

# Programming Paradigms & Practices

Mohamed Sweelam

Software Engineer



# Outline

- 1 Course Objectives
- 2 Understanding Programming Paradigms
- 3 Advanced Programming Techniques
- 4 Project Structure and Code Quality
- 5 Deployment Models
- 6 Conclusion



# Course Objectives

- 1 Provide good Arabic content for the topic



# Course Objectives

- 1 Provide good Arabic content for the topic
- 2 Provide comprehensive insights into programming paradigms



# Course Objectives

- 1 Provide good Arabic content for the topic
- 2 Provide comprehensive insights into programming paradigms
- 3 Explore advanced programming techniques



# Course Objectives

- 1 Provide good Arabic content for the topic
- 2 Provide comprehensive insights into programming paradigms
- 3 Explore advanced programming techniques
- 4 Discuss best practices for project structure and code quality



# Course Objectives

- 1 Provide good Arabic content for the topic
- 2 Provide comprehensive insights into programming paradigms
- 3 Explore advanced programming techniques
- 4 Discuss best practices for project structure and code quality
- 5 Overview of deployment models and strategies



# Understanding Programming Paradigms

## Definition *wikipedia*

Programming paradigms are fundamental styles or approaches to computer programming, offering distinct methodologies for designing and structuring software.

## Importance *ChatGPT*

Understanding different programming paradigms is crucial for selecting the right approach to solve specific problems, leading to more efficient and maintainable code.

## Historical Context *wikipedia*

Programming paradigms have evolved over time, with significant contributions from various programming languages that introduced unique features and concepts, shaping the way we write software today.





# Programming Paradigms Types

- Imperative Programming
- Procedural Programming
- Object-Oriented Programming
- Declarative Programming
- Functional Programming
- Event-Driven Programming
- Aspect-Oriented Programming
- Reactive Programming

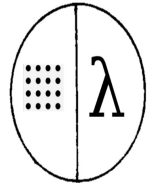
Imperative



Functional



Object-Oriented



# Imperative Programming

- In this paradigm, the program is a sequence of instructions that explicitly change the program state. It focuses on how to achieve a task.



# Imperative Programming

- In this paradigm, the program is a sequence of instructions that explicitly change the program state. It focuses on how to achieve a task.
- C, Python, Java (most common imperative languages have procedural features as well)



# Imperative Programming

- In this paradigm, the program is a sequence of instructions that explicitly change the program state. It focuses on how to achieve a task.
- C, Python, Java (most common imperative languages have procedural features as well)
- **Key Concepts:** Variables, controls (loops, conditionals), and subroutines.



# Imperative Programming

- In this paradigm, the program is a sequence of instructions that explicitly change the program state. It focuses on how to achieve a task.
- C, Python, Java (most common imperative languages have procedural features as well)
- **Key Concepts:** Variables, controls (loops, conditionals), and subroutines.

```
#include <stdio.h>

int main() {
    int Salaries[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int N = sizeof(Salaries) / sizeof(Salaries[0]);
    int result = 0, i;

    for (i = 0; i < N; i++) {
        if (Salaries[i] % 2 == 0) {
            result += Salaries[i];
        }
    }

    printf("Sum of even salaries is %d", result);
    return 0;
}
```



# Procedural Programming

- A subset of imperative programming that structures programs as a series of procedures or functions. These procedures perform operations on data and are reusable.



# Procedural Programming

- A subset of imperative programming that structures programs as a series of procedures or functions. These procedures perform operations on data and are reusable.
- C, Pascal, Python.



# Procedural Programming

- A subset of imperative programming that structures programs as a series of procedures or functions. These procedures perform operations on data and are reusable.
- C, Pascal, Python.
- **Key Concepts:** Procedures, functions, modular programming.





# Procedural Programming

- A subset of imperative programming that structures programs as a series of procedures or functions. These procedures perform operations on data and are reusable.
- C, Pascal, Python.
- **Key Concepts:** Procedures, functions, modular programming.

```
#include <stdio.h>

int SumEven(int arr[], int N) {
    int sum = 0;
    int i;

    for (i = 0; i < N; i++) {
        if (arr[i] % 2 == 0) {
            sum += arr[i];
        }
    }

    return sum;
}

int main() {
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};

    int result = SumEven(arr, 10);

    printf("sum of even numbers is %d \n" , result);

    return 0;
}
```



# Object-Oriented Programming

- Focuses on designing software using objects that represent real-world entities. Objects are instances of classes that encapsulate data and behavior.



# Object-Oriented Programming

- Focuses on designing software using objects that represent real-world entities. Objects are instances of classes that encapsulate data and behavior.
- Java, C#, PHP, Ruby.



# Object-Oriented Programming

- Focuses on designing software using objects that represent real-world entities. Objects are instances of classes that encapsulate data and behavior.
- Java, C#, PHP, Ruby.
- **Key Concepts:** Classes, objects, polymorphism, encapsulation, abstraction.



# Object-Oriented Programming

- Focuses on designing software using objects that represent real-world entities. Objects are instances of classes that encapsulate data and behavior.
- Java, C#, PHP, Ruby.
- **Key Concepts:** Classes, objects, polymorphism, encapsulation, abstraction.
- *let's try an example*



# Declarative Programming

- In this paradigm, the programmer specifies what the program should accomplish, rather than how to accomplish it. This is often used in conjunction with other paradigms like logical or functional programming.



# Declarative Programming

- In this paradigm, the programmer specifies what the program should accomplish, rather than how to accomplish it. This is often used in conjunction with other paradigms like logical or functional programming.
- SQL, HTML, CSS.



# Declarative Programming

- In this paradigm, the programmer specifies what the program should accomplish, rather than how to accomplish it. This is often used in conjunction with other paradigms like logical or functional programming.
- SQL, HTML, CSS.
- **Key Concepts:** Expressions, constraints, high-level abstraction.





# Declarative Programming

- In this paradigm, the programmer specifies what the program should accomplish, rather than how to accomplish it. This is often used in conjunction with other paradigms like logical or functional programming.
- SQL, HTML, CSS.
- **Key Concepts:** Expressions, constraints, high-level abstraction.

```
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;
```



# Advanced Programming Techniques

- Multithreading and Concurrency



# Advanced Programming Techniques

- Multithreading and Concurrency
- Reactive Programming and Asynchronous Streams



# Advanced Programming Techniques

- Multithreading and Concurrency
- Reactive Programming and Asynchronous Streams
- Memory Management and Optimization



# Advanced Programming Techniques

- Multithreading and Concurrency
- Reactive Programming and Asynchronous Streams
- Memory Management and Optimization
- Effective Error Handling and Debugging



# Project Structure and Code Quality

- Organizing Your Codebase
- Implementing Best Practices for Readability and Maintainability
- Writing Clean and Testable Code
- Integrating Continuous Integration and Automated Testing



# Deployment Models

- Understanding Different Deployment Strategies
- Containerization and Orchestration with Docker and Kubernetes
- Continuous Deployment and Delivery Pipelines
- Monitoring and Maintaining Production Environments



# Conclusion

- Recap of Key Learnings
- Emerging Trends in Software Development

