# Lab Sheet: Understanding Packages and Imports in Java

## Objectives

- Understand the concept of packages in Java
- Learn how to create and use packages
- Learn how to import classes from packages

## Prerequisites

- Basic knowledge of Java syntax
- Understanding of classes and objects in Java

## Materials

- Java Development Kit (JDK)
- Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans

## Instructions

### 1. Introduction to Packages

Packages in Java are used to group related classes together. They provide a way to organize files within a project and avoid name conflicts.

A package can be considered similar to a directory or folder in a file system.

## 2. Creating Packages

### Step 1: Create a New Java Project

1. Open your IDE and create a new Java project.
2. Name the project `PackageExample`.

### Step 2: Create a Package

1. Inside the `src` folder of your project, create a new package.
2. Name the package `com.example.utility`.

### Step 3: Create a Class in the Package

1. Right-click on the `com.example.utility` package and create a new Java class.
2. Name the class `MathUtils`.
3. Add the following code to `MathUtils.java`:

```
package com.example.utility;

public class MathUtils {
    public static int add(int a, int b) {
        return a + b;
    }

    public static int subtract(int a, int b) {
        return a - b;
    }
}
```

## 3. Using Packages and Import Statements

### Step 1: Create a New Package for the Main Class

1. Create another package inside the `src` folder named `com.example.main`.

### Step 2: Create the Main Class

1. Right-click on the `com.example.main` package and create a new Java class.
2. Name the class `MainApp`.
3. Add the following code to `MainApp.java`:

```
package com.example.main;


import com.example.utility.MathUtils;

public class MainApp {
    public static void main(String[] args) {
        int sum = MathUtils.add(5, 3);
        int difference = MathUtils.subtract(10, 7);

        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
    }
}
```

## 4. Compiling and Running the Program

**Step 1: Compile the Program**

1. Use your IDE to build the project. This will compile all the classes.

**Step 2: Run the Program**

1. Run the `MainApp` class.
2. You should see the following output:

```
Sum: 8
Difference: 3
```

## 5. Exercises

1. **Exercise 1**: Create a new package named `com.example.shapes` and add a class `Rectangle` with methods to calculate area and perimeter. Use this class in the `MainApp` class.
2. **Exercise 2**: Create a package `com.example.strings` and add a class `StringUtils` with methods to manipulate strings (e.g., reverse a string, convert to uppercase). Use these methods in the `MainApp` class.

## Exercise 6: Geometry Package

**Objective**: Create a package for geometry calculations and use it in the main class.

1. **Step 1**: Create a new package named `com.example.geometry`.
2. **Step 2**: Inside the `com.example.geometry` package, create a class named `Circle`.
3. **Step 3**: Add methods to calculate the area and circumference of a circle.

```
package com.example.geometry;

public class Circle {
    private static double PI = 3.14159;

    public static double calculateArea(double radius) {
        return PI * radius * radius;
    }

    public static double calculateCircumference(double radius) {
        return 2 * PI * radius;
    }
}
```

4. **Step 4**: In the `com.example.main` package, modify the `MainApp` class to use the `Circle` class.

```
package com.example.main;

import com.example.geometry.Circle;

public class MainApp {
    public static void main(String[] args) {
        double radius = 5.0;

        double area = Circle.calculateArea(radius);
        double circumference = Circle.calculateCircumference(radius);

        System.out.println("Area: " + area);
```

```
        System.out.println("Circumference: " + circumference);
    }
}
```

## 7. Summary

- Packages help organize Java classes and prevent naming conflicts.
- The `import` statement is used to bring classes from one package into another.
- By creating packages, you can create modular, maintainable code.