

Intro. OOP & FP. Scala basic syntax



- Repeat **Object Oriented** programming (**OOP**) paradigm
- Understand **Functional** programming (**FP**) paradigm
- Compare **FP** with **OOP**
- Make an overview of the **Scala** programming language
- Get familiar with **Scala** basic syntax

Object Oriented Programming



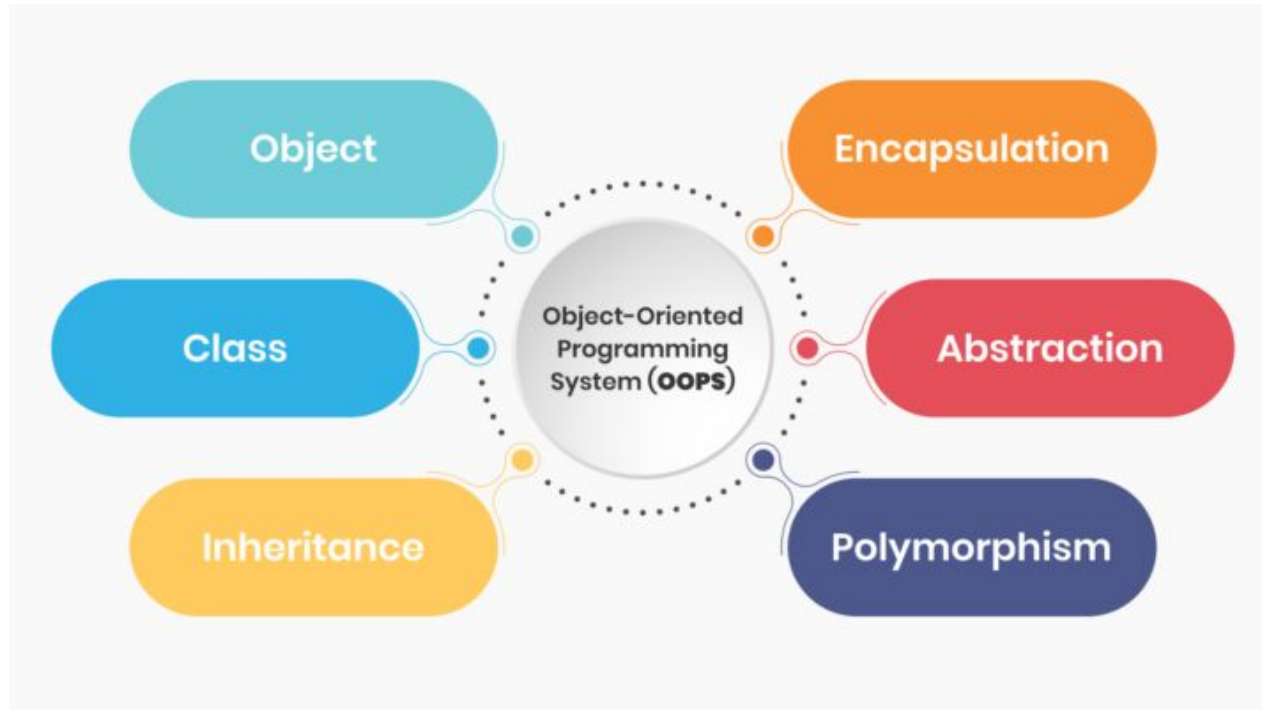
■ What is the greatest difficulty in software engineering?

- **Complexity**

What do I expect from my program besides doing the task correctly?

- ❑ easy to:
 - understand (**abstraction**)
 - change and refactor
 - add new features
- ❑ modular reusable and flexible code
- ❑ effective problem solving

Object Oriented Programming



Object Oriented Design Principles

6

[What is the SOLID?](#)



Examples of Objects



LightBulb

- **state/attributes**
 - on (true or false)
- **behavior**
 - switch on
 - switch off
 - check if on



Car

- **state/attributes**
 - # of gallons of gas in tank
 - total # of miles run so far
 - efficiency (mpg)
- **behavior**
 - drive
 - load gas
 - change efficiency
 - check gas
 - check odometer reading



BankAccount

- **state/attributes**
 - balance
- **behavior**
 - deposit
 - withdraw
 - check balance

Note

- each object is an "instance" of that "type" of object
- each instance has its own values for its attributes
 - e.g., different accounts can have different balances

■ What is the greatest difficulty in software engineering?

- Complexity
 - ❑ Software systems get replaced not when they wear out but when they crumble under their own weight because they have become too complex

Where does the complexity come from?

- Changing requirements
- Changing developers
- Attitudes

... we aren't sure!

Software generally becomes **more complex** the **older** it gets. Constant fight!

■ Why functional programming?

Because it removes one important **dimension** of **complexity**

- *To understand a program part (a function) you need no longer account for the possible **histories** of executions that can lead to that program part*

■ What is Functional Programming?

- Process of building software by
 - composing **Pure Functions** (**Referential Transparency**)
 - avoiding
 - **Mutable Data**
 - **Side Effects**
 - **Shared State**
- Application state flows through **Pure Functions**

How does functional code look like?

```
def double(i: Int): Int = i * 2
```

```
def isPrime(n: Int): Boolean =  
  n != 1 && (2 until n).forall(n % _ != 0)
```

```
def pureFunction(name : String): String = s"My name is $name"
```

```
def impureFunction(name : String): Unit = println(s"My name is $name")
```


Pros

- Pure functions much easier for parallelization and composition
- Declarative style of programming helps to define complex logic in a smaller piece of code
- Testing (especially Unit Testing)
- Ability to define pure core of your application
- Horizontal scalability

Cons

- High entry threshold comparing with OOP
- Difficult to switch your thinking from imperative to functional way

OOP vs FP



Encapsulation
Abstraction
Inheritance
Polymorphism



Pure Functions
Immutable Data
Referential Transparency
No side effects

Or use them together! Hybrid or synergy

Scala





Big Data, Data Science

Backend



Apache Flink



Scala programming language

- Statically typed + Exhaustiveness checking
- Object oriented
- Functional (contains various features and tools to build true functional code)
 - ☐ Higher order functions
 - ☐ Carrying
 - ☐ Match Expression
 - ☐ For Expression
 - ☐ Monads
 - ☐ Various frameworks and libraries (http4s, slick, doobie, cats, zio, akka ...)
 - ☐ ...
- JVM language
- Fully compatible with Java
- ~~Slow~~ Complex Compiler

How Tech Giants use Scala

- **Linkedin**
- **Twitter**
- **Netflix**
- **Tumblr**
- **Foursquare**
- **AirBnB**



NETFLIX



Twitter Technological Stack

- Originally built as a **Ruby on Rails** app (everything was pleasant due to active growing and horizontal scaling problem)
- Almost all *backend services* are moved to **Scala**
 - Though there is some use of plain **Java**
 - A few services are still in **Ruby on Rails**
 - Some services where **performance** is extremely important are using **C++**
- **Java, Kotlin, Objective-C, Swift** in *Mobile Development*
- **Python** is much more common on *Internal tools side* (also Bash)
- **Javascript with React** on the *UI*

Scala and Java

- Less amount of code even comparing with Java 8+ (2-3 times)
- More expressive
- (Scala) Poor support with such code quality tools like Sonar Lint/Cloud

Scala and Groovy

- Statically typed

Scala and Kotlin

- More production development
- Different use cases in production
- Rich libraries and frameworks ecosystem
- More tools for implementing true and complete FP

- [What's next for Scala?](#)

Unique IPs Over the Last 12 Months For com.lihaoyi



Books:

Essential Scala

Functional Programming in Scala

Practical FP in Scala: A hands-on approach

Other:

Tour of Scala & Scala Book from scala-lang.org

Rock the JVM courses - video courses

[Scala Exercises](#) - for practicing

[Coursera Scala Specialization](#)

Scala Basic Syntax

