**1.Counting duplicate characters: Write a program that counts duplicate characters from a given string.**

**package** Problems;

**public** **class** A01\_CountDuplicateChar {

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

{

System.***out***.println("No of Duplicate character from the given string:");

}

String str = "anasooya";

**for** (**int** i = 0; i < str.length(); i++) {

**char** ch = str.charAt(i);

**int** count = 0;

**for** (**int** j = i+1; j < str.length(); j++) {

**if** (ch == str.charAt(j)) {

count++;

}

}

**if** (count > 0) {

System.***out***.println("'" + ch + "' is repeated " + (count) +" times.");

}

}

}

}

**2.Finding the first non-repeated character: Write a program that returns the first non-repeated character from a given string.**

**package** Problems;

**public** **class** A02\_Non\_Repeated\_Char {

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

{

System.***out***.println("The first non-repeated character from the given string:");

}

String name = "anasooya murali";

**for** (**int** i = 0; i < name.length(); i++) {

**int** count = 0;

**char** ch = name.charAt(i);

**for** (**int** j = i + 1; j < name.length(); j++) {

**if** (ch == name.charAt(j)) {

count++;

**break**;

}

}

**if** (count == 0) {

System.***out***.println("'" + ch + "' is the first non-repeated character.");

**break**;

}

}

}

}

**3.Reversing letters and words: Write a program that reverses the letters of each word and a program that reverses the letters of each word and the words themselves.**

**package** Problems;

**public** **class** A03\_Reverse\_String {

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

String name = "anasooya shaiju";

String split[] = name.split(" ");

String output = "";

**for** (**int** i = 0; i < split.length; i++)

{

String str = split[i];

String rev = "";

**for** (**int** j = str.length() - 1; j >= 0; j--) {

rev = rev + str.charAt(j);

}

output = output + rev + " ";

}

System.***out***.print("Reversed string : " + output);

}

}

**4.Checking whether a string contains only digits: Write a program that checks whether the given string contains only digits.**

**package** Problems;

**public** **class** A04\_String\_contains\_Only\_Digit

{

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

String str = "Anasooya 16";

**int** len = str.length();

System.***out***.println("Given string contains only digit: " + *onlyDigits*(str, len));

}

**public** **static** **boolean**

onlyDigits(String str, **int** n)

{

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

**if** (str.charAt(i) < '0'

|| str.charAt(i) > '9') {

**return** **false**;

}

}

**return** **true**;

}

}

**5.Counting vowels and consonants: Write a program that counts the number of vowels and consonants in a given string. Do this for the English language, which has five vowels (a, e, i, o, and u).**

**package** Problems;

**public** **class** A05\_Vowels\_Consonants {

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

{

System.***out***.println("From the given name");

}

**int** vowels=0;

**int** consonants=0;

String name="Anasooya murali";

**char**[] ch=name.toCharArray();

name=name.toLowerCase();

**for**(**int** i=0;i<name.length();i++)

{

**if**(ch[i]=='a' || ch[i]=='e' || ch[i]=='i' || ch[i]=='o' || ch[i]=='u')

vowels++;

**else** **if**(ch[i]>'a'&& ch[i]<'z')

consonants++;

}

System.***out***.println("Count of vowels from the given name :" + vowels);

System.***out***.println("Count of consonants from the given name :" + consonants);

}

}

**6.Counting occurrences of a certain character: Write a program that counts the occurrences of a certain character in a given string.**

**package** Problems;

**public** **class** A06\_Occurrences\_Certain\_Character

{

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

{

System.***out***.println("The counts of the occurrences of a certain character in a given string");

String name = "anasooya Murali";

**for**(**char** search = 'a'; search <= 'z'; search++)

{

**int** count=0;

**for**(**int** i=0; i<name.length(); i++)

{

**if**(name.charAt(i) == search)

count++;

}

**if** (count>0)

{

System.***out***.println("The Character '"+search+"' repeated "+count+" times.");

}

}

}

}

**7.Converting String into int, long, float, or double: Write a program that converts the given String object (representing a number) into int, long, float, or double.**

**package** Problems;

**public** **class** A07\_Converting\_String {

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

String str = "68434";

System.***out***.println("Converting string to int,long,float and double :");

**int** intValue = Integer.*parseInt*(str);

System.***out***.println("Int value: " + intValue);

**long** longValue = Long.*parseLong*(str);

System.***out***.println("Long value: " + longValue);

**float** floatValue = Float.*parseFloat*(str);

System.***out***.println("Float value: " + floatValue);

**double** doubleValue = Double.*parseDouble*(str);

System.***out***.println("Double value: " + doubleValue);

}

}

**8.Removing white spaces from a string: Write a program that removes all white spaces from the given string.**

**package** Problems;

**public** **class** A08\_Removing\_White\_Spaces {

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

String line = "Im Working in Tentacle technology";

line = line.replaceAll(" ", "");

System.***out***.println("After Removing white spaces from the given string: \n"+(line));

}

}

**9.Joining multiple strings with a delimiter: Write a program that joins the given strings by the given delimiter.**

**package** Problems;

**public** **class** A09\_Joining\_Strings {

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

String line1 = "my name is Anasooya";

String line2 = "Working in tentacle technology.";

String joinedString = line1 + "," + line2;

System.***out***.println("Joined string: " + joinedString);

}

}

**11.Checking whether a string is a palindrome: Write a program that determines whether the given string is a palindrome or not.**

**package** Problems;

**public** **class** A11\_Palindrome {

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

String word = "malayalam";

System.***out***.println("The given word is:" +(word));

String reverse= "";

**for** (**int** i = 0; i < word.length(); i++) {

reverse = word.charAt(i) + reverse;

}

System.***out***.println("The reversed word is:" +(reverse));

**if** (reverse.equals(word))

System.***out***.println("So the Given word is:Palindrome");

**else**

System.***out***.println("So the given word is:Not Palindrome");

}

}

**12.Removing duplicate characters: Write a program that removes the duplicate characters from the given string.**

**package** Problems;

**public** **class** A12\_Removing\_DuplicateChar {

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

String name = "anasooya murali";

String result = "";

**for** (**int** i = 0; i < name.length(); i++) {

**char** ch = name.charAt(i);

**if** (result.indexOf(ch) < 0) {

result += ch;

}

}

System.***out***.println("After removing duplicate character:"+(result));

}

}

**13.Removing given characters: Write a program that removes the given character from the given string.**

**package** Problems;

**public** **class** A13\_Removing\_GivenCharacter {

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

String name = "anusooya Murali";

**char** ch = 'a';

String remove = name.replace(String.*valueOf*(ch), "");

System.***out***.println("String after removing the character '" + ch + "': " + remove);

}

}

**14.Finding the character with the most appearances: Write a program that finds the character with the most appearances in the given string.**

**package** Problems;

**public** **class** A14\_Character\_With\_MostAppearance {

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

String name = "anasooya murali";

**char** maxChar = ' ';

**int** maxCount = 0;

**for** (**int** i = 0; i < name.length(); i++) {

**char** Charofname = name.charAt(i);

**int** Countofchar = 0;

**for** (**int** j = 0; j < name.length(); j++) {

**if** (name.charAt(j) == Charofname) {

Countofchar++;

}

}

**if** (Countofchar > maxCount) {

maxChar = Charofname;

maxCount =Countofchar;

}

}

System.***out***.println("The character with the most appearances is: '" + maxChar + "'");

}

}

**15.Sorting an array of strings by length: Write a program that sorts by the length of the given array of strings.**

**package** Problems;

**import** java.util.Arrays;

**public** **class** A15\_Sorting\_Array {

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

String skills[]= {"logical", "analytical", "critical", "computational",};

System.***out***.println("Given skill in string as array: " + Arrays.*toString*(skills));

**for** (**int** i = 0; i < skills.length; i++) {

**for** (**int** j = i+1; j < skills.length; j++) {

**if**(skills[i].length()>skills[j].length()) {

String str = skills[i];

skills[i] = skills[j];

skills[j] = str;

}

}

}

System.***out***.println("Sorted an array of strings by length:"+Arrays.*toString*(skills));

}

}

**MATHEMATICAL EXERCISE**

**1.Write method int calc(int, int) that multiplies two variables, m and n of type int, then divides the product by two, and outputs the remainder with respect to division by 7**.

**package** Mathematic;

**import** java.util.Scanner;

**public** **class** A01\_Divisible {

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

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

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

**int** m = scanner.nextInt();

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

**int** result = (m \* n / 2) % 7;

System.***out***.println("Result: " + result);

}

}

**2.Find the number as well as the sum of natural numbers, which are divisible by 2 or 7 upto a given maximum value (exclusive) and output it to the console. Write method void calcSumAndCountAllNumbersDivBy\_2\_Or\_7(int). Extend it so that it returns the two values instead of performing the console output.**

**package** Mathematic;

**import** java.util.Scanner;

**public** **class** A02\_Statistic

{

**void** calcSumAndCountAllNumbersDivBy\_2\_Or\_7()

{

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

System.***out***.print("Enter a maximum value: ");

**int** max = scanner.nextInt();

**int** count = 0;

**int** sum = 0;

**for** (**int** i = 1; i < max; i++)

{

**if** (i % 2 == 0 || i % 7 == 0)

{

count++;

sum += i;

}

}

System.***out***.println("Count: " + count);

System.***out***.println("Sum: " + sum);

}

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

{

A2\_Statistic obj= **new** A2\_Statistic();

obj.calcSumAndCountAllNumbersDivBy\_2\_Or\_7();

}

}

**3.Create the methods boolean isEven(n) and boolean isOdd(n) that will check if thepassed integer is even or odd, respectively.**

**package** Mathematic;

**public** **class** A03\_Odd\_even {

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

**int** num= 79;

System.***out***.println(num + " is even: " + *isEven*(num));

System.***out***.println(num + " is odd: " + *isOdd*(num));

}

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

**return** n % 2 == 0;

}

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

**return** n % 2 != 0;

}

}

**4.Write method String numberAsText(int) which, for a given positive number, convertsthe respective digits into corresponding text. Start with the following fragment for the last digit of a number:**

**package** Mathematic;

**public** **class** A04\_NumAsText {

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

**int** num = 87;

String text = *numberAsText*(num);

System.***out***.println(text);

}

**static** String numberAsText(**int** num) {

**if** (num == 0) {

**return** "zero";

}

String[] ones = {"", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"};

String[] tens = {"", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"};

String[] teens = {"ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};

String text = "";

**if** (num >= 1000000) {

text += *numberAsText*(num / 1000000) + " million ";

num %= 1000000;

}

**if** (num >= 1000) {

text += *numberAsText*(num / 1000) + " thousand ";

num %= 1000;

}

**if** (num >= 100) {

text += ones[num / 100] + " hundred ";

num %= 100;

}

**if** (num >= 20) {

text += tens[num / 10] + " ";

num %= 10;

} **else** **if** (num >= 10) {

text += teens[num - 10] + " ";

num = 0;

}

**if** (num > 0) {

text += ones[num] + " ";

}

**return** text.trim();

}

}

**5.Write method List calcPerfectNumbers(int) that calculates the perfectnumbers up to a maximum value, say 10,000.**

**package** Mathematic;

**import** java.util.List;

**import** java.util.ArrayList;

**public** **class** A05\_PerfectNum {

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

**int** max = 10000;

List<Integer> perfectNumbers = **new** ArrayList<>();

**for** (**int** i = 1; i <= max; i++) {

**int** sum = 0;

**for** (**int** j = 1; j < i; j++) {

**if** (i % j == 0) {

sum += j;

}

}

**if** (sum == i) {

perfectNumbers.add(i);

}

}

System.***out***.println("Perfect numbers up to " + max + ": \n" + perfectNumbers);

}

}

**6.Write method List calcPrimesUpTo(int) to compute all prime numbers upto a given value. As a reminder, a prime number is a natural number greater than 1 andexclusively divisible by itself and by 1. To compute a prime number, the so-called Sieveof Eratosthenes was described before.**

**package** Mathematic;

**import** java.util.\*;

**public** **class** A06\_PrimeNumber {

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

System.***out***.println("prime number upto 15 to 50 :");

List<Integer> primes = *calcPrimesUpTo*(50);

System.***out***.println("Primes: " + primes);

}

**public** **static** List<Integer> calcPrimesUpTo(**int** n) {

List<Integer> primes = **new** ArrayList<Integer>();

**for** (**int** i = 15; i <= n; i++) {

**int** count = 0;

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

**if** (i % j == 0) {

count++;

}

}

**if** (count == 0) {

primes.add(i);

}

}

**return** primes;

}

}

**7. Compute all pairs of prime numbers with a distance of 2 (twin), 4 (cousin), and 6 (sexy)up to an upper bound for n. For twins then the following is true:**

**package** Mathematic;

**public** **class** A07\_PrimeNumber\_Pair {

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

**int** n = 20;

**for** (**int** i = 2; i <= n - 6; i++) {

**if** (*isPrime*(i) && *isPrime*(i + 6)) {

System.***out***.println("(" + i + "," + (i + 6) + ") is a sexy prime pair");

}

**if** (*isPrime*(i) && *isPrime*(i + 4)) {

System.***out***.println("(" + i + "," + (i + 4) + ") is a cousin prime pair");

}

**if** (*isPrime*(i) && *isPrime*(i + 2)) {

System.***out***.println("(" + i + "," + (i + 2) + ") is a twin prime pair");

}

}

}

**public** **static** **boolean** isPrime(**int** num) {

**if** (num < 2) {

**return** **false**;

}

**for** (**int** i = 2; i < num; i++) {

**if** (num % i == 0) {

**return** **false**;

}

}

**return** **true**;

}

}

**8.Create method int calcChecksum(String) that performs the following position-based calculation for the checksum of a number of any length given as a string, with the n digits modeled as z1 to zn:**

**package** Mathematic;

**public** **class** A08\_CheckSum {

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

String number = "457735";

**int** sum = 0;

**for** (**int** i = 0; i < number.length(); i++) {

sum += (i + 1) \* (number.charAt(i) - '0');

}

**int** checksum = sum % 10;

System.***out***.println("Checksum of " + number + " is " + checksum);

}

}

**9.Compute all combinations of the values a, b, and c (each starting from 1 and less than100) for which the following formula holds: a 2 + b 2 = c 2**

**package** Mathematic;

**public** **class** A09\_Computation\_abc {

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

**for** (**int** a = 1; a < 100; a++) {

**for** (**int** b = 1; b < 100; b++) {

**for** (**int** c = 1; c < 100; c++) {

**if** (a\*a + b\*b == c\*c) {

System.***out***.println(a + "^2 + " + b + "^2 = " + c + "^2");

}

}

}

}

}

}

**10. Compute all combinations of the values a, b, c, and d (each starting from 1 and less than100) for which the following formula holds: a 2 + b 2 = c 2 + d 2**

**package** Mathematic;

**public** **class** A10\_Computation\_abcd {

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

**for** (**int** a = 1; a < 100; a++) {

**for** (**int** b = a; b < 100; b++) {

**for** (**int** c = 1; c < 100; c++) {

**for** (**int** d = c; d < 100; d++) {

**if** (a\*a + b\*b == c\*c + d\*d) {

System.***out***.println(a + "^2 + " + b + "^2 = " + c + "^2 + " + d + "^2");

}

}

}

}

}

}

}

**11. Any natural number greater than 1 can be represented as a multiplication of primes— remember the fact that 2 is also a prime. Write method List calcPrimeFactors(int) that returns a list of prime numbers whose multiplication yields the desired number.**

**package** Mathematic;

**import** java.util.\*;

**public** **class** A12\_PrimeFactor {

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

**int** n = 24;

List<Integer> factors =*calcPrimeFactors*(n);

System.***out***.println("The prime factors of " + n + " are: " + factors);

}

**public** **static** List<Integer> calcPrimeFactors(**int** n) {

List<Integer> factors = **new** ArrayList<>();

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

**while** (n % i == 0) {

factors.add(i);

n /= i;

}

}

**return** factors;

}

}