Programming Paradigms Tutorials
Introduction

Programming paradigms

The term programming paradigm refers to a style of programming. It does not refer to a specific language, but rather it refers to the way you program.

During next tutorials we will focus on functional programming paradigm. The language which will be in use is Scala.

Functional programming

Functional programming is a programming paradigm where you have a style of building the structure and elements of computer programs. Here you treat computation as an evaluation of mathematical functions, and you avoid changing-state and mutable data.

Functional programming consists only of PURE functions. So, what do you understand by Pure functions?

Pure functions are those which take an argument list as an input and whose output is a return value. Now you may feel that all functions are pure as any function takes in values and returns a value.

For example, if a function relies on the global variable or class member’s data, then it is not pure. And in such cases, the return value of that function is not entirely dependent on the list of arguments received as input and can also have side effects. So, what do you understand by the term side effect? A side effect is a change in the state of an application that is observable outside the called function other than its return value. For example: Modifying any external variable or object property such as a global variable, or a variable in the parent function scope chain.

Features of Functional Paradigm

- Pure functions –if the input of the function is an array, the output will be a new array and the input array will not be modified.
- Recursion – A recursive function is a function that calls itself during its execution. This enables the function to repeat itself several times, the result being outputted at the end of each iteration.
- Referential transparency – An expression is said to be referentially transparent if it can be replaced with its corresponding value without changing the program's behavior. As a result, evaluating a referentially transparent function gives the same value for fixed arguments.
- Higher-Order Functions – A programming language is said to have First-class functions when functions in that language are treated like any other variable. For example, in such a language, a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a

value to a variable. Higher-order functions are functions that take at least one first-class function as a parameter.

- Variables are Immutable – In functional programming variables cannot be modify after they have been initialized. You can create new variables, and this allow to maintain state throughout the runtime of a program.

Scala

Standard Hello world program in Scala:

```
object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello, world!")
  }
}
```

The structure of this program is rather familiar to Java programmers. It consists of one method called main which takes the command line arguments – an array of strings, as parameter, as you can see name and type order is different in comparison to Java. The body of this method consists of a single call to the predefined method println with the friendly greeting as argument. The main method does not return a value. Therefore, its return type is declared as Unit which is similar to Java's void type.

What is less familiar to Java programmers is the object declaration containing the main method. Such a declaration introduces what is commonly known as a singleton object, that is a class with a single instance. The declaration above thus declares both a class called HelloWorld and an instance of that class, also called HelloWorld. This instance is created on demand, the first time it is used.

The astute reader might have noticed that the main method is not declared as static here. This is because static members (methods or fields) do not exist in Scala. Rather than defining static members, the Scala programmers declare these members in singleton objects.

Scala Basic Syntaxes and coding conventions

Case Sensitivity – Scala is case-sensitive, which means identifier Hello and hello would have different meaning in Scala.

Class Names – For all class names, the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case – this convention is also known as PascalCase or UpperCamelCase.
For example: class MyClassName

Method Names – All method names should start with a Lower Case letter. If multiple words are used to form the name of the method, then each inner word's first letter should be in Upper Case – this convention is also known as CamelCase.

Example: def myMethodName()

Program File Name – Name of the program file should exactly match the object name. When saving the file you should save it using the object name (Remember Scala is case-sensitive) and append ‘.scala’ to the end of the name.
Example: Assume 'HelloWorld' is the object name. Then the file should be saved as 'HelloWorld.scala'.

def main(args: Array[String]) – Scala program processing starts from the main() method which is a mandatory part of every Scala Program.

Exercise (1pt. each as extra point during tutorials):

1. Rewrite sample Hello world program that use first element of the array as a name and print it after hello word.
2. Write a function that sum numbers from array in Java.
3. Write a function that sum numbers from array using Scala and recursive way. Compare and explain differences between code from ex. 2 and this one.
4. What type is returned by below function method and what will be printed by this code on the screen:

```
object Main extends App {
  object MyObject{
    def method()={
      val x=2*5
      x+3}
  }
  println(MyObject.method())
}
```

5. What type is returned by below function add and what will be printed by this code on the screen:

```
val add=(x:Int,y:Int,z:Int)=>x+y+z
println(add(2,3,7))
```