SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Unit I – INTRODUCTION TO OOP AND JAVA

Topic : Overview of JAVA

Topics to be covered

- Overview of Java
- Three Principles of Java
- Java Program Structure

Overview of Java

- Java is a high-level, object-oriented programming language developed by **James Gosling** at **Sun Microsystems** (now owned by Oracle) in **1995**.
- It is designed to be **platform-independent**, secure, and reliable, making it one of the most widely used programming languages in the world.

Key Characteristics of Java

1. **Simple** – Java has a clean and easy-to-understand syntax, similar to C++.
2. **Object-Oriented** – Everything in Java revolves around objects and classes.
3. **Platform-Independent** – Java follows the **"Write Once, Run Anywhere" (WORA)** principle using the **Java Virtual Machine (JVM)**.
4. **Secure** – Java includes security features like bytecode verification, sandboxing, and exception handling.

Cont…

Key Characteristics of Java

5. **Robust** – Java provides strong memory management (Garbage Collection) and exception handling.
6. **Multithreaded** – Java supports concurrent execution of multiple tasks.
7. **High Performance** – Java uses Just-In-Time (JIT) compilation for faster execution.
8. **Distributed** – Java supports networking and remote method invocation (RMI) for distributed applications.
9. **Dynamic** – Java supports dynamic class loading and runtime binding.

Java Architecture

- **Java Development Kit (JDK)** – Includes the compiler, libraries, and tools to develop Java applications.
- **Java Runtime Environment (JRE)** – Provides libraries and JVM to run Java applications.
- **Java Virtual Machine (JVM)** – Converts Java bytecode into machine code for execution.

Java Applications

- **Web Development** (Spring, Hibernate, Servlets)
- **Mobile Applications** (Android Development)
- **Enterprise Applications** (Banking, E-commerce)
- **Game Development** (LibGDX, jMonkeyEngine)
- **Cloud Computing & Big Data** (Hadoop, Apache Spark)

Overview of Java

Two Paradigms : All computer programs consist of two elements: code and data. Furthermore, a program can be conceptually organized around its code or around its data. These are the two paradigms that govern how a program is constructed. The **first** way is called the **process-oriented model**. This approach characterizes a program as a series of linear steps (**that is, code**). The process-oriented model can be thought of as **code acting on data**. **Procedural languages** such as C **employ this model** to considerable success.

Overview of Java

To manage increasing complexity, the **second** approach, called **object oriented programming**, was conceived. Object-oriented programming organizes a program around its data (**that is, objects**) and a set of well-defined interfaces to that data. An object-oriented program can be characterized as **data controlling access to code**.

Overview of Java

Abstraction: An essential element of object-oriented programming is abstraction. A powerful way to manage abstraction is through the use of hierarchical classifications. The data from a traditional process-oriented program can be transformed by abstraction into its component objects.

The Three OOP Principles

Three OOP principles are encapsulation, inheritance, and polymorphism.

Encapsulation : Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse. One way to think about encapsulation is as a protective wrapper that prevents the code and data from being arbitrarily accessed by other code defined outside the wrapper.

Access to the code and data inside the wrapper is tightly controlled through a well-defined interface.

The Three OOP Principles

Encapsulation

In java a class defines the structure and behavior (data and code) that will be shared by a set of objects. Each object of a given class contains the structure and behavior defined by the class, as if it were stamped out by a mold in the shape of the class. For this reason, **objects are sometimes referred to as instances of a class**. Thus, a **class is a logical construct; an object has physical reality.**

The Three OOP Principles

Encapsulation

When you create a class, you will specify the code and data that constitute that class. Collectively, these **elements are called members of the class**. Specifically, the **data** defined by the class are referred to as **member variables or instance variables**. The **code** that operates on that data is referred to as **member methods or just methods**. In Java programs, the methods define how the member variables can be used. This means that the behavior and interface of a class are defined by the methods that operate on its instance data.

The Three OOP Principles

Encapsulation

When you create a class, you will specify the code and data that constitute that class. Collectively, these **elements are called members of the class**. Specifically, the **data** defined by the class are referred to as **member variables or instance variables**. The **code** that operates on that data is referred to as **member methods or just methods**. In Java programs, the methods define how the member variables can be used. This means that the behavior and interface of a class are defined by the methods that operate on its instance data.

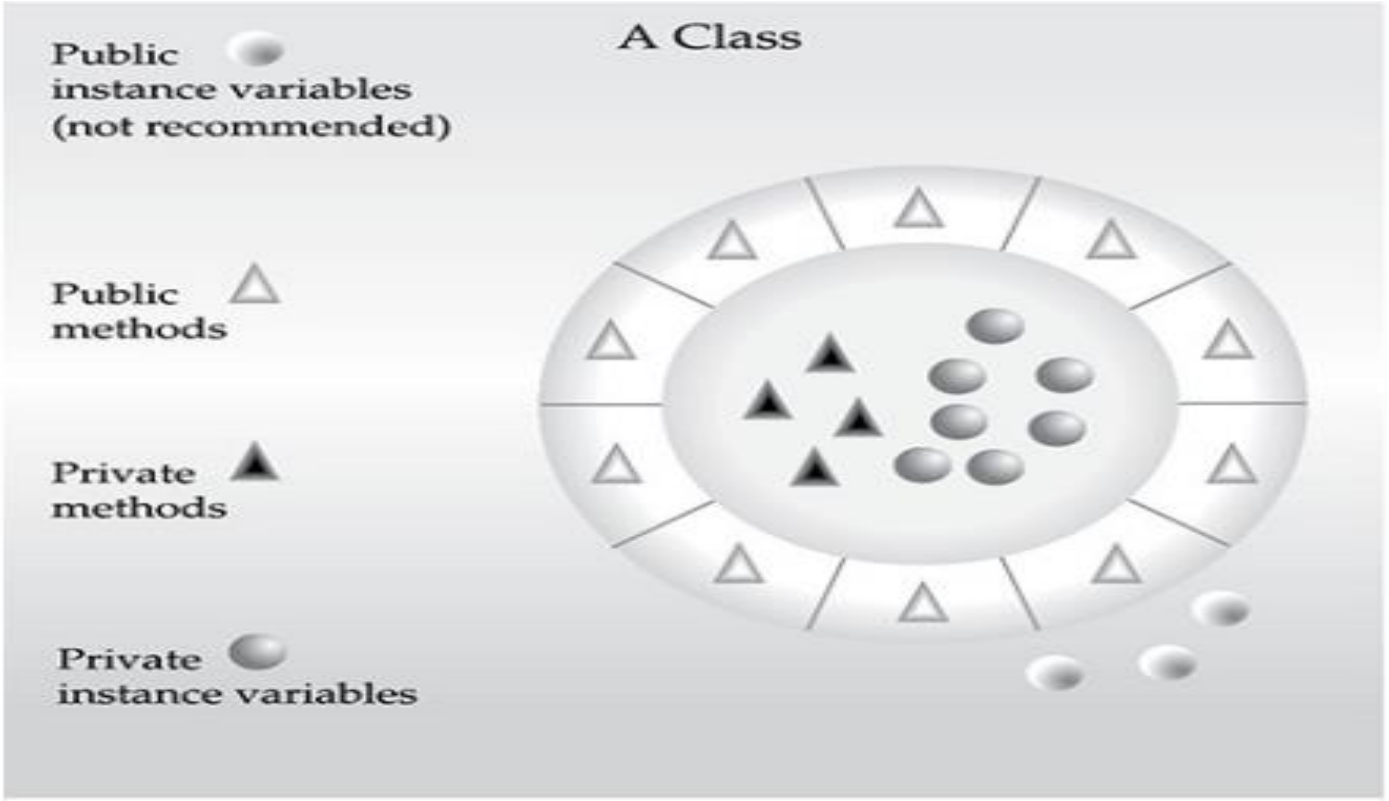
The Three OOP Principles

Encapsulation

Each method or variable in a class may be marked **private or public**. The **public** interface of a class represents **everything that external** users of the class need to know, or may know. The **private** methods and data can only be **accessed by code that is a member of the class**. Therefore, any other code that is not a member of the class cannot access a private method or variable. Since the private members of a class may only be accessed by other parts of your program through the class'public methods, one can ensure that no improper actions take place.

The Three OOP Principles

Encapsulation



Encapsulation: public methods can be used to protect private data.

The Three OOP Principles

Inheritance

Inheritance is the process by which one object acquires the properties of another object. This is important because it supports the concept of hierarchical classification.

Example:

If wanted to describe a more specific **class** of **animals**, such as **mammals**, they would **have** more **specific attributes**, such as type of teeth and mammary glands. This is known as a **subclass** **of animals**, where animals are referred to as **mammals'superclass**.

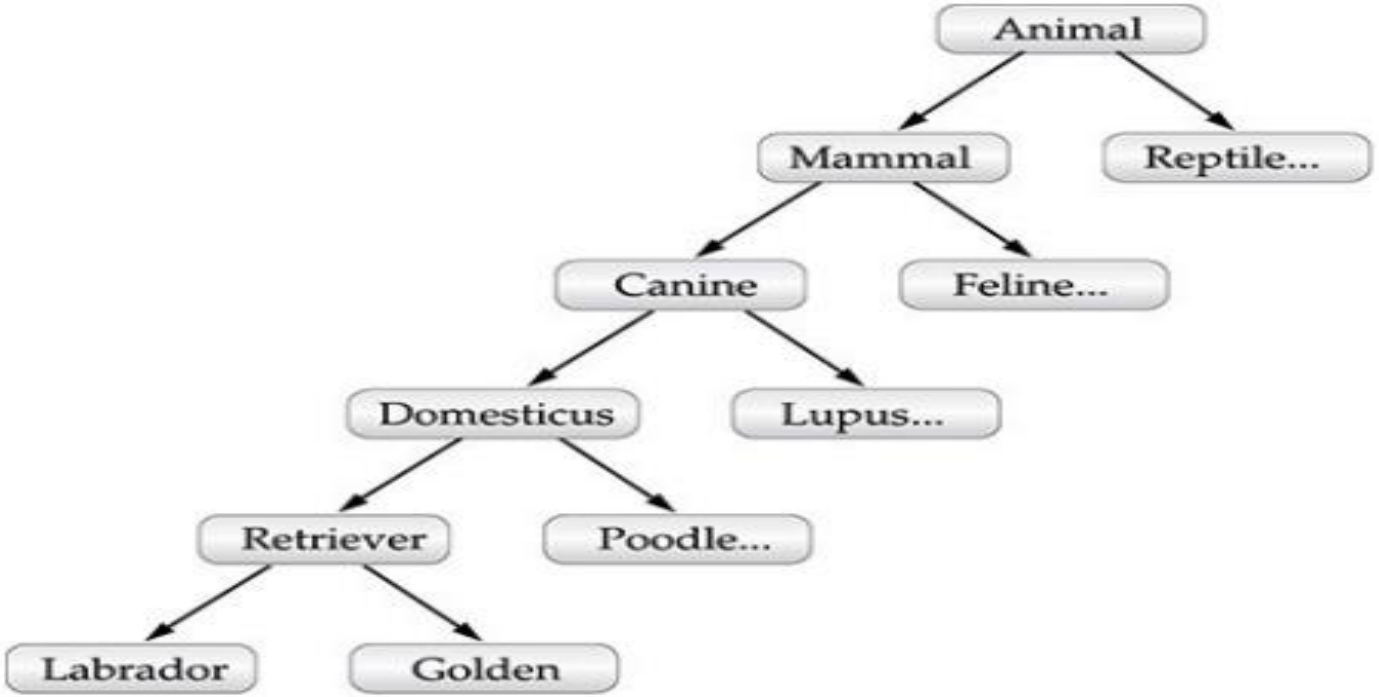
The Three OOP Principles

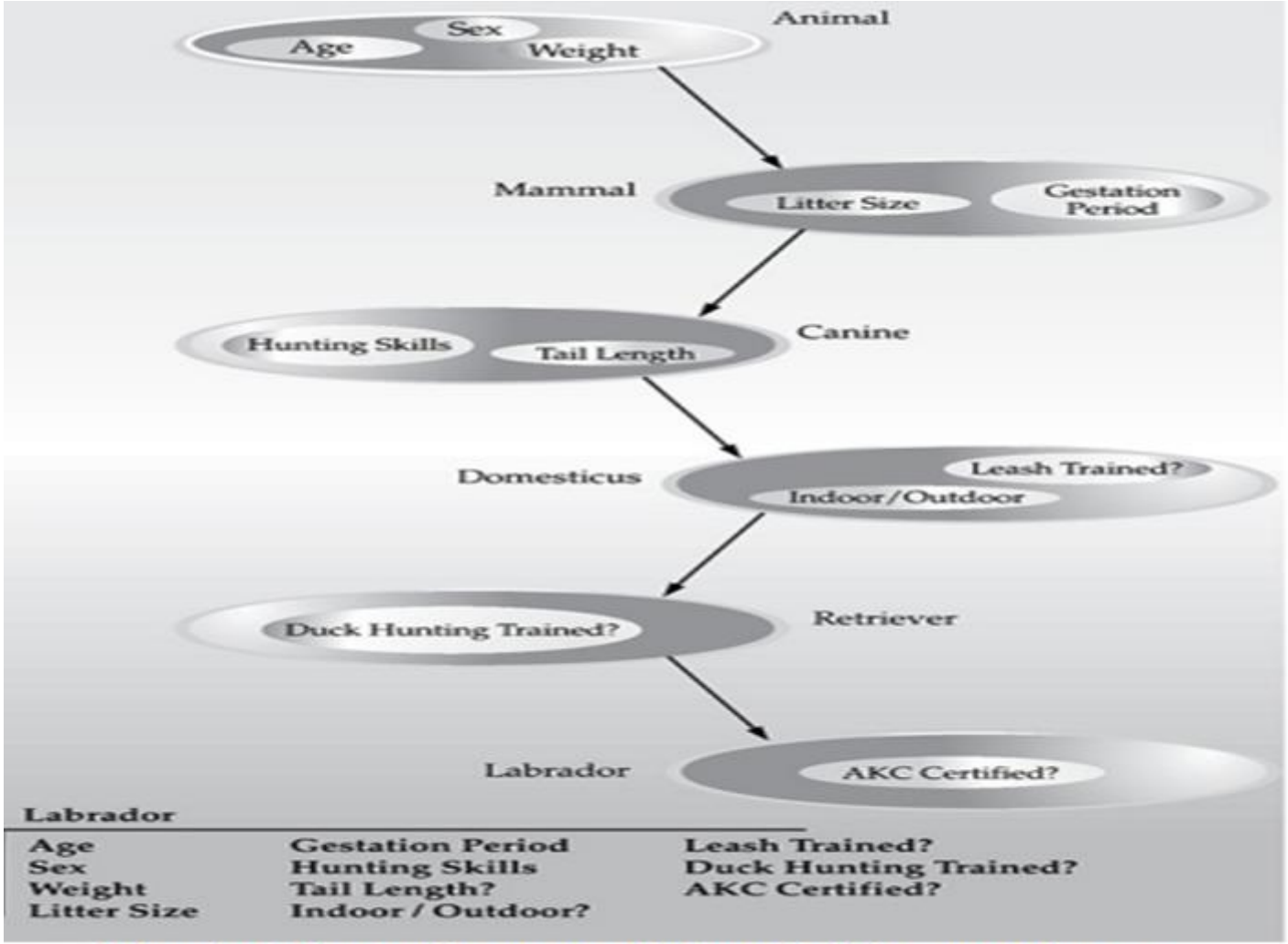
Inheritance

Since mammals are simply more precisely specified animals, they inherit all of the attributes from animals. A **deeply inherited subclass inherits all of the attributes from each of its ancestors** in the class hierarchy. **Inheritance interacts with encapsulation as well**. If a given class encapsulates some attributes, then any subclass will have the same attributes plus any that it adds as part of its specialization. **This is a key concept that lets object-oriented programs grow in complexity linearly rather than geometrically**. A new subclass inherits all of the attributes of all of its ancestors.

The Three OOP Principles

Inheritance





Labrador inherits the encapsulation of all its superclasses.

The Three OOP Principles

Polymorphism

Polymorphism (from Greek, meaning “many forms”) is a feature that allows **one interface to be used for a general class of actions**. The specific action is determined by the exact nature of the situation.

Example:

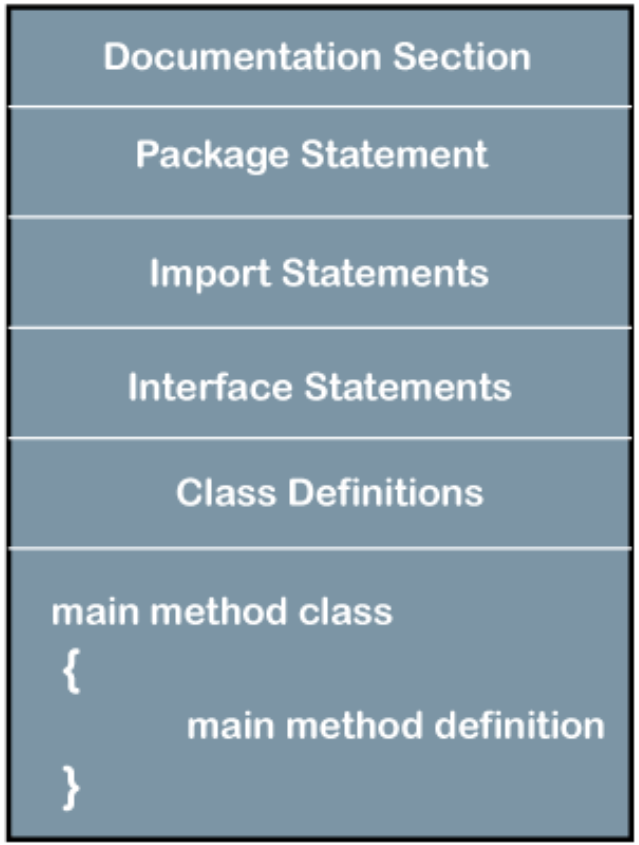
Consider a stack (which is a last-in, first-out list). One need a program that requires three types of stacks. One stack is used for integer values, one for floating-point values, and one for characters. The algorithm that implements each stack is the same, even though the data being stored differs.

The Three OOP Principles

Polymorphism

In a non–object-oriented language, you would be required to create three different sets of stack routines, with each set using different names. However, because of polymorphism, in Java you can specify a general set of stack routines that all share the same names. the concept of polymorphism is often expressed by the phrase "**one interface, multiple methods**." This means that it is possible to design a generic interface to a group of related activities. This helps reduce complexity by allowing the same interface to be used to specify a general class of action. It is the compiler's job to select the specific action

Basic structure of Java program



Structure of Java Program

Documentation section

- **Optional** to improve the readability of the program
- Use **comments** to include **basic information** such as
  - **author's name**
  - **date of creation**
  - **Version**
  - **program name**
  - **company name**
  - **description**
- The comments may be **single-line (//)**

  multi-line (/* … */)

  documentation (/… */**

Package Declaration

- **Optional.** Placed just after the documentation section
- Declare the **package name** in which the class is placed
- There can be **only one package** statement in a Java program. It must be **defined before any class and interface** declaration.
- Use the keyword **package** to declare the package name.

package sample; //where **sample** is the package name

Import Statements

- The package contains the many predefined classes and interfaces.
- To use any class of a particular package, we need to import that class.
- The import statement represents the class stored in the other package.
- It is written **before the class declaration and after the package** statement.

import java.util.Scanner; //it imports the Scanner class only

import java.util.*; //it imports all the class of the java.util package

- Use the **import** keyword to import the class.
- import statement is used to either import a **specific class** or import **all classes of a particular package**.
- Can use multiple import statements.

Interface Section

- Optional. Use the **interface** keyword to create an interface.
- An interface is a slightly different from the class.
  - It contains only **constants** and **method** declarations.
  - It cannot be **instantiated**. i.e. cannot create instance of an class
- Use interface in classes by using the **implements** keyword.
- An interface can also be used with other interfaces by using the **extends** keyword.

```
interface car
{
void start();
void stop();
}
```

Interface Section

```
interface One
{
    public void methodOne();
}


// Implementing the interface
class Three implements One {
    public void methodOne()
    {

        // Implementation of the method
    }
```

```
interface One
{
    void methodOne();
}


// Interface extending defined
interface
interface Three extends One
 {
}
```

Class Definition

- It is **vital** part of a Java program.
- Use the **class** keyword to define the class. The class is a blueprint of a Java program. It contains information about user-defined methods, variables, and constants.
- Without the class, one cannot create any Java program.
- Every Java program has at least one class that contains the main() method.

```
class Student //class definition
{
}
```

Main Method Class

- The main() method is essential for all Java programs as the execution of all Java programs starts from the main() method. i.e. it is an entry point of the class.
- It must be inside the class. Inside the main method, user create objects and call the methods.

```
public class Student //class definition
{
public static void main(String args[])
{
//statements
}
}
```

Methods and behavior

- Defines the functionality of the program.
- The methods are the set of instructions that user want to perform.
- These instructions execute at runtime and perform the specified task.

```
public class Demo //class definition
{
public static void main(String args[])
{
void display()
{
System.out.println("Welcome to javatpoint");
}
//statements
}
}
```

Summary

The three OOP principles **Polymorphism, Encapsulation, and Inheritance** Work Together When properly applied, polymorphism, encapsulation, and inheritance combine to produce a programming environment that supports the development of far more robust and scalable programs than does the process-oriented model.


Thank
you