Sri Eshwar College of Engineering
An Autonomous Institution
Affiliated to Anna University, Chennai

Celebrating 15 Years of Excellence

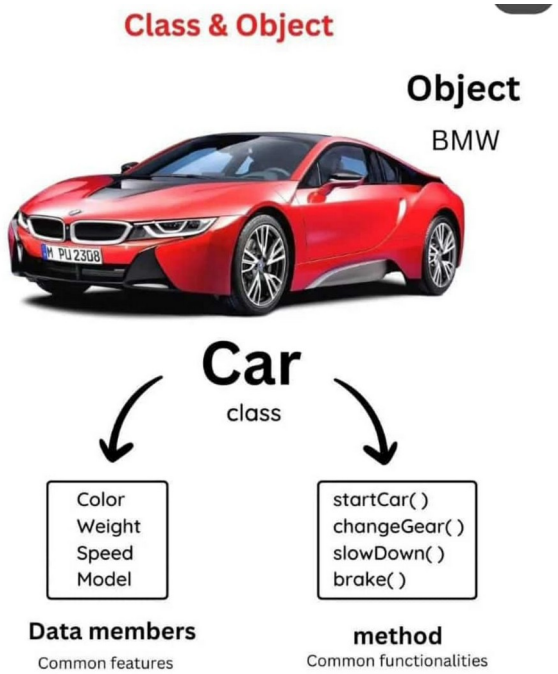
Overview of OOP

U23CS491 - OOPs using JAVA

Session by

Mr R. GIRIDHARAN, AP/CSE

- Introduction to OOP
- OOP Paradigms
- Building Blocks of OOP
- 4 Principles of OOP



Module I: Foundations of JAVA

Overview of OOP – Object-oriented programming paradigms - Features of Object Oriented Programming – Java Buzzwords - Overview of Java - JVM - JDK - Programming Structures in Java - Classes & its types in Java – Data Types, Variables - Operators - Keywords - Control Statements - Wrapper Classes -Constructors - Methods - Access specifiers - Arrays & its types - java.util.Arrays - Doc comments - I/O classes

Course Outcomes:

CO No	Outcome	K Level
CO1	**(Understand)** Understand the core concepts of Java programming	K2
CO2	**(Understand)** Understand the principles of object-oriented programming	K2
CO3	**(Understand)** Understand the concepts of strings and collections	K2
CO4	**(Apply)** Apply Exception-Handling and Multithreading concepts in applications	K3
CO5	**(Apply)** Apply JavaFX & JDBC in application development	K3

Introduction to OOP:

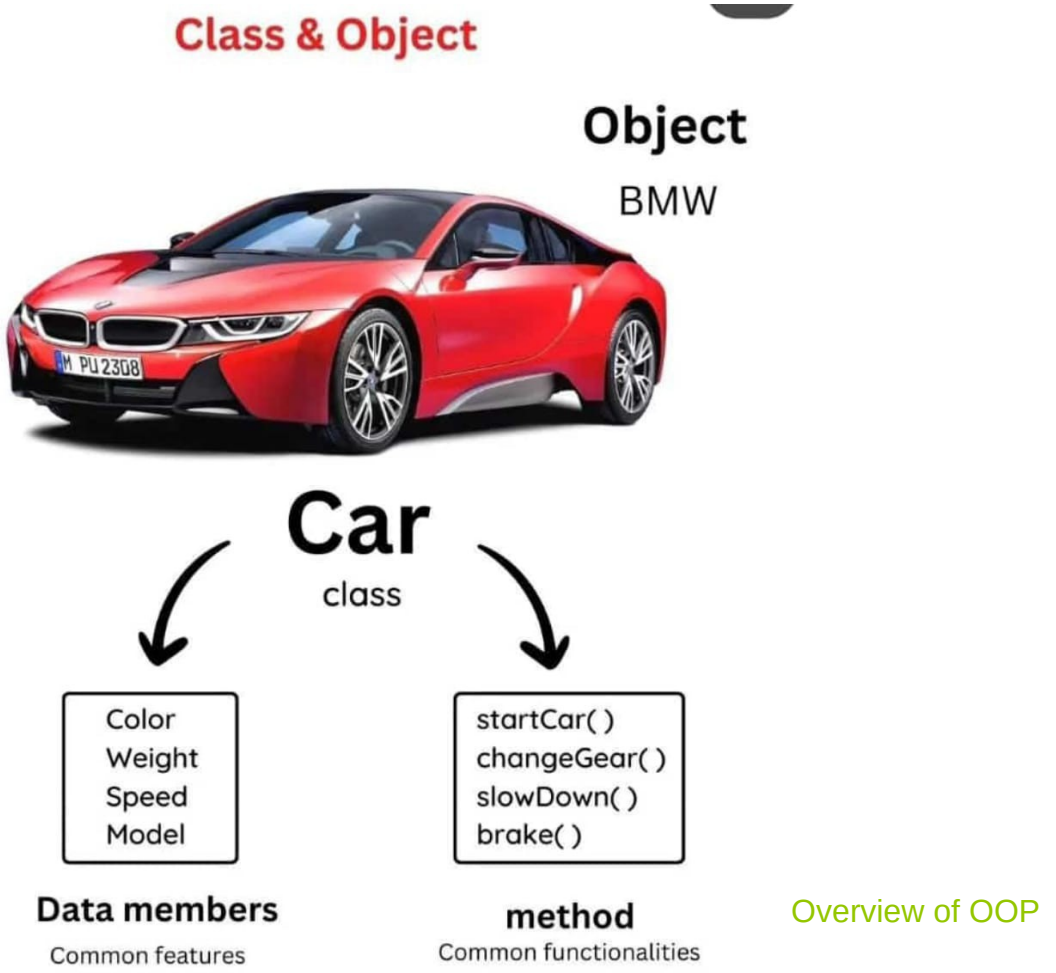
- OOP stands for **Object-Oriented Programming**.
- Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.
- Object-oriented programming has several advantages over procedural programming:
  - OOP is faster and easier to execute
  - OOP provides a clear structure for the programs
  - OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
  - OOP makes it possible to create full reusable applications with less code and shorter development time

Object Oriented Programming Paradigms:

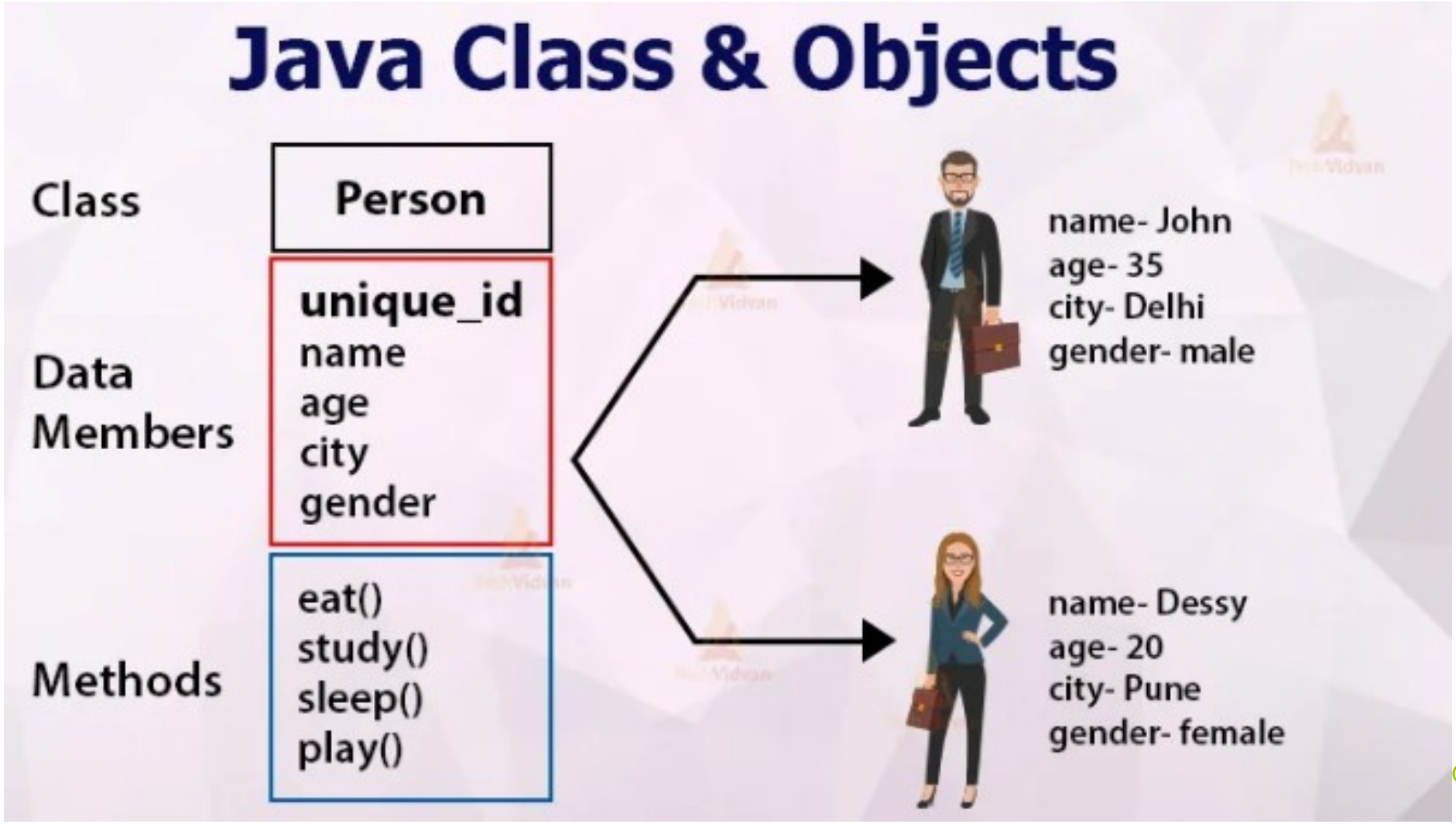
- Object-Oriented Programming (OOP) is a programming paradigm in computer science that relies on **classes** and **objects**.
- It is used to structure a software program into simple, reusable pieces of code **blueprints** (usually called **classes**), which are used to create **individual instances** of **objects**.

Building blocks of OOPs:

- **Class:** A template or blueprint or prototype
- **Object:** A instance of the class



Class and Objects:



Class:

- Class is a **template** or **blueprint** of an **object**, and an **object** is an **instance** of a **class**. It defines the **shape** and **nature** of an **object**.
- Once a **class** is **defined**, any **number** of **objects** can be **created** from that **class**.

```
class Vehicle{
    int numberOfWheels;
    String brandName, color;
    double price;
    void start( );
    void changeGear( );
}
```

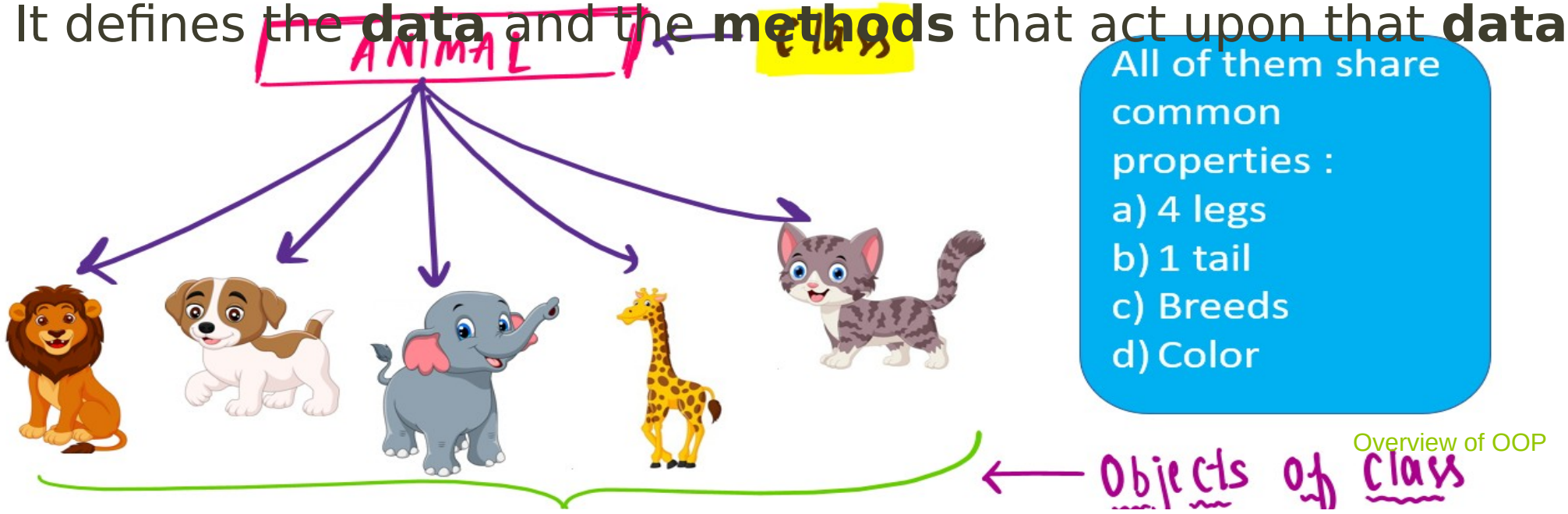

```
Vehicle car = new Vehicle( );
```

```
Vehicle bike = new Vehicle( );
```


```
Vehicle truck = new Vehicle( );
```

Class:

- It is a **logical entity** that **does not occupy** any **space/memory**.
- **Memory** is allocated only when we **create** the **objects** of a **class**. A class contains **properties** and **methods** to define the **state** and **behaviour** of its object.
- It defines the **data** and the **methods** that act upon that **data**.



Object:

- A class is a **template** or **blueprint** from which **objects** are **created**.
- So, an **object** is a class's **instance** (result).
- An object is *a **real-world entity***.
- The object is an ***entity*** that has a ***state*** and ***behavior***.

Four Principles of OOP:

- **Inheritance:** child classes inherit data and behaviours from the parent class. **Use: Code Reusability**
- **Encapsulation:** containing information in an object, exposing only selected information. **Use: Code Security**
- **Abstraction:** only exposing high-level public methods for accessing an object. **Use: Hides Complexity**
- **Polymorphism:** many methods can do the same task. **Use: Code Reusability**

Inheritance:

- Inheritance is the mechanism in which one class (child) acquire all the features of another class (Parent).
- We can achieve inheritance by using the **extends keyword**.
- It facilitates the reusability of the code.

Mom and Daughter

Some properties of mom inherits by her daughter

Encapsulation:

- An encapsulation is the process of **binding data** and **functions** into a **single unit.**
- A **class** is an example of encapsulation.

School bag can keep your book, pen, erasers, lunch box so on ...

Abstraction:

- A method of **hiding irrelevant information from** the **user**.
- We can make a class abstract by using the keyword **abstract**.
- We use **abstract class** and **interface** to achieve abstraction.

Even though it performs a lot of actions it doesn't show us the process. It has hidden its process by showing only the main things like getting inputs and giving the output.

Polymorphism:

- The polymorphism is the ability to appear in **many forms**.
- In other words, **single action** in different ways.
- Achieved using Method Overriding & Method Overloading

Text Books & References

TEXTBOOKS

- Herbert Schildt., “Java: The Complete Reference,” 12th Edition, McGraw Hill Education, New Delhi, 2019.
- Cay S.Horstmann., “Core Java Fundamentals,” Volume 1, 11th Edition, Prentice Hall, 2018.

REFERENCES

- Deitel P and Deitel H, "Java: How to Program", 11th Edition, Prentice Hall, 2018.
- James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley and Daniel Smith, "The Java Language
- Specification – Java SE", 13th Edition, Oracle America Inc., USA, 2019.
- Matt Weisfeld, "The Object-Oriented Thought Process", 5th Edition, Addison-Wesley Professional, US, 2019
- Daniel Liang L, "Introduction to Java Programming", 10th Edition, Pearson Education, New Delhi, 2015.


Thank You!