Car Analogy**:
        - Set up manufacturing plants and distribution networks
        - Train dealership staff
        - Market and sell the car to customers
7. **Maintenance**
    - This is where we provide ongoing support and make improvements to the car.
    - **Car Analogy**:
        - Provide service and repairs for cars
        - Handle warranty issues
        - Issue recalls for defective parts
        - Gather feedback from customers to improve future car models
        - Develop and release software updates for the car's systems

Real World Library System Analogy for Object Oriented Analysis

A library provides various services to its members, allowing them to borrow, return, and reserve books. It maintains a collection of books, magazines, and digital resources such as e-books and audiobooks. Each book has details such as title, author, ISBN, genre, and availability status.

Library members can register for an account by providing personal details, such as name, address, and contact information. Once registered, they can borrow books for a limited period and must return them by the due date. Late returns may result in fines, which members must pay before borrowing more books. Members can also request book reservations if a book is currently unavailable.

The library staff manages book inventory, adds new books, updates book information, and removes outdated or damaged books from the system. They also handle membership registrations, process book returns, and collect fines. Library employees can generate reports on borrowed books, overdue items, and member activities.

Some libraries offer digital services where members can access e-books and online journals. These digital resources may have limited access periods, similar to physical books. Additionally, libraries organize events such as book fairs, reading sessions, and author meetups for their members.

The system should support searching for books by title, author, or genre, tracking borrowed books, processing returns, handling overdue fines, and managing member records. It should also generate reports on book availability.

1. Noun Verb Analysis

<u>Identify Potential Classes</u>

A **library** provides various **services** to its **members**, allowing them to borrow, return, and reserve books. It maintains a collection of **books**, **magazines**, and **digital resources** such as **e-books** and **audiobooks**. Each book has details such as **title**, **author**, **ISBN**, **genre**, and **availability status**.

Library **members** can register for an **account** by providing **personal details**, such as **name**, **address**, and **contact information**. Once registered, they can borrow **books** for a limited period and must return them by the **due date**. Late returns may result in **fines**, which **members** must pay before borrowing more **books**. Members can also request **book reservations** if a book is currently unavailable.

The **library staff** manages book **inventory**, adds new books, updates book information, and removes outdated or damaged books from the **system**. They also handle membership registrations, process book returns, and collect **fines**. Library **employees** can generate reports on borrowed books, **overdue items**, and member activities.

Some libraries offer **digital services** where members can access **e-books** and **online journals**. These digital resources may have limited access periods, similar to physical books. Additionally, libraries organize events such as **book fairs**, **reading sessions**, and **author meetups** for their members.

The system should support searching for books by **title**, **author**, or **genre**, tracking borrowed books, processing returns, handling overdue fines, and managing member records. It should also generate **reports** on book availability.

Identified ***Potential*** Classes

1. Library
2. Services
3. Members
4. Books
5. Magazines
6. Digital Resources
7. E-Books
8. Audiobooks
9. Title
10. Author
11. ISBN
12. Genre
13. Availability Status
14. Account
15. Personal Details
16. Name
17. Address
18. Contact Information
19. Due Date
20. Fines
21. Book Reservations
22. Library Staff
23. Inventory
24. System
25. Employees
26. Digital Services
27. Book Fairs
28. Reading Sessions
29. Author Meetups
30. Reports

Eliminate Potential Classes

1. ~~Library~~: Outside The Scope
2. ~~Services~~: An event or Operation
3. **Members**
4. ~~Books~~: Meta Language
5. ~~Magazines~~: Meta Language
6. ~~Digital Resources~~: Meta Language
7. ~~E-Books~~: Meta Language
8. ~~Audiobooks~~: Meta Language
9. ~~Title~~: An attribute
10. ~~Author~~: An attribute
11. ~~ISBN~~: An attribute
12. ~~Genre~~: An attribute
13. ~~Availability Status~~: An attribute
14. ~~Account~~: Redundant (Member means the same class)
15. ~~Personal Details~~: Meta Language (Personal Details is a category of member class)
16. ~~Name~~: An attribute
17. ~~Address~~: Attribute
18. ~~Contact Information~~: An attribute
19. ~~Due Date~~: An attribute
20. ~~Fines~~: Attribute
21. ~~Book Reservations~~: Meta Language
22. ~~Library Staff~~: Redundant
23. **Inventory**
24. ~~System~~: Outside the Scope
25. **Employees**
26. ~~Digital Services~~: Event or Operation
27. ~~Book Fairs~~: Event or Operation
28. ~~Reading Sessions~~: Event or Operation
29. ~~Author Meetups~~: Event or Operation
30. **Reports**

We can eliminate Books, Magazines, E-Books, Audio Books as Meta Language. We will create a single class called **Library Item** to store them. We can use inheritance here.

Also, Book reservation as meta language. We will create a class called **Reservation** to store reservation information.

Since Title, Author, ISBN, Genre and availability status are attributes of the **Library Item** class, we eliminate them.

Account and Member means the same thing. So, we will keep **Member** and eliminate the Account. Personal Details are a **part of the** Member class. So, we eliminate it using "Meta Language" rule.

Name, Address, Contact Information are attributes of Member class. Therefore, we eliminate them as well.

Due date, Fine are attributes of Reservation class. Hence, we eliminate them as well.

Library Staff and Employees are the same thing. Let's keep **Employee** and eliminate the **Library Staff**.

System and Library both are outside the scope. Therefore, we eliminate both.

Here are the final entity classes:

1. Member
2. Employee
3. LibraryItem
4. Inventory
5. Reservation
6. Reports

Identify Responsibilities

A library provides various services to its members, allowing them to borrow, return, and reserve books. It **maintains a collection** of books, magazines, and digital resources such as e-books and audiobooks. Each book has details such as title, author, ISBN, genre, and availability status.

Library members can **register** for an account by **providing personal details**, such as name, address, and contact information. Once registered, they can **borrow books** for a limited period and must **return** them by the due date. Late returns may result in **fines**, which members must **pay** before borrowing more books. Members can also **request book** reservations if a book is currently unavailable.

The library staff **manages book inventory**, **adds new books**, **updates book information**, and **removes outdated or damaged books** from the system. They also **handle membership registrations**, **process book returns**, and **collect fines**. Library employees can **generate reports** on borrowed books, overdue items, and member activities.

Some libraries offer digital services where members can **access e-books** and online journals. These digital resources may have limited access periods, similar to physical books. Additionally, libraries **organize events** such as book fairs, reading sessions, and author meetups for their members.

The system should support **searching for books** by title, author, or genre, tracking borrowed books, processing returns, **handling overdue fines**, and **managing member records**. It should also **generate reports** on book availability.

Identified Responsibilities

1. Inventory: Maintain the collection
2. Login: Register User
    a. Provide personal Details
3. Reservations: Borrow Books
4. HandleRegistration: Handle Membership Registrations
5. Reservation: Process book returns
6. FineManagement: Collect Fines
7. Inventory: Access resources
8. EventManager: Organizing Events
9. Inventory: Searching for Books
10. FineManagement: Handling Overdue Fines
11. UserCredentials: Managing Member Records
12. GenerateReport: Generate Reports

As you noticed we have created some new classes.

Which are

1. Login	- Boundary Class
2. HandleRegistration	- Control Class
3. FineManagement	- Control Class
4. EventManager	- Control Class
5. UserCredentials	- Entity Class

Identifying Collaborators

1. Maintain the collection
   a. Inventory ⇔ LibraryItem
2. Register User
   a. Login ⇔ UserCredentials
3. Borrow Books
   a. Reservation ⇔ Inventory
   b. Inventory ⇔ LibraryItem
4. Handle Membership Registration
   a. HandleRegistration ⇔ UserCredentials
5. Process Book Returns
   a. FineManagement ⇔ Reservation
6. Access Resources
   a. Inventory ⇔ LibraryItem
7. Collect Fines
   a. FineManagement ⇔ Reservation
8. Organizing Events
   a. EventManager
9. Searching for Books
   a. Inventory ⇔ LibraryItem
10. Handling Overdue Fines
    a. FineManagement ⇔ Reservation
11. Managing Member Records
    a. UserCredentials
12. Generate Reports
    a. GenerateReport ⇔ Employee
    b. GenerateReport ⇔ Inventory

- Now let’s create a few CRC cards for the above scenario.

Member	
Maintain personal details	
View borrowed/reserved books	
Request book reservations	Reservation
Pay overdue fines	FineManagement
Authenticate login	UserCredentials

Inventory	
Maintain collection of library items	LibraryItem
Add/update/remove items	
Search items by title, author, or genre	LibraryItem
Track availability status	LibraryItem
Assist in borrow/return process	Reservation

Reservation	
Handle borrowing of items	Inventory
Track due dates and overdue items	
Process book returns	Inventory, FineManagement
Manage book reservations	Member

FineManagement	
Calculate overdue fines	Reservation
Collect payments	Member
Restrict borrowing if fines unpaid	Reservation

UserCredentials	
Manage login credentials for members	Login
Authenticate users during login	Login
Maintain member registration info	HandleRegistration

- GPT Generated code for above scenario

```java
import java.util.Scanner;

// Class representing a library member
class Member {
    private String name;
    private String contactInfo;

    public Member(String name, String contactInfo) {
        this.name = name;
        this.contactInfo = contactInfo;
    }

    public String getName() {
        return name;
    }
}

// Class representing a library item (books, magazines, etc.)
class LibraryItem {
    private String title;
    private String author;
    private String isbn;
    private String genre;
    private boolean isAvailable;

    public LibraryItem(String title, String author, String isbn, String
genre) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
        this.genre = genre;
        this.isAvailable = true; // By default, the item is available
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void borrowItem() {
        isAvailable = false;
    }

    public void returnItem() {
        isAvailable = true;
    }

    public String getTitle() {
        return title;
    }
}

// Class representing the inventory of library items
class Inventory {
    private LibraryItem[] items;
    private int count;

    // Fixed size array for inventory
    public Inventory(int capacity) {
        items = new LibraryItem[capacity];
        count = 0;
    }

    public void addItem(LibraryItem item) {
        if (count < items.length) {
            items[count] = item;
            count++;
        } else {
            System.out.println("Inventory full! Cannot add more items.");
        }
    }

    public LibraryItem searchItem(String title) {
        for (int i = 0; i < count; i++) {
            if (items[i].getTitle().equalsIgnoreCase(title) &&
items[i].isAvailable()) {
                return items[i];
            }
        }
        return null; // Item not found or not available
    }

    public void reportInventory() {
        System.out.println("Library Inventory Report:");
        for (int i = 0; i < count; i++) {
            System.out.println("Title: " + items[i].getTitle() + ",
Available: " + items[i].isAvailable());
        }
    }
}

// Class representing a reservation
class Reservation {
    private Member member;
    private LibraryItem item;

    public Reservation(Member member, LibraryItem item) {
        this.member = member;
        this.item = item;
    }

    public void processReservation() {
        if (item.isAvailable()) {
            item.borrowItem();
            System.out.println("Reservation successful for " +
member.getName() + " for the item: " + item.getTitle());
        } else {
            System.out.println("Item is not available for reservation.");
        }
    }
}
```

```java
// Main class to demonstrate the library system
public class LibrarySystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Creating a fixed-size inventory with a capacity of 10 items
        Inventory inventory = new Inventory(10);

        // Sample member registration
        System.out.print("Enter member name: ");
        String memberName = scanner.nextLine();
        System.out.print("Enter contact info: ");
        String contactInfo = scanner.nextLine();
        Member member = new Member(memberName, contactInfo);

        // Adding items to the inventory
        inventory.addItem(new LibraryItem("The Great Gatsby", "F. Scott
Fitzgerald", "123456789", "Fiction"));
        inventory.addItem(new LibraryItem("1984", "George Orwell",
"987654321", "Dystopian"));

        // Searching for an item
        System.out.print("Enter the title of the item to borrow: ");
        String titleToBorrow = scanner.nextLine();
        LibraryItem itemToBorrow = inventory.searchItem(titleToBorrow);
        if (itemToBorrow != null) {
            Reservation reservation = new Reservation(member, itemToBorrow);
            reservation.processReservation();
        } else {
            System.out.println("Item not found or not available.");
        }

        // Generate and display inventory report
        inventory.reportInventory();

        scanner.close();
    }
}
```