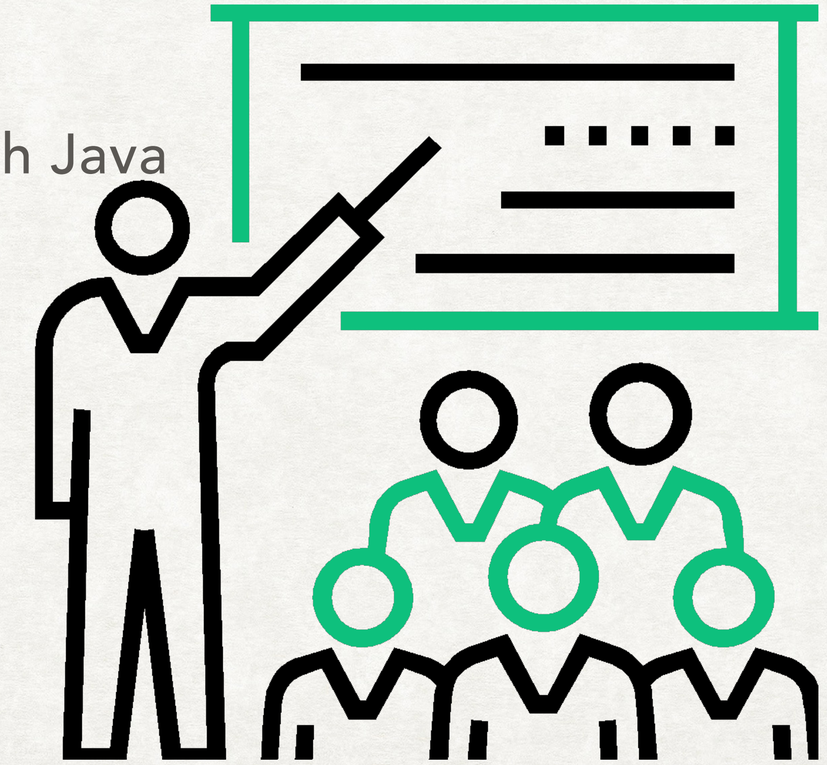
ADVANCED FUNCTIONAL THINKING

DR SURBHI SARASWAT
SCHOOL OF COMPUTER SCIENCE

MODULE OBJECTIVES

At the end of this module, you will be able to:

- Provide an introduction to Scala
- Scala features
- Discuss OOPs concepts within Scala compared with Java



1.1 HISTORY OF SCALA

Martin Odersky created and developed Scala in 2001

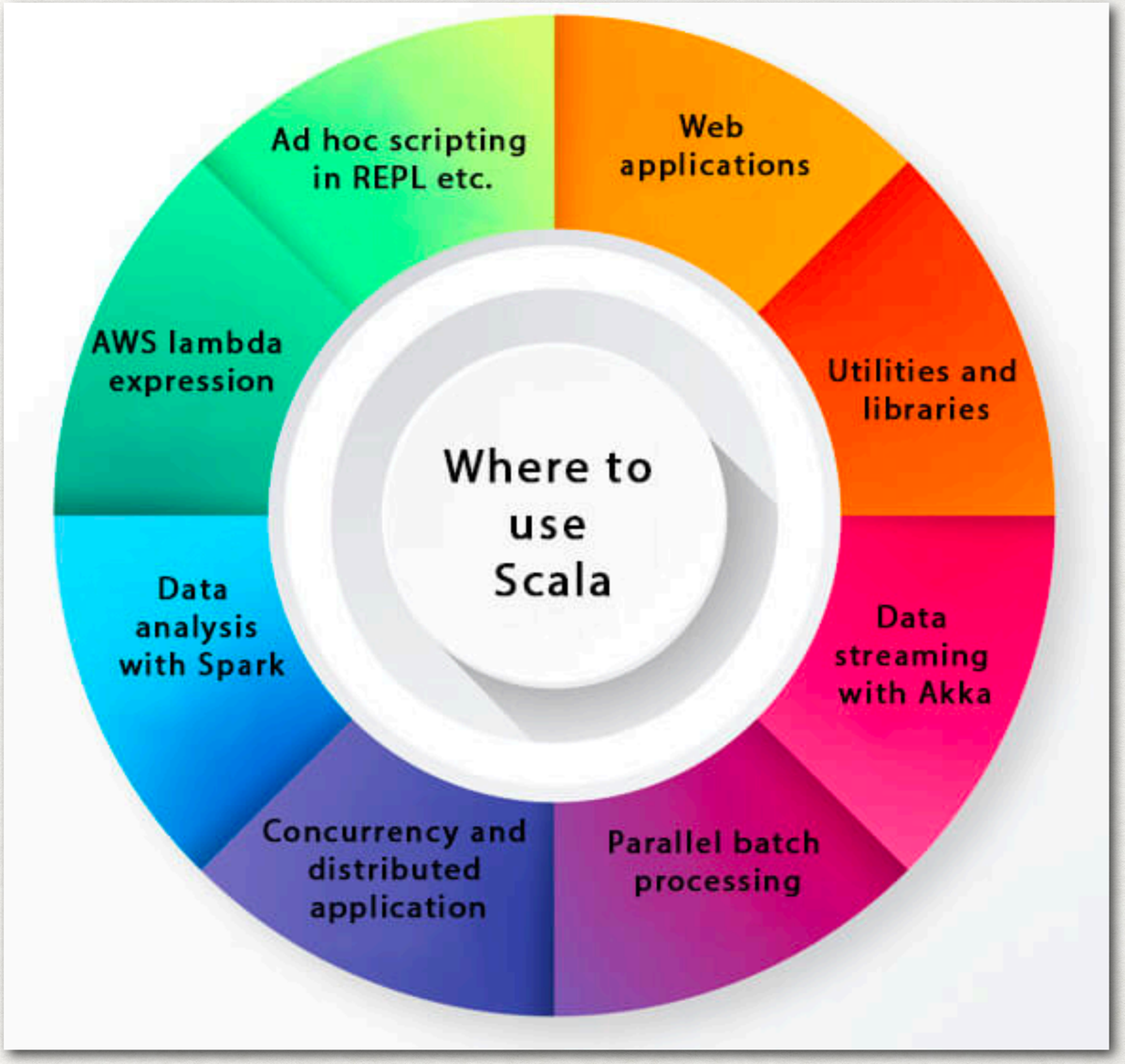
In 2006, released Scala 2.0

In 2004, released as Java Platform 1.5 version

In 2011, Typesafe Inc. founded to support and promote Scala.

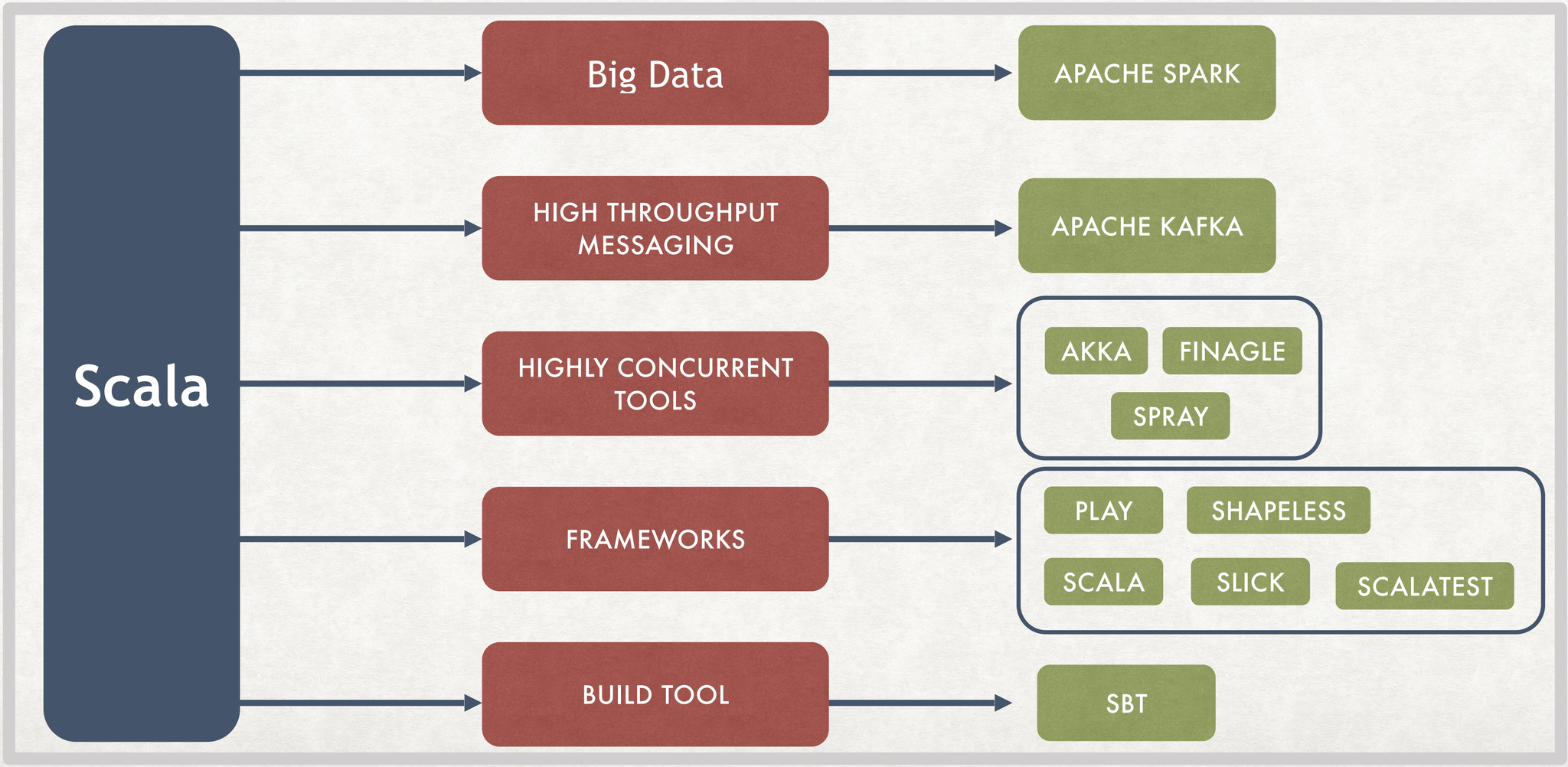
- It was designed by Martin Odersky. It was officially released for java platform in early 2004 and for .Net framework in June 2004. Later on, Scala dropped .Net support in 2012.
- Scala is influenced by Java, Haskell, Lisp, Pizza etc. and influenced to F#, Fantom, Red etc.
- File extension of scala source file may be either .scala or .sc.
- You can create any kind of application like web application, enterprise application, mobile application, desktop based application etc.

1.1.1 USERS OF SCALA



1.1.2 SCALA ECOSYSTEM

- In Scala, you can create any type of application in less time and coding whether it is a web-based, mobile-based or desktop-based application. Scala provides you with powerful tools and API by using which you can create applications.



SCALA ECOSYSTEM

- Big Data:
  - Apache Spark is a leading open source platform for large scale data processing.
- High Throughput Messaging:
  - Kafka is a high throughput distributed messaging system.
- Highly Concurrent Systems:
  - Akka from Lightbend (formerly known as Typesafe) allows you to build highly concurrent and distributed systems.
  - Finagle from Twitter allows you to create highly concurrent servers where the underlying APIs are protocol agnostic.
  - Spray is a high-throughput HTTP server.
- Frameworks:
  - Play from Lightbend (formerly known as Typesafe) allows you to easily build scalable web applications.
  - Shapeless brings a lot of added functionality when it comes to dealing with types.
  - ScalaTest allows you to test your Scala applications easily.
  - Slick is a rich data access layer.
  - Scalaz provides additional semantics for functional programming.
- Build Tool:
  - SBT is a popularly used to develop Scala applications.

FEATURES OF SCALA

- Scalable
- Concise
- Object-Oriented
- High-Level
- Functional
- Statically Typed
- Compatible
- Interactive (REPL)

FEATURES OF SCALA

Scalable: Scala, short for Scalable Language, is a hybrid functional programming language.

- Scala smoothly integrates the features of object-oriented and functional languages.
- Scala is compiled to run on the Java Virtual Machine.
- Many existing companies, that depend on Java for business-critical applications, are turning to Scala to boost their development productivity, applications scalability and overall reliability.

Object-oriented: Scala is a pure object-oriented language in the sense that every value is an object. Classes are extended by **subclassing** and a flexible **mixin-based composition** mechanism as a clean replacement for multiple inheritances.

- Scala is an object-oriented language in pure form.
- Every value is an object and every operation is a method call.
- For example, when you say 1 + 2 in Scala, you are actually invoking a method named + defined in class Int.
- You can define methods with operator-like names that clients of your API can then use in operator notation.

FEATURES OF SCALA

Functional: Scala is also a functional language in the sense that every function is a value and every value is an object so ultimately every function is an object.

- Scala provides a lightweight syntax for defining **anonymous functions,**
- It supports **higher-order functions**,
- It allows functions to be **nested**, and supports **currying**.

- **Compatible**
  - Scala is compiled into Java Byte Code which is executed by the Java Virtual Machine (JVM). This means that Scala and Java have a common runtime platform. You can easily move from Java to Scala.
  - The Scala compiler compiles your Scala code into Java Byte Code, which can then be executed by the 'scala' command. The 'scala' command is similar to the java command, in that it executes your compiled Scala code.
  - Scala can Execute Java Code. Scala enables you to use all the classes of the Java SDK and also your own custom Java classes, or your favorite Java open source projects.
  - Another aspect of full interoperability is that Scala heavily re-uses Java types. Scala's Ints are represented as Java primitive integers of type int, Floats are represented as floats, Booleans as booleans, and so on. Scala arrays are mapped to Java arrays. Scala also re-uses many of the standard Java library types.

1.1.3 FEATURES OF SCALA

Concise: Scala programs tend to be short.

- Scala programmers have reported reductions in the number of lines of up to a factor of ten compared to Java. These might be extreme cases.
- A more conservative estimate would be that a typical Scala program should have about half the number of lines of the same program written in Java.
- Fewer lines of code mean not only less typing, but also less effort at reading and understanding programs and fewer possibilities of defects.
- There are several factors that contribute to this reduction in lines of code.

High-Level:

- Programmers are constantly grappling with complexity. To program productively, you must understand the code on which you are working. Overly complex code has been the downfall of many a software project. Unfortunately, important software usually has complex requirements. Such complexity can't be avoided; it must instead be managed.
- Scala helps you manage complexity by letting you raise the level of abstraction in the interfaces you design and use. As an example, imagine you have a String variable name, and you want to find out whether or not that String contains an upper case character.

1.1.4 SCALA IS CONCISE

In Java, a class with a constructor often looks like this

```
 // this is Java
 class MyClass {
    private int index;
     private String name;
    public MyClass(int index, String name) {
this.index = index;
    this.name = name;
    }
 }
```

In Scala , you would likely to write this instead:

```
 class MyClass(index: Int, name: String)
```

1.1.5 SCALA IS HIGH LEVEL

In Java, a class with a constructor often looks like this

```
boolean nameHasUpperCase = false;  // this is Java
for (int i = 0; i < name.length(); ++i) {
     if (Character.isUpperCase(name.charAt(i))) {
          nameHasUpperCase = true;
          break;
    }
 }
```

In Scala , you would likely to write this instead:

```
val nameHasUpperCase = name.exists(_.isUpper)
```

1.1.3 FEATURES OF SCALA

Statically Typed: Scala, unlike some of the other statically typed languages (C, Pascal, Rust, etc.), does not expect you to provide redundant type information. You don't have to specify a type in most cases, and you certainly don't have to repeat it.

Abstractions are used in a safe and coherent manner. In particular, the type system supports:

- **Generic classes:** Scala has built-in support for classes parameterized with types. This can restrict the reuse of the class abstraction. Such generic classes are particularly useful for the development of collection classes.
- **Polymorphic methods:** Methods in scala can parameterize by type or values.
- A static type system classifies variables and expressions according to the kinds of values they hold and compute. Scala stands out as a language with a very advanced static type system. Starting from a system of nested class types much like Java's, it allows you to parameterize types with generics, to combine types using intersections, and to hide details of types using abstract types. These give a strong foundation for building and composing your own types, so that you can design interfaces that are at the same time safe and flexible to use.
- If you like dynamic languages, such as Perl, Python, Ruby, or Groovy, you might find it a bit strange that Scala's static type system is listed as one of its strong points. After all, the absence of a static type system has been cited by some as a major advantage of dynamic languages. The most common arguments against static types are that they make programs too verbose, prevent programmers from expressing themselves as they wish, and make impossible certain patterns of dynamic modifications of software systems. However, often these arguments do not go against the idea of static types in general, but against specific type systems, which are perceived to be too verbose or too inflexible. For instance, Alan Kay, the inventor of the Smalltalk language, once remarked: "I'm not against types, but I don't know of any type systems that aren't a complete pain, so I still like dynamic typing."

FEATURES OF SCALA

- **Scala can do Concurrent & Synchronize processing**
  - Scala allows you to express general programming patterns in an effective way. It reduces the number of lines and helps the programmer to code in a type-safe way. It allows you to write codes in an immutable manner, which makes it easy to apply concurrency and parallelism (Synchronize).
- **Interactive (REPL):** The interactive mode of the scala command is very convenient for experimentation. An interactive interpreter like this is called a REPL: Read, Evaluate, Print, Loop.
  - Experimenting with the scala command in the interactive mode (REPL) is a great way to learn the details of Scala.
  - If you type scala on the command line without a file argument, the interpreter runs in interactive mode. You type in definitions and statements that are evaluated on the fly. If you give the command a scala source file argument, it compiles and runs the file as a script, as in our scala upper1-script.scala example. Finally, you can compile Scala files separately and execute the class file, as long as it has a main method, just as you would normally do with the java command.
  - Whenever we refer to executing a script, we mean running a Scala source file with the scala command.

1.2 SCALA VS JAVA

SCALA	JAVA
Scala is a mixture of both object oriented and functional programming.	Java is a general purpose object oriented language.
The process of compiling source code into byte code is slow.	The process of compiling source code into byte code is fast.
Scala support operator overloading.	Java does not support operator overloading.
Scala supports lazy evaluation.	Java does not support lazy evaluation.
Scala is not backward compatible.	Java is backward compatible means the code written in the new version can also run in older version without any error.
Any method or function present is Scala are treated like they are variable.	Java treats functions as an object.
In Scala, the code is written in compact form.	In Java, the code is written in long form.
Scala variables are by default immutable type.	Java variables are by default mutable type.
Scala treated everything as an instance of the class and it is more object oriented language as compare to Java.	Java is less object oriented as compare to Scala due to presence of primitives and statics.
Scala does not contain static keyword.	Java contains static keyword.
In Scala, all the operations on entities are done by using method calls.	In Java, operators are treated differently and is not done with method call.