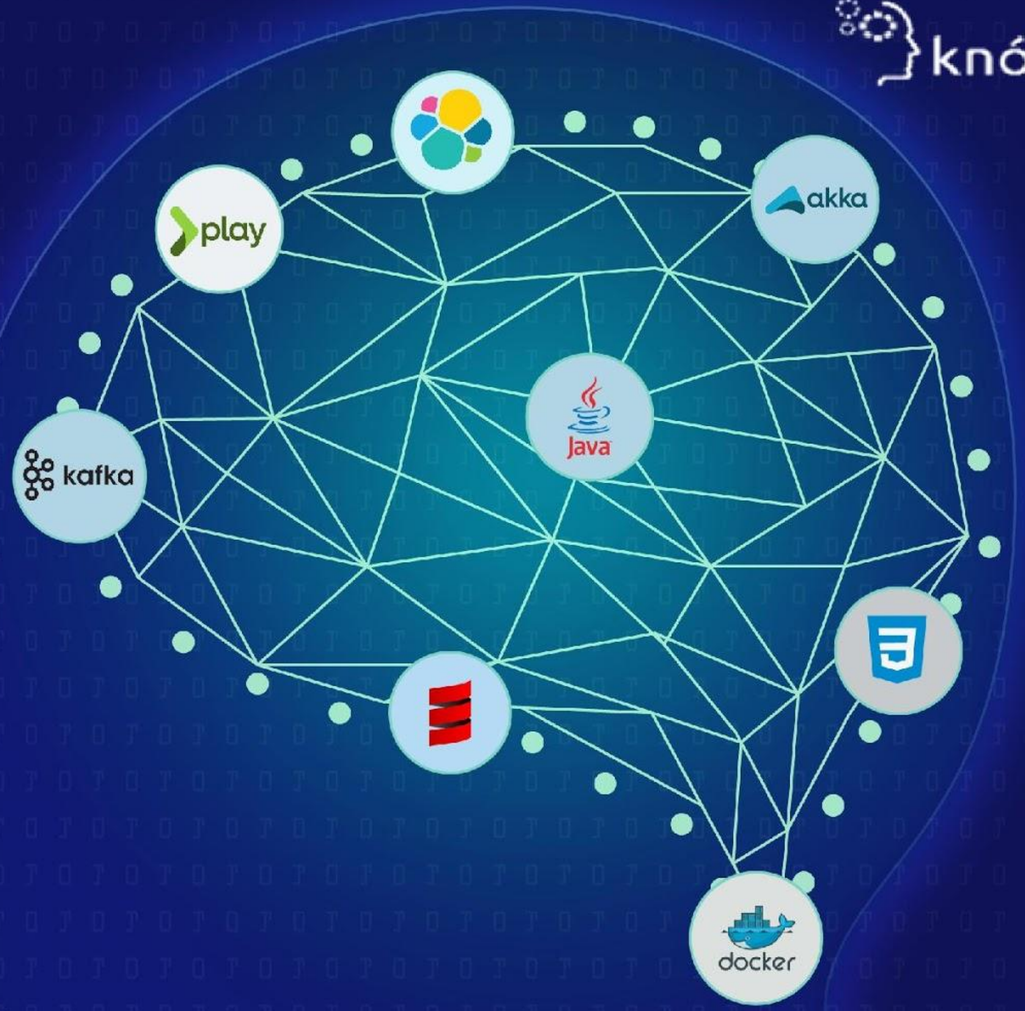
# Classes & Properties




**Presented By**:
Himanshu Gupta

# Agenda

- Creating a Primary Constructor
- Controlling the Visibility of Constructor Fields
- Defining Auxiliary Constructors
- Defining a Private Primary Constructor
- Providing Default Values for Constructor Parameters

# Creating a Primary Constructor

Problem

We want to create a primary constructor for a class, and we will see that the approach is different than Java.

# Creating a Primary Constructor

Solution

The primary constructor of a Scala class is a combination of:

- The constructor parameters.
- Methods that are called in the body of the class.
- Statements and expressions that are executed in the body of the class.

# Creating a Primary Constructor

Example

## Demo

# Controlling the Visibility of Constructor Fields

Problem

We want to control the visibility of fields that are used as constructor parameters in a Scala class.

# Controlling the Visibility of Constructor Fields

Solution

The visibility of constructor fields in a Scala class is controlled by whether the fields are declared as val, var, without either val or var, and whether private is also added to the fields:

- If a field is declared as a var, Scala generates both getter and setter methods for that field.
- If the field is a val , Scala generates only a getter method for it.
- If a field doesn’t have a var or val modifier, Scala gets conservative, and doesn’t generate a getter or setter method for the field.
- Additionally, var and val fields can be modified with the private keyword, which prevents getters and setters from being generated.

# Controlling the Visibility of Constructor Fields

Example

## Demo

# Defining Auxiliary Constructors

Problem

We want to define one or more auxiliary constructors for a class to give consumers of the class different ways to create object instances.

# Defining Auxiliary Constructors

Solution

There are several important points to its recipe:

- Auxiliary constructors are defined by creating methods named this.
- Each auxiliary constructor must begin with a call to a previously defined constructor.
- Each constructor must have a different signature.
- One constructor calls another constructor with the name this.

# Defining Auxiliary Constructors

Example

## Demo

# Defining a Private Primary Constructor

Problem

We want to make the primary constructor of a class private, such as to enforce the Singleton pattern.

# Defining a Private Primary Constructor

Solution

To make the primary constructor private, insert the private keyword in between the class name and any parameters the constructor accepts.

# Defining a Private Primary Constructor

Example

## Demo

# Providing Default Values for Constructor Parameters

Problem

We want to provide a default value for a constructor parameter, which gives other classes the option of specifying that parameter when calling the constructor, or not.

# Providing Default Values for Constructor Parameters

Solution

Give the parameter a default value in the constructor declaration.

# Providing Default Values for Constructor Parameters

Example

## Demo

# Overriding Default Accessors and Mutators

Problem

We want to override the getter or setter methods that Scala generates for us.

# Overriding Default Accessors and Mutators

Solution

The recipe for overriding default getter and setter methods is:

- Create a private var constructor parameter with a name you want to reference from within your class.
- Define getter and setter names that you want other classes to use.
- Modify the body of the getter and setter methods as desired.

# Overriding Default Accessors and Mutators

Example

## Demo

# Q/A


Thanks