|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Object Oriented Programming** | **Semester:** | **III** |
| **Date of Performance:** | **13 / 10 / 2023** | **Batch No:** | **A - 3** |
| **Faculty Name:** | **Pragya Gupta** | **Roll No:** | **16014022050** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** | **\_\_\_ / 25** |

|  |  |  |
| --- | --- | --- |
| **Writing Program (07)** | **Performance in lab**  **and Viva (05 + 03)** | **Post lab questions, conclusion and**  **completion (03 + 02 + 05)** |
|  |  |  |

**Experiment No: 9**

**Title: Use of Packages**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Learn the how to make user defined packages. |

|  |
| --- |
| **COs to be achieved:** |
| **CO2:** Understand concepts of Object-Oriented Programming and basic characteristics of Java. |

|  |
| --- |
| **Tools used:** |
| 1. [Packages In Java - GeeksforGeeks](https://www.geeksforgeeks.org/packages-in-java/) 2. [Packages in Java: How to Create and Use Packages in Java? | Edureka](https://www.edureka.co/blog/packages-in-java/) |

|  |
| --- |
| **Theory:** |
| 1. (About Types of packages)   In Java, a package is a namespace that organizes a set of related classes and interfaces. Java packages can be categorized into two types based on their accessibility:   * 1. **Built-in Packages:** These are the packages that are automatically imported and made available for use in any Java program. Some commonly used built-in packages include: * java.lang: This package provides fundamental classes that are necessary for basic functionalities. Classes such as String, Math, Object, etc., are part of this package. * java.util: This package contains utility classes like ArrayList, HashMap, Date, and various other utility classes for data manipulation. * java.io: This package provides classes for input and output operations, such as reading and writing to files or streams. * java.net: This package provides classes for networking operations, such as creating sockets, URLs, and handling network communication. * java.awt and javax.swing: These packages provide classes for creating graphical user interfaces (GUI) in Java.   1. **User-defined Packages:** These are the packages that are created by developers to organize their classes and interfaces in a better way. User-defined packages help in avoiding naming conflicts, control access, and make code more maintainable. To create a user-defined package, you can use the package keyword at the beginning of the Java file, followed by the package name.   For example:  package com.example.myPackage;  Here, com.example.myPackage is an example of a user-defined package name.  Packages in Java not only help in organizing the code but also facilitate access control and encapsulation. They provide a way to group related classes and interfaces in a single unit, making the code more modular and manageable. |

|  |
| --- |
| **Code:** |
| 1. **Create a package named *mySeries* that includes a *Series* class which will perform following operations.**    1. **power**    2. **factorial using recursion**    3. **Sum**   **Use these methods to compute the following series**  **upto n terms, take n from the user.**  package mySeries;  public class Series {      // power      public static double power(double base, int exponent) {          if (exponent == 0) {              return 1;          } else if (exponent > 0) {              return base \* power(base, exponent - 1);          } else {              return 1 / power(base, -exponent);          }      }      // factorial using recursion      public static int factorial(int n) {          if (n == 0) {              return 1;          } else {              return n \* factorial(n - 1);          }      }      // series sum      public static double seriesSum(double x, int n) {          double sum = 1;          for (int i = 1; i <= n; i++) {              sum += (power(n, i) \* power(x, i)) / factorial(i);          }          return sum;      }  }  import java.util.\*;  import mySeries.Series;  public class exp9\_q1  {      public static void main(String[] args)      {          Scanner sc = new Scanner(System.in);          System.out.print("enter the value of x: ");          double x = sc.nextDouble();          System.out.print("enter the value of n: ");          int n = sc.nextInt();          System.out.print("series sum: " + Series.seriesSum(x, n));          sc.close();      }  }   1. **Create a package named *myMath* that includes a *Fraction* class which will perform following operations.**    1. **Sum**    2. **Difference**    3. **Product**    4. **GCD**    5. **LCM**   **Use these methods to compute the sum, difference, product and division of two fractions.**  package myMath;  public class Fraction {      private int numerator;      private int denominator;      public Fraction(int numerator, int denominator) {          this.numerator = numerator;          if (denominator != 0) {              this.denominator = denominator;          } else {              throw new IllegalArgumentException("Denominator cannot be zero.");          }          simplify();      }      private int gcd(int a, int b) {          if (b == 0) {              return a;          }          return gcd(b, a % b);      }      private void simplify() {          int gcd = gcd(Math.abs(numerator), Math.abs(denominator));          numerator /= gcd;          denominator /= gcd;      }      public Fraction add(Fraction other) {          int newNumerator = this.numerator \* other.denominator + other.numerator \* this.denominator;          int newDenominator = this.denominator \* other.denominator;          return new Fraction(newNumerator, newDenominator);      }      public Fraction subtract(Fraction other) {          int newNumerator = this.numerator \* other.denominator - other.numerator \* this.denominator;          int newDenominator = this.denominator \* other.denominator;          return new Fraction(newNumerator, newDenominator);      }      public Fraction multiply(Fraction other) {          int newNumerator = this.numerator \* other.numerator;          int newDenominator = this.denominator \* other.denominator;          return new Fraction(newNumerator, newDenominator);      }      public int gcd(Fraction other) {          return gcd(this.numerator \* other.denominator, other.numerator \* this.denominator);      }      public int lcm(Fraction other) {          return (this.denominator \* other.denominator) / gcd(other);      }      @Override      public String toString() {          return numerator + "/" + denominator;      }  }  import myMath.Fraction;  public class exp9\_q2  {      public static void main(String[] args)      {            Fraction fraction1 = new Fraction(1, 2);          Fraction fraction2 = new Fraction(1, 3);          // sum          Fraction sumResult = fraction1.add(fraction2);          System.out.println("sum: " + sumResult);          // difference          Fraction diffResult = fraction1.subtract(fraction2);          System.out.println("difference: " + diffResult);          // product          Fraction productResult = fraction1.multiply(fraction2);          System.out.println("product: " + productResult);          // GCD          int gcdResult = fraction1.gcd(fraction2);          System.out.println("GCD: " + gcdResult);          // LCM          int lcmResult = fraction1.lcm(fraction2);          System.out.println("LCM: " + lcmResult);      }  } |

|  |
| --- |
| **Output:** |
|  |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. **Which package is always imported by default?**   The java.lang package is always imported by default in Java. This package contains fundamental classes and interfaces that are automatically available to all Java programs. Some of the commonly used classes from the java.lang package include String, Object, System, and Math.   1. **Explain the various access specifiers in JAVA with respect to packages.**   In Java, access specifiers are keywords that define the accessibility or visibility of a class, method, or field. They control how these elements can be accessed from other parts of the program. When it comes to packages, the following access specifiers can be used:   * **Public:** A public class, method, or field is accessible from any other class or package. This means that any other class can access the public members of the class. For example, a public class can be accessed from any other package. * **Protected:** A protected class, method, or field is accessible within its own package and to its subclasses, even if they are in different packages. This means that the protected members can be accessed within the package and by the subclasses in any package. * **Default (Package-private):** If no access specifier is used, then the default access is applied. A class, method, or field with default access is only accessible within its own package. It cannot be accessed from outside the package, even by the subclasses. * **Private:** A private class, method, or field is only accessible within the class in which it is declared. It cannot be accessed from outside the class, not even by the subclasses. |

|  |
| --- |
| **Conclusion:** |
| In this Java experiment, we gained hands-on experience in creating user-defined packages. By organizing related classes and resources into distinct packages, we enhanced code modularity and encapsulation, promoting a more structured and manageable codebase. |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |