**Packages in java**

Packages:

**Packages are used to organize the java files or structure the java file.**

Example:

In the forming the plants are planted in the organized manner because they have different food or water or nutrient requirement to handle the scenario they are planted in the same row

* Similarly , for example in the refrigerator in the home fruits are packed in the plastic container to organize
* Similarly, In shops organized the product to customer convenient

**Why are they doing this ?**

* Because we make our life easy.
* To try to group the based on the business logic
* Use the same named java file in the project (To avoid conflict in same name)

**How to Structure the code in java ?**

**Module**

**Package 1 Package2**

**Classes Classes**

**Interfaces Interfaces**

**Enums Enums**

**Annotations Annotations**

**Modules were introduced in JDK 9.** Prior to JDK 9, packages and types existed, but modules were not part of the structure. Modules serve as an optional means of organization code, complementing the existing packages.

A Java project consists of one or more modules, each housing zero or more packages. Within a package , there exist one or more types, with “type” being a general term encompassing user-defined data type. The four main user defined data types are Classes, interfaces, inheritance,enum , annotations .

Understanding packages in Java is mandatory since every real project groups their classes , interfaces etc.. into various packages for various benefits.

**Package in Java**

In Java, packages are used to organize classes into namespaces . A package is a collection of related classes, interfaces, annotations and enum grouped together in a single unit .

**Advantages:**

* Organize large codebases into logical units, making it easier to manage and maintain code.
* Way to encapsulate classes and interfaces, hiding implementation details from other parts of the code.
* Control access to classes and interfaces , using access modifiers like **public, private , protected and default.**
* Way to avoid naming conflict between classes and interfaces, by giving them unique names within a package.

**Note :**

**If you do not use a package statement , your type ends up in an unnamed or default package maintained by the JVM behind.** Generally speaking an unnamed or default package is only for small or temporary applications or when you are just beginning the development process. Otherwise , classes, interfaces, annotations and enums belong in named packages.

Creating a package:

To attach a class to a package, you can use the **package keyword** followed by the package name , at the top of your source code file. For example :

**package com.example.myapp;**

**public class Myclass{**

**// class code goes here**

**}**

Package must be the first line in the source file . There can be only one package statement in each source file,and it applies to all type in the file

**Approach 1 :**

Package com.lucky( in here we have to mention the organization name).businesslogic

So we can create N number of packages based on the requirement.

**Approach 2 :**

In java, there are various architectural patterns that involve dividing the application into different layers based on function and responsibility. Some common layers used in java application are:

**Model layer(POJO classes Plain Old Java Object)-:**

The model layer is responsible for representing the data of an application.

**Service layer:**

The service layer is responsible for implementing the **business logic and processing data received from the model layer.** It typically contains classes that implement service interfaces and perform data validation, transformation, and other business logic.

**Utility layer:**

The utility layer provides common functionality that can be **reused across the application**. **It typically contains helper classes, constants, and utility methods**.

**Controller / view layer :**

The controller/ view layer is responsible for handling user input and presenting data to the user . **It typically contains classes that handle HTTP requests, map urls to service methods, and render views for display**.

**Presentation layer(contain html pages so on ):**

The presentation layer is responsible for presenting data to the user in a format that can be easily understood . It typically contains the classes that implement user interfaces such as web pages or mobile app screens.

These layers help to separate concerns and maintain code modularity, making it easier to maintain and extend the application over time