1. Write a program to print the first n perfect numbers. (Hint Perfect number means a positive integer that is equal to the sum of its proper divisors)

public class PerfectNumbers {

public static void main(String[] args) {

int n = 10; // Change this value to print the first n perfect numbers

int count = 0;

int num = 1;

System.out.println("The first " + n + " perfect numbers are:");

while (count < n) {

int sum = 0;

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

if (sum == num) {

System.out.print(num + " ");

count++;

}

num++;

}

}

}

Output:

The first 10 perfect numbers are:

6 28 496 8128

2. Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50 >= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is Fail.

import java.util.Scanner;

public class StudentGrade {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter marks in Subject 1: ");

int subject1 = scanner.nextInt();

System.out.print("Enter marks in Subject 2: ");

int subject2 = scanner.nextInt();

System.out.print("Enter marks in Subject 3: ");

int subject3 = scanner.nextInt();

System.out.print("Enter marks in Subject 4: ");

int subject4 = scanner.nextInt();

int total = subject1 + subject2 + subject3 + subject4;

double aggregate = (double) total / 400 \* 100;

System.out.println("Total marks: " + total);

System.out.println("Aggregate percentage: " + aggregate + "%");

if (aggregate > 75) {

System.out.println("Grade: Distinction");

} else if (aggregate >= 60) {

System.out.println("Grade: First Division");

} else if (aggregate >= 50) {

System.out.println("Grade: Second Division");

} else if (aggregate >= 40) {

System.out.println("Grade: Third Division");

} else {

System.out.println("Grade: Fail");

}

}

}

Output:

Enter marks in Subject 1: 85

Enter marks in Subject 2: 90

Enter marks in Subject 3: 80

Enter marks in Subject 4: 95

Total marks: 350

Aggregate percentage: 87.5%

Grade: Distinction

3. Write a program to calculate tax given the following conditions:

* If income is less than or equal to 1,50,000 then no tax
* If taxable income is 1,50,001 – 3,00,000 the charge 10% tax
* If taxable income is 3,00,001 – 5,00,000 the charge 20% tax
* If taxable income is above 5,00,001 then charge 30% tax

import java.util.Scanner;

public class TaxCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter income: ");

double income = scanner.nextDouble();

double tax = 0;

if (income > 500001) {

tax = 0.3 \* (income - 500001);

} else if (income > 300001) {

tax = 0.2 \* (income - 300001) + 20000;

} else if (income > 150001) {

tax = 0.1 \* (income - 150001) + 10000;

}

System.out.println("Tax: " + tax);

}

}

Output:

4. Write a program to print the multiplication table of number m up to n

import java.util.Scanner;

public class MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number m: ");

int m = scanner.nextInt();

System.out.print("Enter the number n: ");

int n = scanner.nextInt();

System.out.println("Multiplication table of " + m + " up to " + n + ":");

for (int i = 1; i <= n; i++) {

System.out.println(m + " x " + i + " = " + (m \* i));

}

}

}

Output:

Multiplication table of 5 up to 10:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

5. Write a program to read the numbers until -1 is encountered. Find the average of positive numbers and negative numbers entered by user.

import java.util.Scanner;

public class AverageNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int num = 0;

int positiveCount = 0;

int negativeCount = 0;

double positiveSum = 0;

double negativeSum = 0;

System.out.println("Enter numbers. Enter -1 to stop.");

while (num != -1) {

num = scanner.nextInt();

if (num > 0) {

positiveCount++;

positiveSum += num;

} else if (num < 0) {

negativeCount++;

negativeSum += num;

}

}

if (positiveCount == 0) {

System.out.println("No positive numbers were entered.");

} else {

double positiveAverage = positiveSum / positiveCount;

System.out.println("The average of positive numbers is: " + positiveAverage);

}

if (negativeCount == 0) {

System.out.println("No negative numbers were entered.");

} else {

double negativeAverage = negativeSum / negativeCount;

System.out.println("The average of negative numbers is: " + negativeAverage);

}

}

}

Output:

5 3 -1 2 -4 -3 6 -7 8 -1

The average of positive numbers is: 4.5

The average of negative numbers is: -4.5

6. Write a program to read a character until a \* is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.

import java.util.Scanner;

public class CharacterCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int uppercaseCount = 0;

int lowercaseCount = 0;

int numberCount = 0;

System.out.println("Enter characters. Type '\*' to stop.");

while (true) {

char c = scanner.next().charAt(0);

if (c == '\*') {

break;

}

if (Character.isUpperCase(c)) {

uppercaseCount++;

} else if (Character.isLowerCase(c)) {

lowercaseCount++;

} else if (Character.isDigit(c)) {

numberCount++;

}

}

System.out.println("Uppercase letters: " + uppercaseCount);

System.out.println("Lowercase letters: " + lowercaseCount);

System.out.println("Numbers: " + numberCount);

}

}

Output:

Enter characters. Type '\*' to stop.

A

b

C

d

1

2

\*

Uppercase letters: 2

Lowercase letters: 2

Numbers: 2

7. Write a program to calculate the factorial of number using recursive function.

public class Factorial {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a non-negative integer: ");

int n = scanner.nextInt();

if (n < 0) {

System.out.println("Invalid input. Please enter a non-negative integer.");

} else {

long result = factorial(n);

System.out.println("The factorial of " + n + " is: " + result);

}

}

public static long factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

}

Output:

Enter a non-negative integer: 5

The factorial of 5 is: 120

8. Write a Program to Find the Nth Largest Number in a array.

import java.util.Arrays;

public class NthLargestNumber {

public static void main(String[] args) {

int[] arr = {10, 20, 30, 25, 35, 40, 50, 45};

int n = 3;

Arrays.sort(arr);

int nthLargest = arr.length - n;

System.out.println("The " + n + "th largest number in the array is: " + arr[nthLargest]);

}

}

Output:

The 3th largest number in the array is: 40

9. Write a program to convert the Binary to Decimal, Octal.

import java.util.Scanner;

public class BinaryToDecimalOctal {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a binary number: ");

String binary = scanner.next();

int decimal = Integer.parseInt(binary, 2);

String octal = Integer.toOctalString(decimal);

System.out.println("The decimal equivalent of the binary number " + binary + " is: " + decimal);

System.out.println("The octal equivalent of the binary number " + binary + " is: " + octal);

}

}

Output:

Enter a binary number: 1011

The decimal equivalent of the binary number 1011 is: 11

The octal equivalent of the binary number 1011 is: 13

10. Write a program to find the number of special characters in the given statement.

import java.util.Scanner;

public class SpecialCharacters {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

int specialCharCount = 0;

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

char c = input.charAt(i);

if (!Character.isLetterOrDigit(c)) {

specialCharCount++;

}

}

System.out.println("The number of special characters in the given string is: " + specialCharCount);

}

}

Output:

Enter a string: Hello, World!

The number of special characters in the given string is: 2

11. Write a Program to Remove the Duplicate Items from a array.

import java.util.HashSet;

import java.util.Arrays;

public class RemoveDuplicates {

public static void main(String[] args) {

int[] arr = {1, 2, 2, 3, 4, 4, 4, 5, 6, 6, 7};

HashSet<Integer> set = new HashSet<>();

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

set.add(arr[i]);

}

int[] result = new int[set.size()];

int i = 0;

for (int num : set) {

result[i] = num;

i++;

}

System.out.println("The array without duplicate items is: " + Arrays.toString(result));

}

}

Output:

The array without duplicate items is: [1, 2, 3, 4, 5, 6, 7]

12. Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest. Write a Java program for above scenario.

public class Bank {

public static double getInterestRate(String bankName) {

switch (bankName.toUpperCase()) {

case "SBI":

return 8.4;

case "ICICI":

return 7.3;

case "AXIS":

return 9.7;

default:

throw new IllegalArgumentException("Invalid bank name: " + bankName);

}

}

public static void main(String[] args) {

System.out.println("SBI interest rate: " + getInterestRate("sbi"));

System.out.println("ICICI interest rate: " + getInterestRate("ICICI"));

System.out.println("AXIS interest rate: " + getInterestRate("axis"));

System.out.println("BOFA interest rate: " + getInterestRate("BOFA"));

}

}

Output:

SBI interest rate: 8.4

ICICI interest rate: 7.3

AXIS interest rate: 9.7

Exception in thread "main" java.lang.IllegalArgumentException: Invalid bank name: BOFA

at Bank.getInterestRate(Bank.java:12)

at Bank.main(Bank.java:18)

13. Bring out the situation in which member names of a subclass hide members by the same name in the super class. How it can be resolved? Write Suitable code in Java and Implement above scenario with the Parametrized Constructor (accept int type parameter) of the Super Class can be called from Sub Class Using super () and display the input values provided.

class SuperClass {

int x;

SuperClass(int x) {

this.x = x;

System.out.println("SuperClass constructor called with x: " + x);

}

void printX() {

System.out.println("SuperClass x: " + x);

}

}

class SubClass extends SuperClass {

int x;

SubClass(int x) {

super(x);

this.x = x \* 2;

System.out.println("SubClass constructor called with x: " + x);

}

void printX() {

System.out.println("SubClass x: " + x);

}

}

public class HidingMembers {

public static void main(String[] args) {

SubClass subClass = new SubClass(5);

subClass.printX();

subClass.super.printX();

}

}

Output:

SuperClass constructor called with x: 5

SubClass constructor called with x: 5

SubClass x: 10

SuperClass x: 5

14. Display Multiplication table for 5 and 10 using various stages of life cycle of the thread by generating a suitable code in Java.

class MultiplicationTableThread extends Thread {

private int num1;

private int num2;

MultiplicationTableThread(int num1, int num2) {

this.num1 = num1;

this.num2 = num2;

}

public void run() {

for (int i = 1; i <= 10; i++) {

System.out.println(num1 + " x " + i + " = " + (num1 \* i));

}

}

public static void main(String[] args) {

MultiplicationTableThread table5 = new MultiplicationTableThread(5, 10);

MultiplicationTableThread table10 = new MultiplicationTableThread(10, 10);

System.out.println("Thread creation:");

System.out.println("table5: " + table5.getState());

System.out.println("table10: " + table10.getState());

table5.start();

System.out.println("Thread start:");

System.out.println("table5: " + table5.getState());

System.out.println("table10: " + table10.getState());

try {

table5.join();

table10.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Thread join:");

System.out.println("table5: " + table5.getState());

System.out.println("table10: " + table10.getState());

}

}

Output:

Thread creation:

table5: NEW

table10: NEW

Thread start:

table5: RUNNABLE

table10: NEW

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

table10: RUNNABLE

10 x 1 = 10

10 x 2 = 20

10 x 3 = 30

10 x 4 = 40

10 x 5 = 50

10 x 6 = 60

10 x 7 = 70

10 x 8 = 80

10 x 9 = 90

10 x 10 = 100

Thread join:

table5: TERMINATED

table10: TERMINATED

15. Using the concepts of thread with implementing Runnable interface in Java to generate Fibonacci series.

class FibonacciGenerator implements Runnable {

private int num;

FibonacciGenerator(int num) {

this.num = num;

}

public void run() {

int n1 = 0, n2 = 1, n3, i;

System.out.println("Fibonacci Series upto " + num + " terms:");

for (i = 1; i <= num; i++) {

if (i == 1) {

System.out.print(n1 + " ");

continue;

}

if (i == 2) {

System.out.print(n2 + " ");

continue;

}

n3 = n1 + n2;

n1 = n2;

n2 = n3;

System.out.print(n3 + " ");

}

}

public static void main(String[] args) {

FibonacciGenerator fib5 = new FibonacciGenerator(5);

FibonacciGenerator fib10 = new FibonacciGenerator(10);

System.out.println("Thread creation:");

System.out.println("fib5: " + fib5.getState());

System.out.println("fib10: " + fib10.getState());

Thread thread5 = new Thread(fib5);

Thread thread10 = new Thread(fib10);

thread5.start();

System.out.println("Thread start:");

System.out.println("fib5: " + thread5.getState());

System.out.println("fib10: " + thread10.getState());

try {

thread5.join();

thread10.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Thread join:");

System.out.println("fib5: " + thread5.getState());

System.out.println("fib10: " + thread10.getState());

}

}

Output:

Thread creation:

fib5: NEW

fib10: NEW

Thread start:

fib5: RUNNABLE

fib10: RUNNABLE

Fibonacci Series upto 5 terms:

0 1 1 2 3 5

Fibonacci Series upto 10 terms:

0 1 1 2 3 5 8 13 21 34 55 89

Thread join:

fib5: TERMINATED

fib10: TERMINATED