Implement pow(x, n), which calculates x raised to the power n

CODE:

import java.util.Scanner;

public class PowerFunction {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input base (x) and exponent (n)

System.out.print("Enter the base (x): ");

double x = scanner.nextDouble();

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

int n = scanner.nextInt();

// Calculate the result of pow(x, n)

double result = power(x, n);

// Display the result

System.out.println("Result of " + x + " raised to the power " + n + " = " + result);

scanner.close(); // Closing the scanner object

}

public static double power(double x, int n) {

// Base case: if exponent is 0, return 1

if (n == 0) {

return 1;

}

// If exponent is negative, calculate power of (1/x) and change sign of exponent

if (n < 0) {

x = 1 / x;

n = -n;

}

// Recursive calculation for positive exponent

double half = power(x, n / 2);

double result = half \* half;

// If exponent is odd, multiply result by x

if (n % 2 != 0) {

result \*= x;

}

return result;

}

}

OUTPUT:

C:\javap>javac PowerFunction.java

C:\javap>java PowerFunction

Enter the base (x): 2.000000

Enter the exponent (n): 10

Result of 2.0 raised to the power 10 = 1024.0

