# 20CYS312 - Principles of Programming Languages
# Exploring Programming Paradigms

**Assignment - 01**

**Presented By**
**Manbendra Satpathy**
**CB.EN.U4CYS21039**
**TIFAC-CORE in Cyber Security**
**Amrita Vishwa Vidyapeetham, Coimbatore Campus**

Feb 2024


AMRITA
VISHWA VIDYAPEETHAM

# Outline



# Functional Programming Paradigm

- **Introduction to Functional Paradigm :**
  - **Definition :** Programming paradigm treating computation as the evaluation of mathematical functions.
  - **Key Principles :** Immutability, Pure Functions, First-Class Functions, Higher-Order Functions, Referential Transparency.
  - **Benefits :** Predictable code, ease of testing, and reasoning.
- **Key Concepts :**
  - **Immutability :** Data cannot be changed once it's created.
  - **Pure Functions :** Functions that always produce the same output for the same input, without side effects.
  - **First-Class Functions :** Functions are treated as first-class citizens, can be assigned to variables, passed as arguments, and returned from other functions.
  - **Higher-Order Functions :** Functions that can take other functions as arguments or return them as results.
  - **Referential Transparency :** An expression can be replaced with its value without changing the program's behavior.
- **Common Use Cases :**
  - Data processing and transformation.
  - Parallel and concurrent programming.
  - Mathematical and scientific computations.
  - State management in UI development.
- **Examples :**
  - Python , JavaScript , Haskell , Scala , Rust

# Functional Programming Paradigm - Kotlin

- **Definition :**
  - Kotlin is a statically typed programming language developed by JetBrains that runs on the Java Virtual Machine (JVM) and can also be compiled to JavaScript.
- **Key Concepts :**
  - Conciseness and expressiveness.
  - Interoperability with Java.
  - Null safety.
  - Coroutines for asynchronous programming.
- **Features :**
  - Smart casts.
  - Extension functions.
  - Type inference.
  - Data classes.
  - Lambdas and higher-order functions.
- **Advantages :**
  - Improved code readability.
  - Interoperability with existing Java code.
  - Null safety reduces NullPointerExceptions.
  - Concise syntax reduces boilerplate code.
- **Disadvantages :**
  - Smaller ecosystem compared to Java.
  - Learning curve for developers new to Kotlin.
  - Build times can be longer than Java in some cases.

# Functional Programming Paradigm - Kotlin ( Contd. )

- **Popularity :**
  - Kotlin has gained popularity for Android app development.
  - Widely used in various domains including web development and server-side applications.
- **Community :**
  - Active and growing community.
  - Strong support from JetBrains and Google.
- **Future :**
  - Continued integration with Android development.
  - Further enhancements to language features.
  - Increasing adoption in different software development domains.
- **Use Cases :**
  - Android app development.
  - Web development (server-side).
  - General-purpose programming.

# Object - Oriented Programming Paradigm

- **Introduction to Object - Oriented Programming Paradigm :**
  - **Definition :** Programming paradigm that organizes code into objects, which encapsulate data and behavior, promoting modularity and reusability.
  - **Key Principles :** Encapsulation, Inheritance, Polymorphism, Abstraction.
  - **Benefits :** Modularity, Reusability, Code Organization, and Maintenance.
- **Key Concepts :**
  - **Encapsulation :** Bundling of data and methods that operate on the data into a single unit (object).
  - **Inheritance :** Mechanism for creating a new class by inheriting properties and behaviors from an existing class.
  - **Polymorphism :** Ability of a function or method to operate on different types of data or objects.
  - **Abstraction :** Simplifying complex systems by modeling classes based on their essential characteristics.
- **Common Use Cases :**
  - Software design for modeling real-world entities.
  - Graphical user interface (GUI) development.
  - Simulation and modeling applications.
  - Large-scale systems requiring modularity and maintainability.
- **Examples :**
  - Python , Java , C++ , Ruby , PHP

# Object - Oriented Programming Paradigm - Dart

- **Definition :**
  - Dart is a general-purpose programming language developed by Google. It is designed for building web, mobile, and server applications, with a focus on ease of use, performance, and strong support for modern development workflows.
- **Key Concepts :**
  - Object-oriented programming.
  - Strong typing with type inference.
  - Asynchronous programming with Future and Stream.
  - Hot-reload for faster development.
- **Features :**
  - Just-in-time (JIT) and ahead-of-time (AOT) compilation.
  - Garbage collection for automatic memory management.
  - Standard library with rich functionality.
  - Flutter framework for building UIs.
- **Advantages :**
  - Productive development with hot-reload.
  - Strong type system enhances code quality.
  - Versatility for web, mobile, and server-side development.
  - Flutter for building cross-platform mobile apps.
- **Disadvantages :**
  - Smaller ecosystem compared to some other languages.
  - Learning curve for developers new to Dart.
- **Popularity :**
  - Growing popularity, especially in the Flutter community.
  - Used by Google for various projects, including Flutter.

# Object - Oriented Programming Paradigm - Dart ( Contd. )

- **Community :**
  - Active and growing community.
  - Strong support from Google, especially for Flutter development.
- **Future :**
  - Continued integration with Flutter for cross-platform development.
  - Enhancements to language features and tools.
  - Increased adoption in web and server-side development.
- **Use Cases :**
  - Web development (frontend and backend).
  - Mobile app development with Flutter.
  - Server-side programming.

# Code Snippet

- **Kotlin Code Snippet :**

Main.kt | scratch.kts

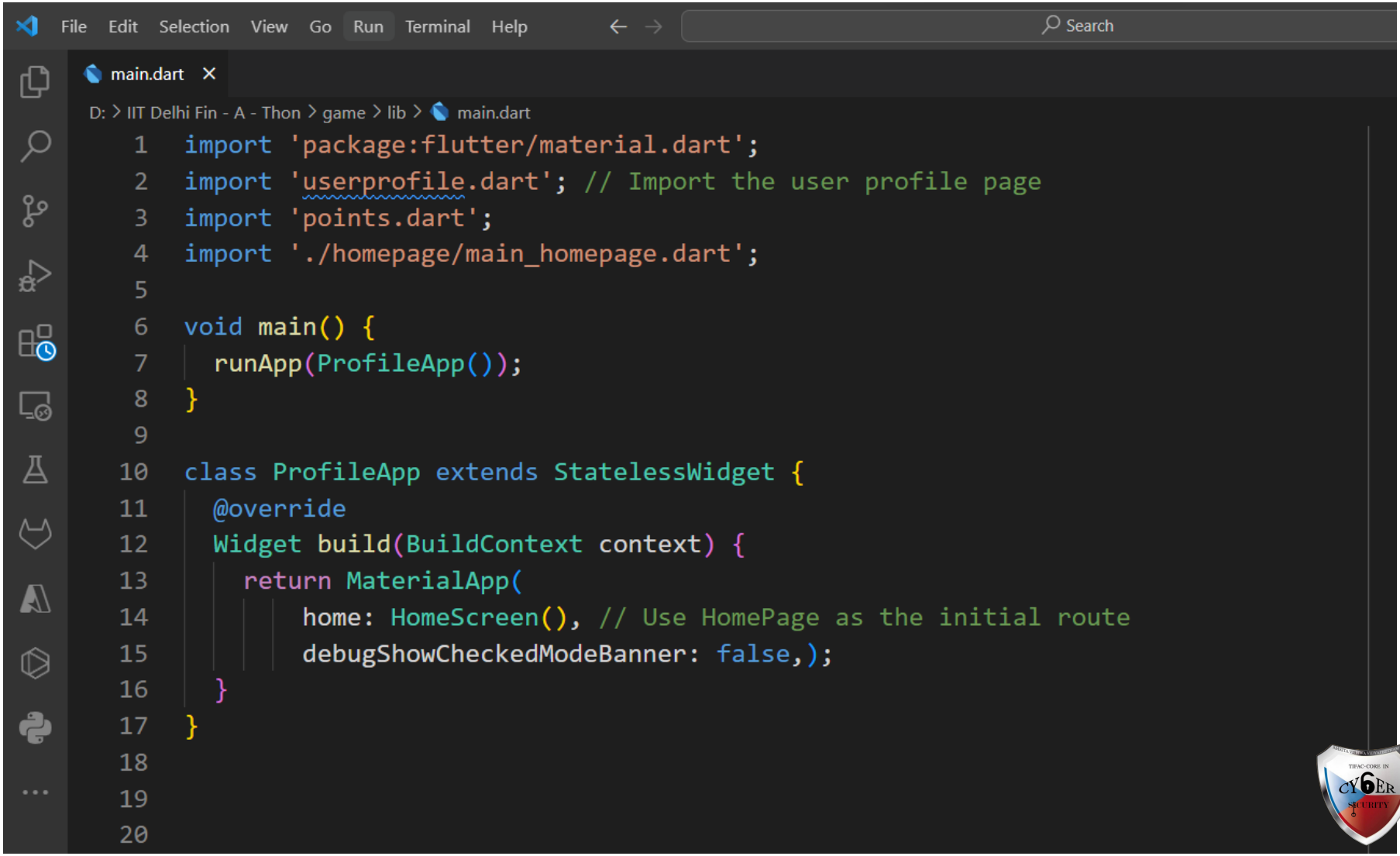
Use classpath of module <no module> | Interactive mode | Use REPL

```
1  val x = 1
2  val y = 2
3
4  var i = x + y
5  while(i++ < 8)
6      println(i)
7
8  println("Hello world!")
```

```
1  val x: Int
2  val y: Int
3
4  var i: Int
5  4
6  5
7  6
8  7
9  8
10
```

# Code Snippet ( Contd. )

- **Dart Code Snippet :**



# Comparison

- **Purpose :**
  - **Functional Programming Paradigm :**
    - Dealing with computation as the evaluation of mathematical functions.
  - **Object Oriented Programming Paradigm :**
    - Organizing code into objects, encapsulating data and behavior.
- **State Management :**
  - **Functional Programming Paradigm :**
    - Emphasizes immutability, avoiding shared state.
  - **Object Oriented Programming Paradigm :**
    - Uses objects to encapsulate and manage state.
- **Modularity :**
  - **Functional Programming Paradigm :**
    - Functions are modular and can be composed.
  - **Object Oriented Programming Paradigm :**
    - Encourages modularity through encapsulation and inheritance.
- **Data and Behavior :**
  - **Functional Programming Paradigm :**
    - Separation of data and behavior.
  - **Object Oriented Programming Paradigm :**
    - Objects encapsulate both data and behavior.

# Comparison ( Contd. )

- **Concurrency :**
  - **Functional Programming Paradigm :**
    - Embraces immutability and pure functions, easing concurrent programming.
  - **Object Oriented Programming Paradigm :**
    - Objects with shared state can complicate concurrency.
- **Abstraction :**
  - **Functional Programming Paradigm :**
    - Relies on higher-order functions and abstraction through composition.
  - **Object Oriented Programming Paradigm :**
    - Uses classes and objects for abstraction.
- **Flow of Data :**
  - **Functional Programming Paradigm :**
    - Emphasis on the flow of data through functions.
  - **Object Oriented Programming Paradigm :**
    - Data encapsulation within objects.
- **Composition :**
  - **Functional Programming Paradigm :**
    - Emphasizes the composition of functions to create more complex functions.
  - **Object Oriented Programming Paradigm :**
    - Composition through object composition and inheritance.

# Comparison ( Contd. )

- **Testing and Debugging :**
  - **Functional Programming Paradigm :**
    - Pure functions are easy to test; immutability aids debugging.
  - **Object Oriented Programming Paradigm :**
    - Object state can complicate testing; debugging may be more complex.
- **Design Patterns :**
  - **Functional Programming Paradigm :**
    - Often relies on functional programming patterns like map, filter, and reduce.
  - **Object Oriented Programming Paradigm :**
    - Common design patterns include Singleton, Observer, and Factory.
- **Problem Suitability :**
  - **Functional Programming Paradigm :**
    - Well-suited for problems with complex data transformations, parallelism, and stateless operations.
  - **Object Oriented Programming Paradigm :**
    - Well-suited for problems modeling real-world entities and systems with shared state.

# Discussions

- Both functional programming and object-oriented programming have their strengths and weaknesses.
- The choice between them often depends on the nature of the problem, the development team's preferences, and the project requirements.
- Understanding the characteristics and trade-offs of each paradigm can guide developers in making informed decisions in software design.
- Functional programming excels in scenarios with complex data transformations and stateless operations.
- Object-oriented programming is well-suited for modeling real-world entities and systems with shared state.
- Mastery of both paradigms can make a developer more versatile and adaptable to different project requirements.

# References


- https://www.geeksforgeeks.org/functional-programming-paradigm/
- https://www.educative.io/blog/functional-programming-vs-oop
- https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/
- https://learn.saylor.org/mod/page/view.php?id=22041
- https://www.scaler.com/topics/oops-advantages/
- https://www.javatpoint.com/dart-features
- https://krify.co/advantages-and-disadvantages-of-kotlin/
- https://www.spaceotechnologies.com/blog/kotlin-features/
- https://auberginesolutions.com/blog/dart-vs-kotlin-which-one-is-better-in-2023/
- https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/
- https://www.geeksforgeeks.org/kotlin-features/
- https://builtin.com/software-engineering-perspectives/kotlin
- https://www.geeksforgeeks.org/kotlin-programming-language/
- https://www.geeksforgeeks.org/benefits-advantages-of-oop/
- https://www.educba.com/object-oriented-programming-paradigm/