A 'Scalable' Language

In the following lesson, we will go over what makes Scala a scalable programming language.

We'll cover the following

- Scala is Object-Oriented
- Scala is Functional

What does it mean for a language to be *scalable*? Scalability is the property of a language to grow with an increase in demands.

While there are many reasons for Scala's scalability, the most prominent is its combination of being both object-oriented and functional. There are other languages that provide both functional and object-oriented programming, but none are able to fuse them together into one uniform language the way Scala does.

Functional
Object-Oriented

Scala is Object-Oriented

Smalltalk was released in the 1970s and was one of the first pure object-oriented programming languages. You can now find object-oriented programming everywhere. It has dominated the world of programming languages.

The concept behind object-oriented programming is quite simple: All but the most trivial programs require some form of structure.

The most clear-cut way of achieving this is by using the concept of storage containers. A programming language can be broadly divided into data and operations to be performed on data. What we can do is store specific data and specific operations in some type of container. Furthermore, these containers are made to be general. Hence, they not only store data and operations but they are made to be general. Hence, they not only store data and operations but they are themselves values which can be stored in other containers and passed as

parameters to other operations. In object-oriented programming, these containers are known as *objects*.

Alan Kay, the inventor of Smalltalk, remarked that in this way the simplest object has the same construction principle as a full computer: it combines data with operations under a formalized interface. Martin Odersky, the inventor of Scala, infers that because of Alan Kay's remark, objects have a lot to do with language scalability: the same techniques apply to the construction of small programs as well as large programs.

And now, even though object-oriented programming can be found in a multitude of languages, very few actually follow the principles set by Smalltalk. Scala, on the other hand, is a pure object-oriented programming language with every value being an object and every operation is a method call. So, `+` is not just an operator in Scala, it is a method which is a part of a much larger class of methods and data.

Scala is Functional

While it is a pure object-oriented programming language, Scala is also fully functional.

Functional programming is guided by two main ideas. The first idea is that functions are first-class values. In programming languages, normal functions are passed values like strings and integers. In a functional programming language, functions hold the same status as integers and strings, hence, they can be passed as values to other functions and can also be returned as results of a function. Furthermore, like a variable, you can also declare a function within another function.

The second idea is that methods/functions should not have side effects. In other words, when a method is called on some value, it does not change the data in place, rather it creates a new modified value. This is why in Scala variables and data structures are immutable.

In the next lesson, we will explore the primary features of Scala.