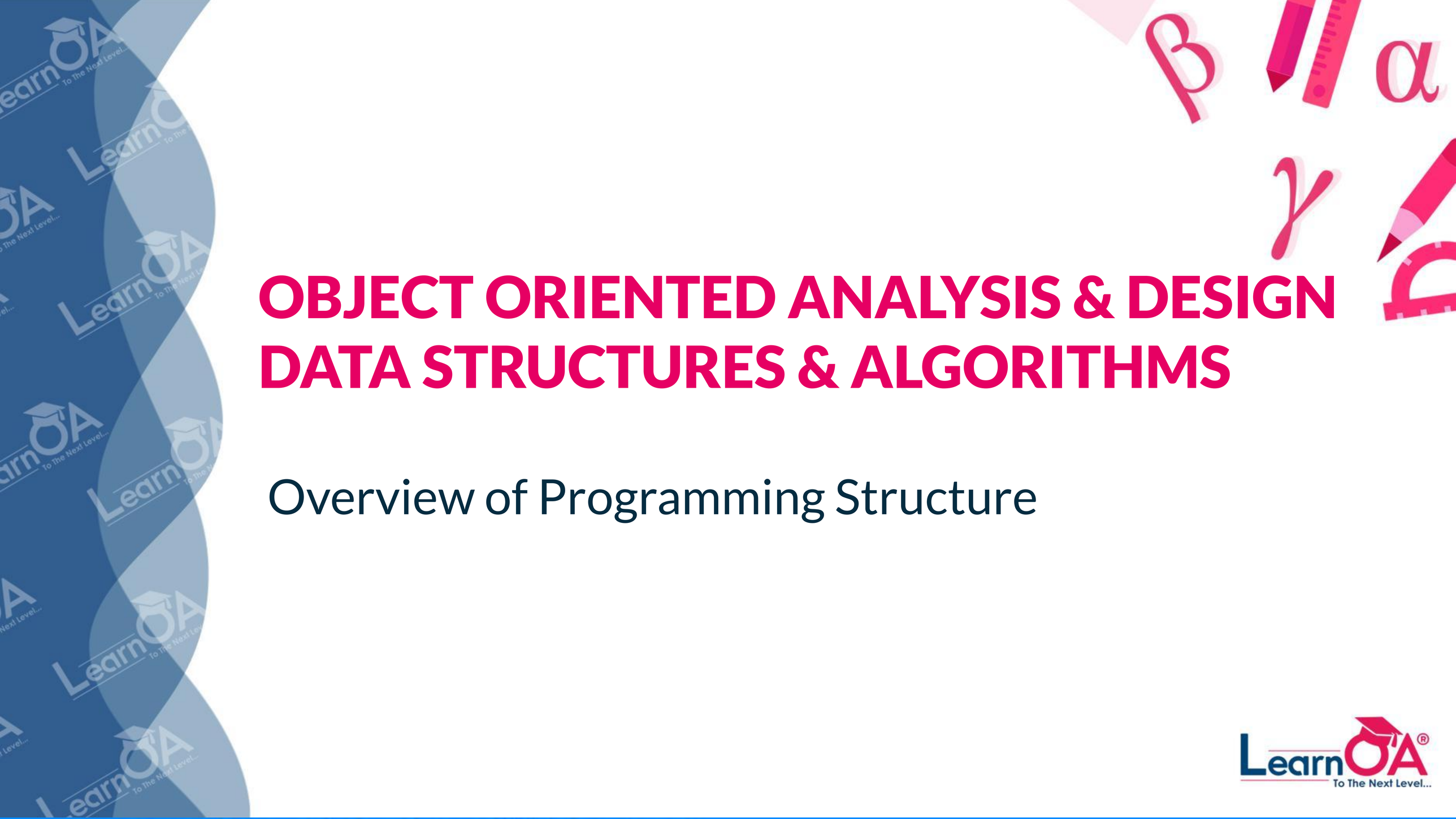


OBJECT ORIENTED ANALYSIS & DESIGN
DATA STRUCTURES & ALGORITHMS

Overview of Programming Structure

Object Oriented Programming (OOPs)

Object-Oriented Programming is a methodology or paradigm to design a program using Classes and Objects

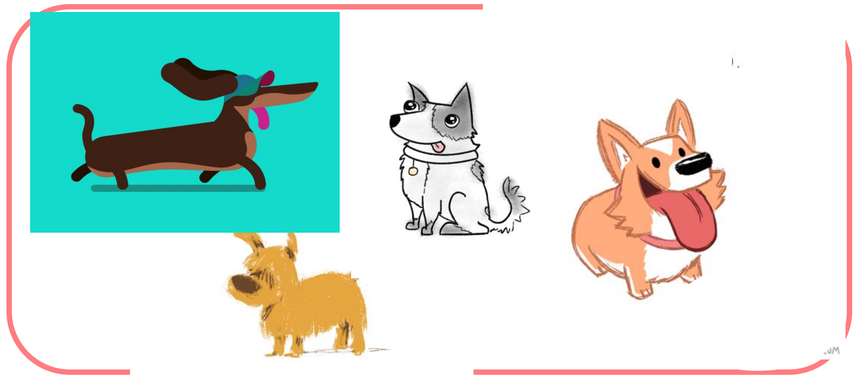
Object Oriented Programming (OOPs)

For different values of data members (breed size, age, and color) in Java class, you will get a different dog object

Object Oriented Programming (OOPs)

Real world entities that has their own properties and behaviours

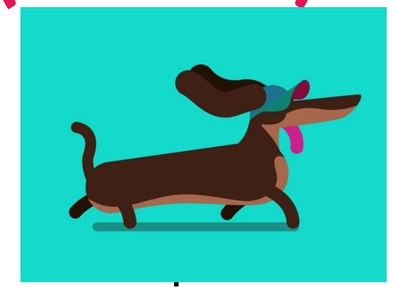
Class Dog



Blueprint from which an objects properties and behaviours are decided

Properties	Behaviour
`breed`	`eat()`
`size`	`sleep()`
`age`	`run()`
`color`	`bark()`

Object Oriented Programming (OOPs)



breed size age color	breed size age color	breed size age color	breed size age color
Behaviour	**Behaviour**	**Behaviour**	**Behaviour**
eat() sleep() run() bark()	eat() sleep() run() bark()	eat() sleep() run() bark()	eat() sleep() run() bark()

Java Classes

A class is a **blueprint of object** having properties

```
class <class_name>{
    field;
    method;
}
```

Java Classes (Contd.)

A class may contain

```
class <class_name>{
    field;
    method;
}
```

- Fields
- Methods
- Constructors
- Blocks
- Nested Class & Interface

Java Objects

A real-world entity that has state and behaviour is known as an object

State 01

It is the data (value) of an object

02 Behavior

It is the functionality) of an object

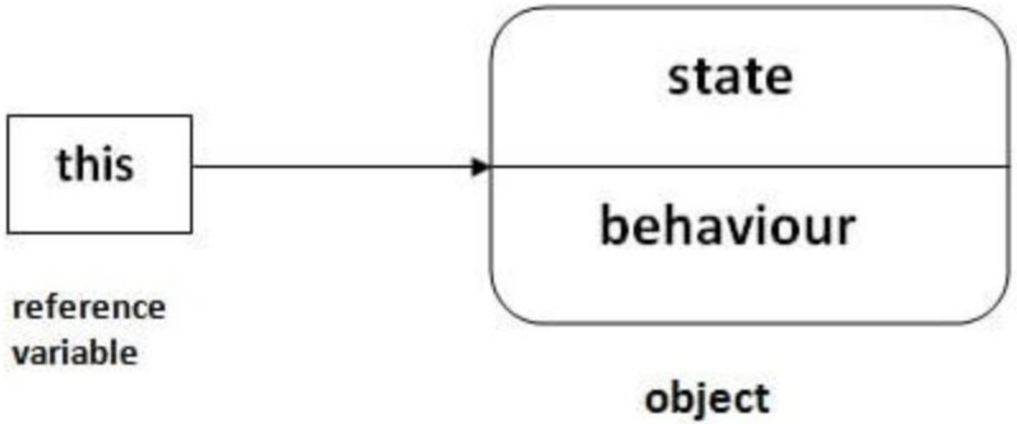
03 Identity

The object identity is typically implemented via a unique ID that is used internally by the JVM

CHARACTERISTICS

this keyword

There can be a lot of usage of Java this keyword. In Java, this is a reference variable that refers to the current object.



Java Methods

Java Methods

A method is a set of code that is grouped together to perform a specific operation

A method must be written inside a class

Each method has its own signature

Java provides two types of methods

- Pre Defined or Standard Library Methods
- User Defined Methods

Java User Defined Methods

To use a method, you need to perform two steps:

- Method Initialization
- Method Invocation

Java Methods

Method Initialization

```
modifier returnType nameOfMethod (Parameter List)
{
    // method body
}
```

- ✔ A method can be parameterized or non-parameterized
- ✔ Method definition consists of a method header and a method body
- ✔ You can ***Overload Method*** i.e. Provide same name to more than one method but their data type or parameter list must be different

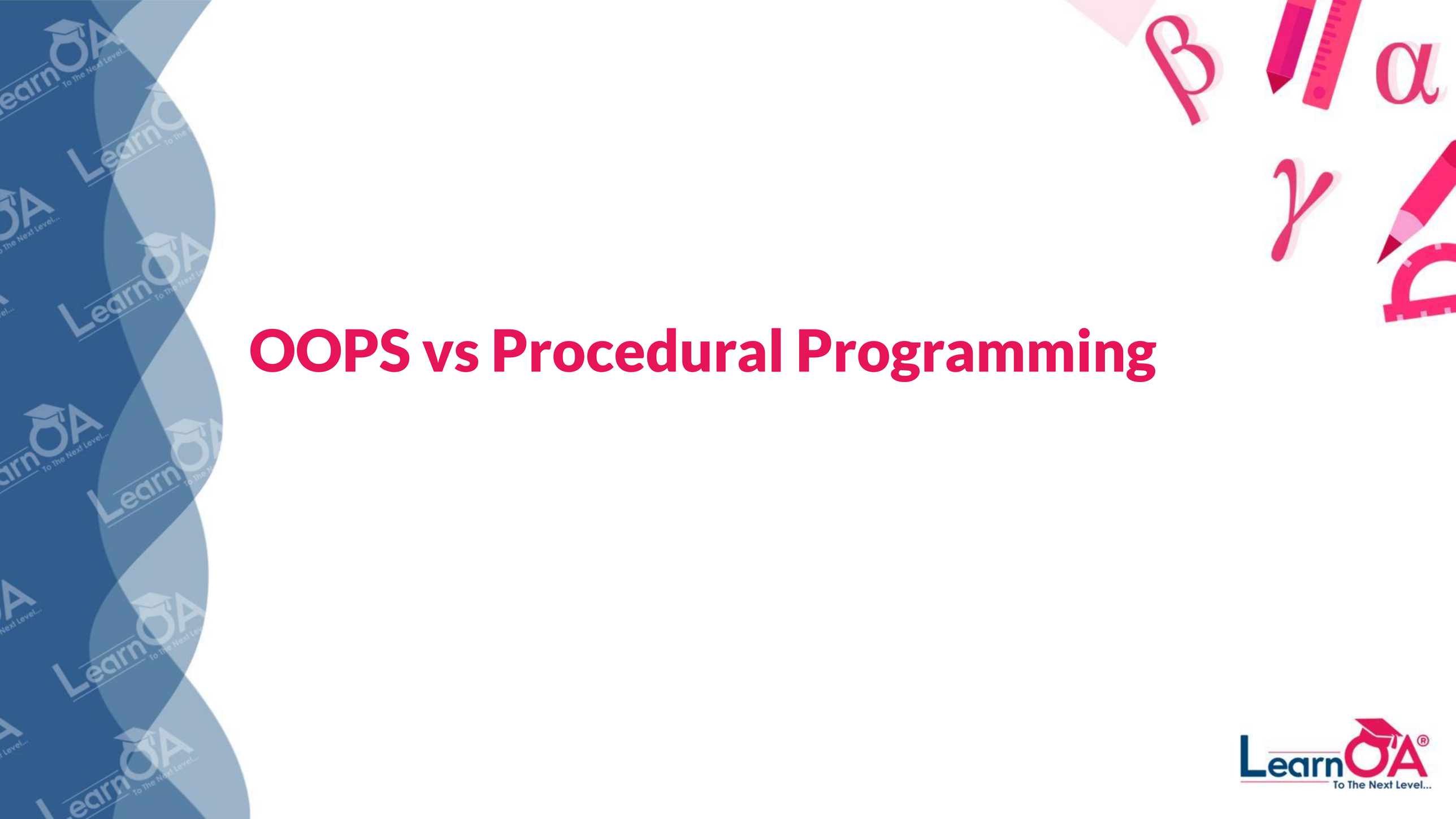
Java Methods

Method Invocation

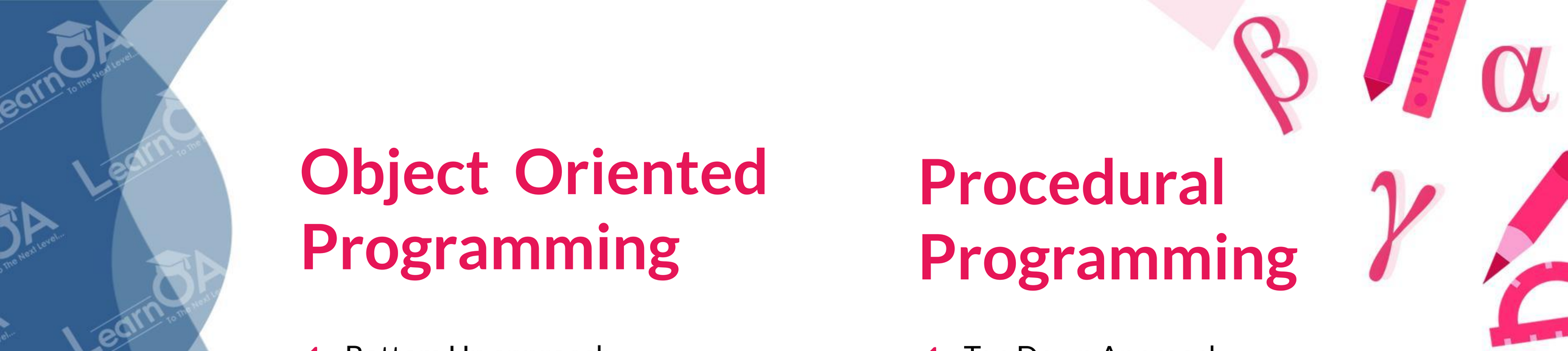
```
methodName()

methodName(parameter1, parameter2...)
```

- ✔ To use a method it needs to be invoked or called
- ✔ When a program invokes a method, the program control gets transferred to the called method
- ✔ A method can be called in two ways:
  - Call by Value
  - Call by Reference



OOPS vs Procedural Programming



Object Oriented Programming

- ✔ Bottom Up approach
- ✔ Divided into objects
- ✔ Has ***Access Modifiers***
- ✔ Objects can move & communicate with each other through member functions
- ✔ More secure
- ✔ Supports ***overloading***

Procedural Programming

- ✔ Top Down Approach
- ✔ Divided into functions
- ✔ Doesn’t have Access Modifiers
- ✔ Data can move freely from function to function in the system
- ✔ Less Secure
- ✔ Do not support overloading

Static Vs Non-Static

Static vs Non Static

Non-static variable	Static variable
• Non-static variable also known as instance variable while because memory is allocated whenever is created. • Non-static variable are specific to an object. • Non-static variable can access with object reference. • **Syntax**	• Memory is allocated at the time of loading of class so that these are also known as class variables. • Static variable are common for every object that mean these memory location can be shareable by every object reference or same class. • static variable can access with class reference. • **Syntax**
Obj_ref.variable_name	class_name.variable_name

Static vs Non Static

Non-static Method	Static Method
• These method never be preceded by static keyword **Example:**	• These method always preceded by static keyword **Example:**
void fun() { .... }	static void fun() { .... }
• Memory is allocated multiple time whenever method is calling.	• Memory is allocated only once at the time of class loading.


Thank You!