Experiment No. 5

Implement a program for creation of user defined packages and its use.

Instructions:

This manual consists of three parts:
A) Theory and Concepts,
B) Problems for Implementation, and
C) Write-up Questions.

1. Students must understand the **theory and concepts** provided before implementing the problem statement(s) for **Experiment 5**.
2. They should **practice the given code snippets** within the theory section.
3. Later, they need to **implement the problems provided**.
4. **Write-up:** Students are required to **write answers** to the questions on journal pages, **maintain a file**, and get it checked regularly. The file should include index, write-up, and implementation code with results.
5. **Referencing**: Include proper sources or references for the content used.
6. Use of Generative AI: Clearly mention if you have used any AI tools (e.g., ChatGPT, Copilot, Gemini) to generate text, explanations, or code. Cite the AI-generated content appropriately in the write-up.

Part A. Theory and Concepts:

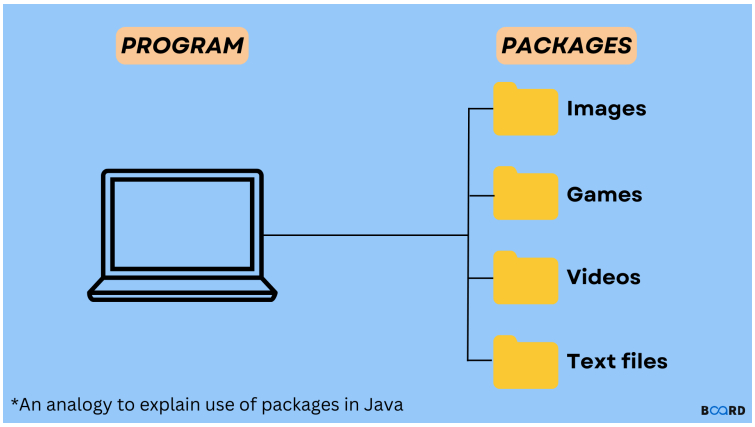
Packages in Java are a mechanism that encapsulates a group of classes, sub-packages, and interfaces. Packages are used for:

- **Prevent naming conflicts** by allowing classes with the same name to exist in different packages, like -
    - college.staff.cse.**Employee**
    - college.staff.mech.**Employee**.
- They make it easier to organize, locate, and use classes, interfaces, and other components.
- Packages also provide controlled access for Protected members that are accessible within the same package and by subclasses.
- Also for default members (no access specifier) that are accessible only within the same package.

Example: Imagine you have different types of files on your computer. To keep things organized, you create separate folders:

- A **Projects** folder for various work or academic projects.
- A **Songs** folder for storing music files.
- A **Movies** folder for keeping different movies.

This helps in avoiding clutter, making it easier to find files, and ensuring better management of content. Similarly, in Java, we use **packages to organize related classes,** just like folders in a computer.



1) Types of Packages in Java:

1. **Built-in Packages** - Provided by Java (e.g., java.util, java.io, java.lang).
2. **User-Defined Packages** - Created by the user to organize and manage their own classes.

Predefined Packages in Java (Built-in Packages) - Java APIs contains the following predefined packages, as shown in the below figure:

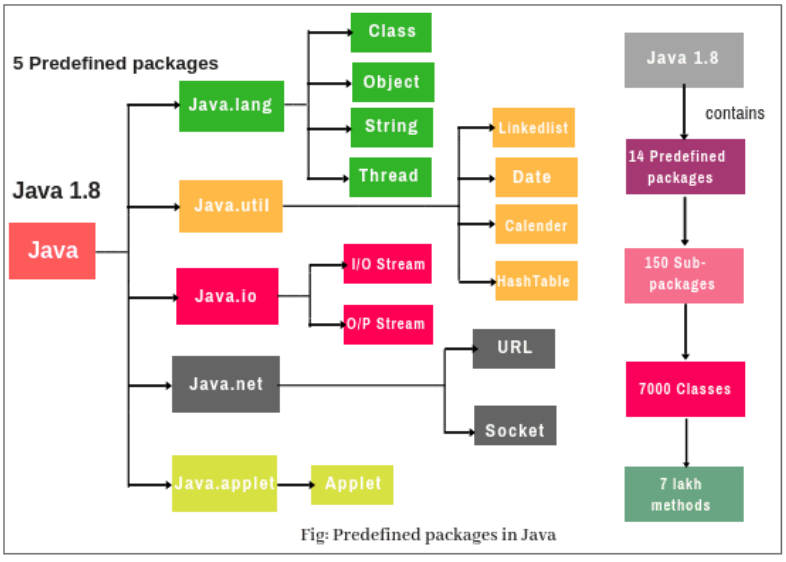


Fig: Predefined packages in Java

Predefined packages in Java are those which are developed by the Sun Microsystem. They are also called built-in packages. These packages consist of a large number of predefined classes, interfaces, and methods that are used by the programmer to perform any task in his programs.

2) Working of Java Packages

- **Directory Structure**

  Package names and directory structures are closely related. For example, if a package name is **company.department.team**, then the directory structure will be:

  - **company/** (Top-level folder)
    - **department/** (Inside company)
      - **team/** (Inside department)
        - Contains Java class files.

  This hierarchical structure helps in better management of classes.

- **Naming Conventions**

  Package names follow a standard convention similar to domain names written in reverse order.

  For example:

  - A software company might use:
    - com.techsolutions.software
    - com.techsolutions.hardware
    - com.techsolutions.research
  - In an e-commerce platform, the convention might be:
    - com.shopify.products
    - com.shopify.orders
    - com.shopify.customers

Naming Convention for User-defined Package in Real Time Project

While developing your project, you must follow some naming conventions regarding packages declaration. Let's take an example to understand the convention.

Look at the below a complete package structure of the project.

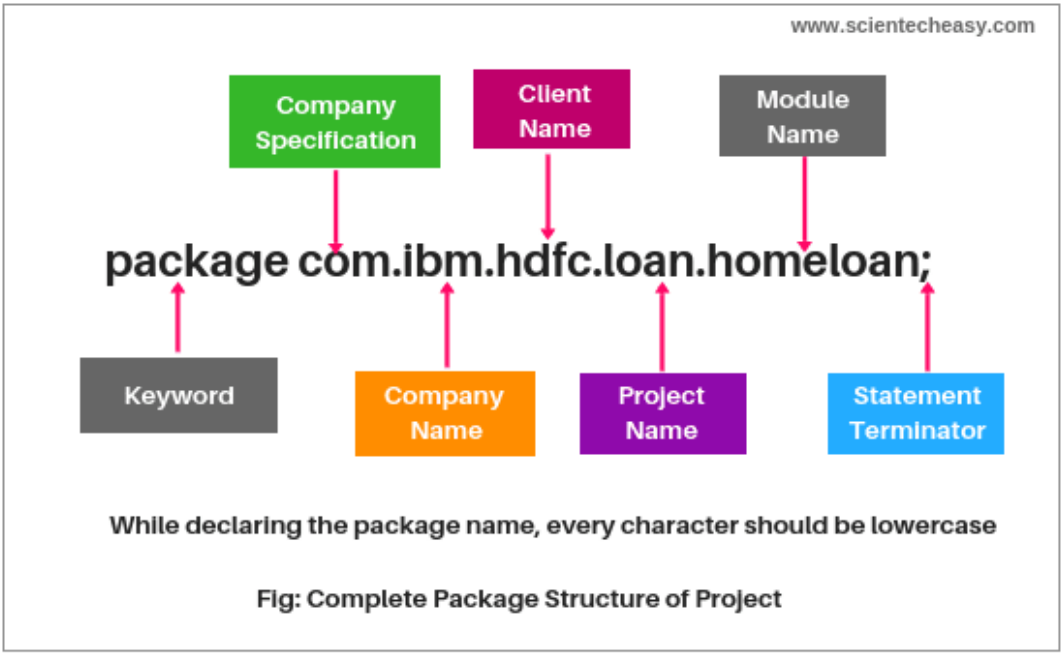


Fig: Complete Package Structure of Project

1. Suppose you are working in IBM and the domain name of IBM is www.ibm.com. You can declare the package by reversing the domain like this: `package com.ibm;`

   where,

   com → It is generally the company specification name, and the folder starts with com, which is called root folder.

   ibm → Company name where the product is developed. It is the sub folder.
2. **hdfc** → Client name for which we are developing our product or working on the project.
3. **loan** → Name of the project.
4. **homeloan** → It is the name of the modules of the loan project. There are a number of modules in the loan project like a home loan, car loan, or personal loan. Suppose you are working for the Home loan module.

This is a complete packages structure, like a professional which is adopted in the company. Look at another example below:

```
package com.tcs.icici.loan.carloan.penalty;
```

Note: *Keep in mind the Root folder should be always the same for all the classes.*

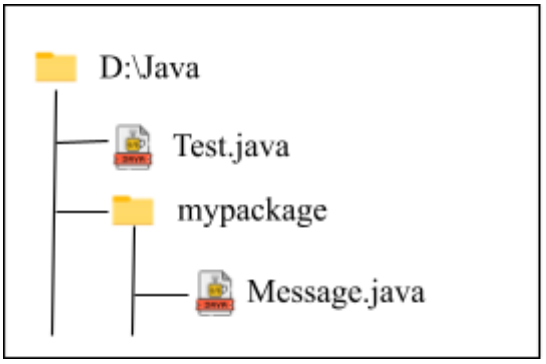
Creating a User-Defined Package:

To create a user-defined package:

1. Use the **package** keyword at the beginning of the Java file.
2. Save the file in the appropriate directory.
3. Compile the file using **javac -d . ClassName.java**

4. Use **import** to access the package in other Java programs.

The folder setup we will use is:



Example:

Step 1: Creating the Package

File: D:\Java\mypackage\Message.java

Create a Message.java file inside the mypackage folder

```
// Step 1: Define the package
package mypackage;

public class Message {
    public void display() {
        System.out.println("Hello from the mypackage!");
    }
}
```

Step 2: Compile the package

```
Open the terminal or command prompt, and navigate to the project folder
```

```
Command
javac -d . mypackage/Message.java
```

```
Output:
 Java creates a mypackage/Message.class file.
```

Step 3: Using the Package in Another Program (Import the package)

File: D:\Java\Test.java

Create this file in the same location as the **mypackage folder,** so that **Test.java** and the **mypackage** folder are in the **same directory.**

```
// Step 3: Import the package

import mypackage.Message;

public class Test {
    public static void main(String[] args) {
        Message obj = new Message(); // Creating an object of the
class
        obj.display(); // Calling the method
    }
}
```

```
//Commands

javac Test.java
java Test
```

```
Output:
Hello from the mypackage!
```

3) Advantages of Packages

- **Encapsulation**: Groups related classes together.
- **Avoids Name Conflicts**: Classes with the same name can exist in different packages.
- **Code Reusability**: Easily import and use the package across multiple programs.
- **Access Control**: Packages can be used to restrict access to certain classes.

Additional Learning Resources:

1. **Video tutorial:** https://www.youtube.com/watch?v=Bua6LQO2vQ8
   (Note: Try implementing the example discussed in the tutorial)
2. **Readings:** https://www.scientecheasy.com/2020/06/packages-in-java.html/
3. **Readings:** https://www.geeksforgeeks.org/packages-in-java/

Part B. Problems for Implementation:

Aim: Implementation of user defined package.

1. Implement a package **LibraryManagement** with classes **Book** and **Member**. The Book class should have attributes like **title, author, and ISBN**, while the Member class should store member details. Use this package to create a simple library system.

2. Create a package **Ecommerce** containing classes **Product, Customer, and Order.** Implement methods for placing an order, displaying product details, and calculating total order cost. Use this package in another program.

3. Create a package named **MathOperations** that contains classes for mathematical functions like **floor**, **round**, and **ceil**. Implement a program that uses these functions to perform operations on different numbers. (The **Math** class in Java contains the methods **floor()**, **ceil()**, and **round())**

Part C. Write-up Questions:

1. What is a package in Java? Explain its types.
2. How do you create and import user-defined packages in Java?
3. What are the advantages of using packages?
4. What is the difference between

   `import packageName.*` and `import packageName.ClassName`?

Conclusion:

By the end of this experiment, students should be able to create and use their own Java packages, understand their role in organizing code, and control access to different classes effectively.