1. GCD

public class GCD1{

public static int GCDnum(int n1,int n2){

if(n2== 0) {

return n1;

}

else{

return GCDnum(n2, n1%n2);

}

}

public static void main(String[] args) {

int num1 = 10;

int num2 = 8;

int gcd = GCDnum(num1, num2);

System.out.println("GCD of "+num1+" & "+num2+" is: "+gcd);

}

}

1. LCM

public class LCM{

public static int GCDnum(int n1,int n2){

if(n2== 0) {

return n1;

}

else{

return GCDnum(n2, n1%n2);

}

}

public static int LCMnum(int n1, int n2) {

return (n1 \* n2)/GCDnum(n1, n2);

}

public static void main(String[] args) {

int num1 = 10;

int num2 = 8;

int lcm = LCMnum(num1, num2);

System.out.println("LCM of "+num1+" & "+num2+" is: "+lcm);

}

}

1. Prime Factors

public class PrimeFactors{

public static void primeFactors(int number) {

while (number % 2 == 0) {

System.out.print(2 + " ");

number = number/2;

}

for (int i = 3; i <= Math.sqrt(number); i= i+2) { //for numbers aboce 3

while (number % i == 0) {

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

number /= i;

}

}

if (number > 2) { //for remainder is its more than 2 then...

System.out.print(number);

}

System.out.println();

}

public static void main(String[] args) {

int number = 44;

System.out.print("Prime factors : ");

primeFactors(number);

}

}

1. Palindrome

public class PalindromeN{

public static boolean isPalindrome(int num) {

String str = Integer.toString(num);

return isPalindromeRecursive(str, 0, str.length() - 1); //recursion

}

private static boolean isPalindromeRecursive(String str, int start, int end) {

if (start >= end) { //check if starting n end index match -> Palindrome

return true;

}

if (str.charAt(start) != str.charAt(end)) { //to check if 1st n last character match

return false;

}

return isPalindromeRecursive(str, start + 1, end - 1); //recursion

}

public static void main(String[] args) {

int num = 550055;

if (isPalindrome(num)) {

System.out.println(num + ": palindrome.");

} else {

System.out.println(num + ": not Palindrome.");

}

}

}

1. Prime

public class Prime{

static boolean findPrime(int n, int i){

if(n<=2){

return (n==2)? true:false; //for numbers <= 2 coz only 2 prime;

}

if(n%i==0){

return false;

}

if (i\*i > n) {

return true;

}

else{

return findPrime(n, i+1);

}

}

public static void main(String[] args){

int num = 23;

boolean res = findPrime(num, 2);

System.out.println(res);

if(res==true){

System.out.println("Prime");

}

else{

System.out.println("not Prime");

}

}

}

1. Armstrong

public class Armstrong{

public static int countDigits(int number) {

int count = 0;

while (number != 0) {

number /= 10;

count++;

}

return count;

}

public static boolean checkArm(int number) {

int originalNumber = number;

int numDigits = countDigits(number);

int sum = 0;

while (number > 0) {

int digit = number % 10;

sum += Math.pow(digit, numDigits);

number /= 10;

}

return sum == originalNumber;

}

public static void main(String[] args) {

int number = 137;

if (checkArm(number)) {

System.out.println(number + " Armstrong.");

}

else{

System.out.println(number + " not Armstrong.");

}

}

}

1. Perfect

public class Armstrong{

public static int countDigits(int number) {

int count = 0;

while (number != 0) {

number /= 10;

count++;

}

return count;

}

public static boolean checkArm(int number) {

int originalNumber = number;

int numDigits = countDigits(number);

int sum = 0;

while (number > 0) {

int digit = number % 10;

sum += Math.pow(digit, numDigits);

number /= 10;

}

return sum == originalNumber;

}

public static void main(String[] args) {

int number = 137;

if (checkArm(number)) {

System.out.println(number + " Armstrong.");

}

else{

System.out.println(number + " not Armstrong.");

}

}

}

1. Ramanujan Number

public class RNum {

public static void main(String[] args) {

int number = 1729; // Change this to the number you want to check

if (isRamanujan(number)) {

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

} else {

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

}

}

public static boolean isRamanujan(int number) {

int cubeRoot = (int) Math.cbrt(number);

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

int cube1 = i \* i \* i;

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

int cube2 = j \* j \* j;

if (cube1 + cube2 == number) {

return true;

}

}

}

return false;

}

}