DSA Assignment 1

     Q1.Armstrong Number  
      Problem: Write a Java program to check if a given number is an Armstrong . **package** dsa1;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **boolean** isArmstrong(**int** n){

**int** original = n ;

**int** sum = 0 ;

**int** num = String.*valueOf*(n).length();

**while** (n != 0){

**int** digit = n % 10;

sum += Math.*pow*(digit,num);

n/=10;

}

**return** sum == original;

}

**public** **static** **void** main(String[] args) {

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("Enter a number:");

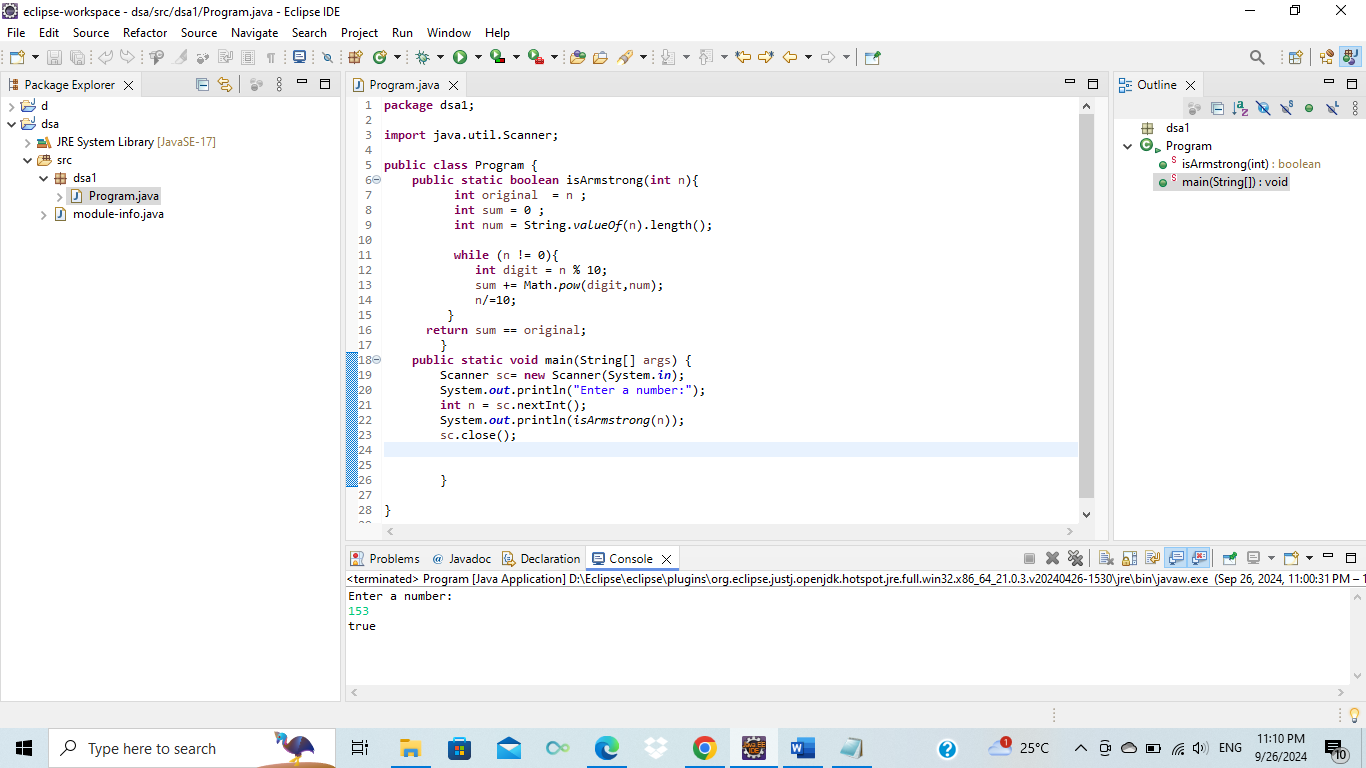
**int** n = sc.nextInt();

System.***out***.println(*isArmstrong*(n));

sc.close();

}

}



**Flowchart:**

      1.Start

      2.Input the number

      3.Store the number in a variable (original)

      4. Count number of digits

      5. Initialize sum to 0

      6. For each digit of given number :

              > Extract the last digit using %.

              > Raise the digit to the power of no of digits

              > Add result to (sum)

              > Remove the last digit from the number

     7.If sum is equal to original,return TRUE otherwise return FALSE

     8.End

Time Complexity :

    >> O(d),where d is no of digits in the input number.

         The program extracts each digit,raises it to the power of d , and sums the result.

Space Complexity :

   >> O(1) : only few integer variables are used to store the result

Q2 Prime Number

Problem: Write a Java program to check if a given number is prime.

package org.example1;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number:");

int n=sc.nextInt();

boolean flag= false;

if(n==0 || n==1) {

flag=true;

}

for (int i = 2; i <= n / 2; ++i) {

if (n % i == 0) {

flag = true;

break;

}

}

if (!flag)

System.out.println(n + " is a prime number.");

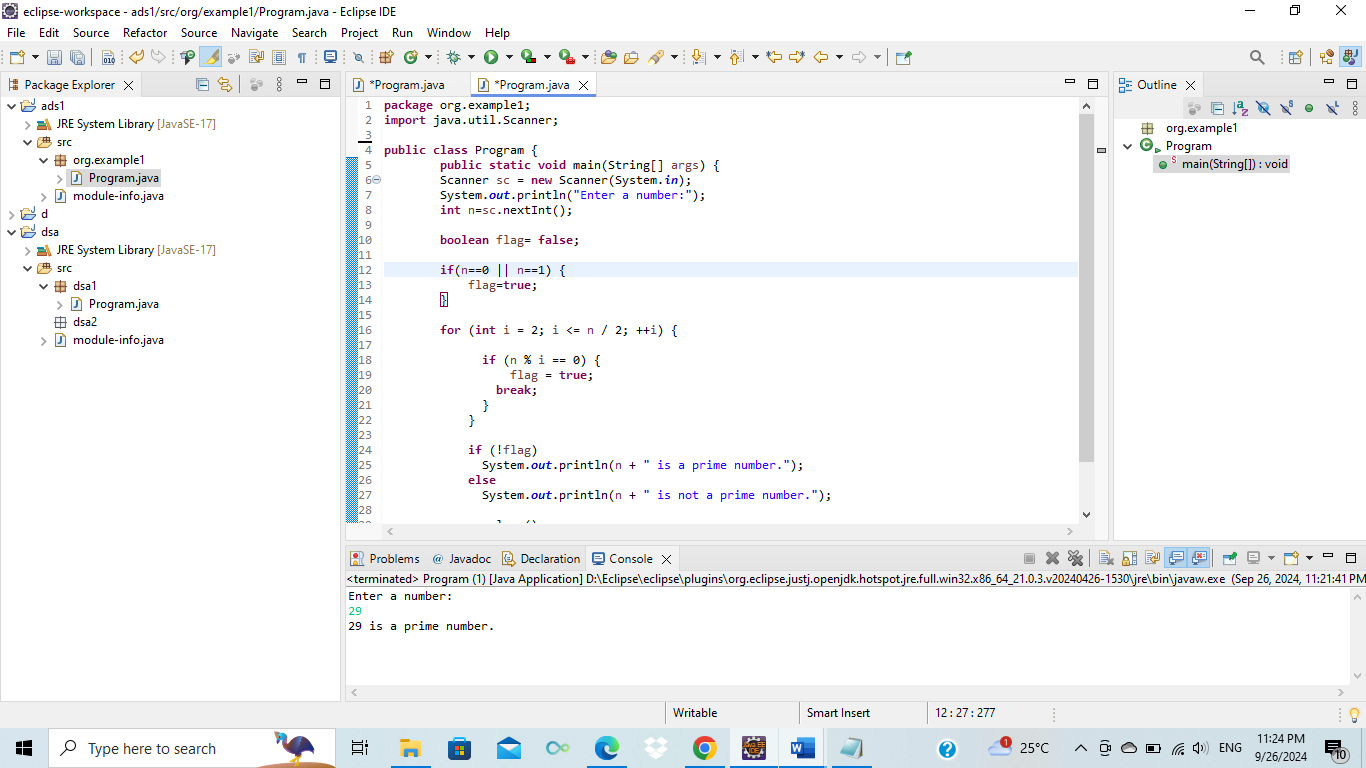
else

System.out.println(n + " is not a prime number.");

sc.close();

}

}



Flowchart

1. Start
2. Input  number n from user
3. Initialize flag to false
4. Check if n is 0 or 1
5. For loop from i=2 to n/2

Check if (n%i==0)

* If yes set flag= true and break the loop
* If no continue the loop

1. Chek if flag is false

If true, print n is a prime number

If false, print  n is not a prime number

1. end

Q3/// 3 .Problem: Write a Java program to compute the factorial of a given number.

//

//Test Cases:

//

//Input: 5

//Output: 120

//Input: 0

//Output: 1

**package** org.example1;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **int** factorial1(**int** a) {

**int** res = 1;

**if**(a==0 || a==1)

**return** res;

**while**(a>1) {

res=res\*a;

a--;

}

**return** res;

}

**public** **static** **int** factorial(**int** a) {

**if**(a==0 || a==1)

**return** 1;

**return** a=a\**factorial*(a-1);

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter a number:");

**int** b= sc.nextInt();

System.***out***.println(*factorial*(b));

System.***out***.println(*factorial1*(b));

sc.close();

}

}

**Flowchart:**

1.Start

2. Input the number.

3. call factorial method

Initialize res = 1.

If b is 0 or 1, return res as the factorial.

While (b >1)

res \* b

b--

Call recursive method Factoral

If b is 0 or 1, return 1 as the factorial.

return b \* facorial(b-1) (recursive call).

end

Q4  Fibonacci Series

Problem: Write a Java program to print the first n numbers in the Fibonacci series.

Program

**package** org.example2;

**import** java.util.Scanner;

**public** **class** Program { // Fibonacci series

**public** **static** **void** main(String[] args) {

System.***out***.println("Enter no: ");

Scanner sc = **new** Scanner(System.***in***);

**int** n = sc.nextInt();

**int** a = 0, b = 1;

**for** (**int** i = 0; i < n; i++) {

System.***out***.print(a + " ");

**int** next = a + b;

a = b;

b = next;

}

}

}

Flowchart

1. Start
2. Input a number.
3. Get Input using scanner

4. Set Initial Values: a = 0 and b = 1.

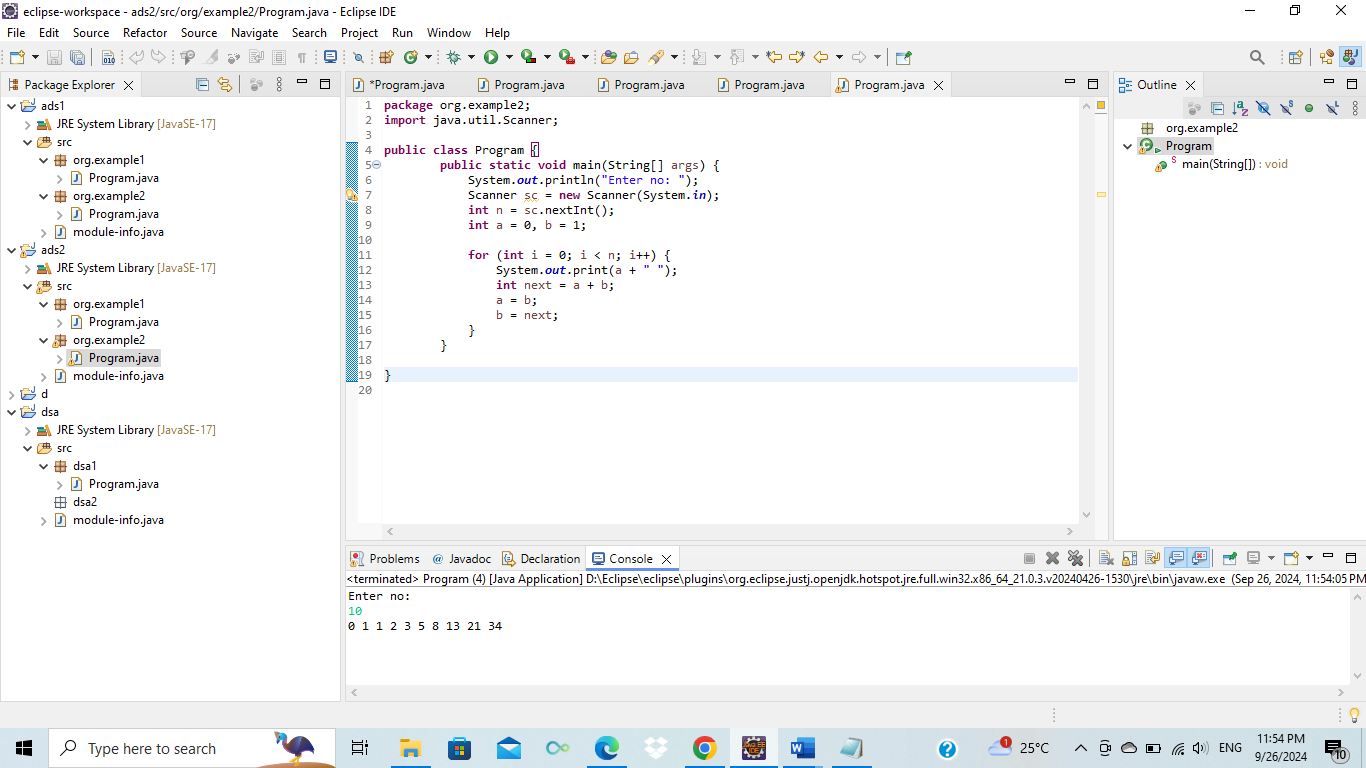
5. for Loop (i=0; i<n; i++)

6. Print the Number:

7. Calculate Next Number: a + b.

8. when (i<n) loop end

9. End:



Q5. Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.  
  
**package** org.example3;

**import** java.util.Scanner;

**public** **class** Program {//gcd

**public** **static** **int** gcd(**int** a, **int** b) {

**if** (b == 0) **return** a;

**return** *gcd*(b, a % b);

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**try** {

System.***out***.println("Enter two numbers:");

**int** a = sc.nextInt();

**int** b = sc.nextInt();

System.***out***.println("GCD: " + *gcd*(a, b));

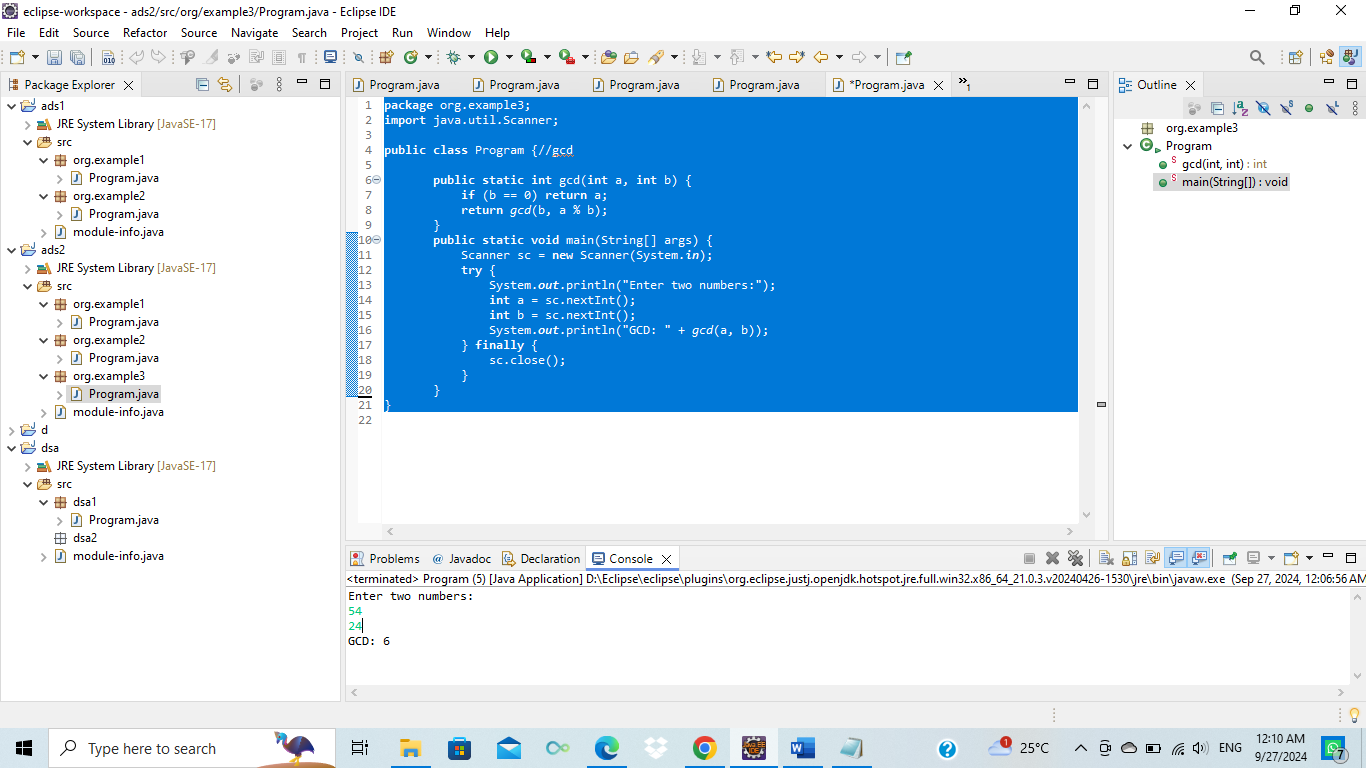
} **finally** {

sc.close();

}

}

}



Q10. Leap Year

Problem: Write a Java program to check if a given year is a leap year.

**package** org.example4;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **boolean** isLeapYear(**int** year) {

**if** (year % 4 == 0) {

**if** (year % 100 == 0) {

**return** year % 400 == 0;

} **else** {

**return** **true**;

}

}

**return** **false**;

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter a year:");

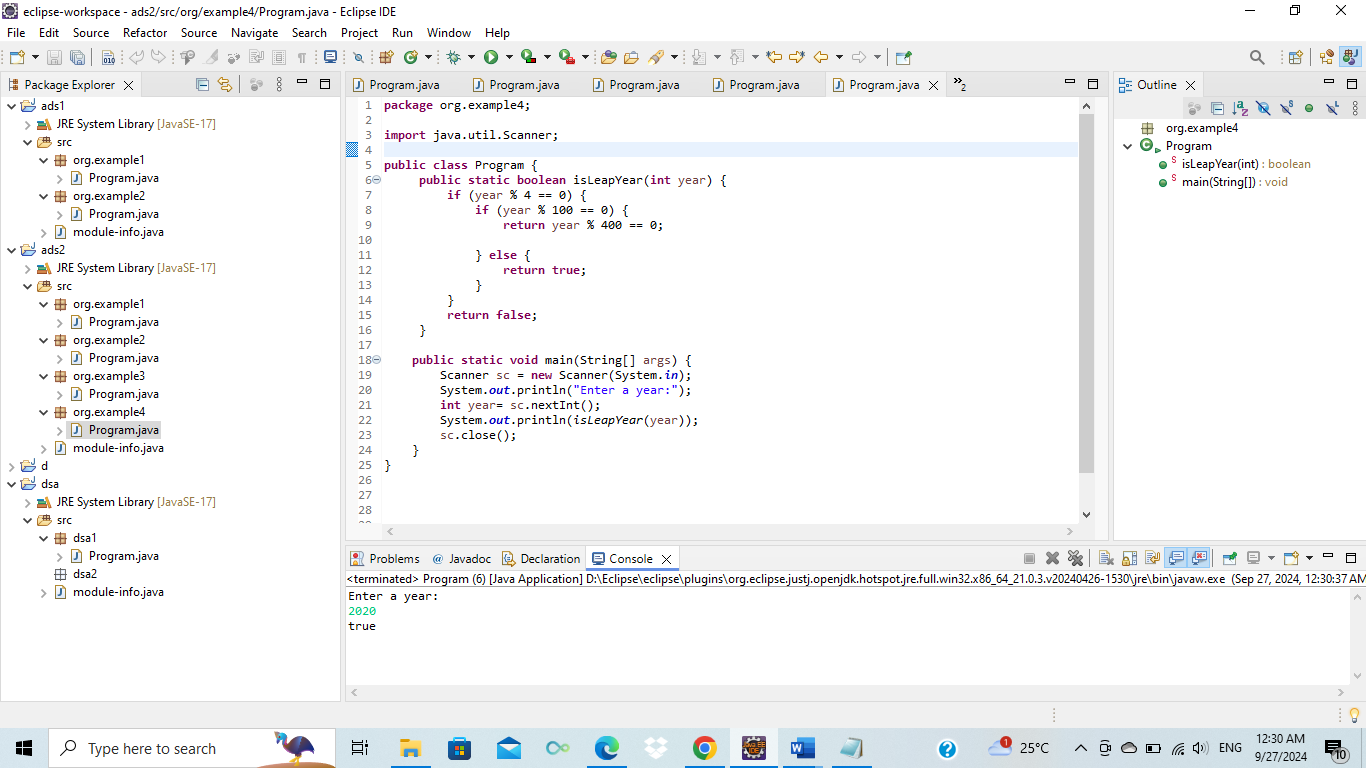
**int** year= sc.nextInt();

System.***out***.println(*isLeapYear*(year));

sc.close();

}

}



**Flowchart**:

Start

Input year

Check If year leap or not

If \_ else condition

year % 4 == 0

year % 100 == 0

year % 400 == 0; then True

otherwise false.

End

6. Find Square Root

Problem: Write a Java program to find the square root of a given number (using integer approximation).

Test Cases:

Input: x = 16

Output: 4

Input: x = 27

Output: 5

7. Find Repeated Characters in a String

Problem: Write a Java program to find all repeated characters in a string.

Test Cases:

Input: "programming"

Output: ['r', 'g', 'm']

Input: "hello"

Output: ['l']

8. First Non-Repeated Character

Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"

Output: 't'

Input: "aabbcc"

Output: null

9. Integer Palindrome

Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

Input: 121

Output: true