1. import java.util.Scanner;

public class Q1\_BinaryToDecimal {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Read the binary number from the user

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

String binaryStr = input.nextLine();

// Convert the binary number to decimal

int decimal = 0;

int base = 1;

for (int i = binaryStr.length() - 1; i >= 0; i--) {

if (binaryStr.charAt(i) == '1') {

decimal += base;

}

base \*= 2;

}

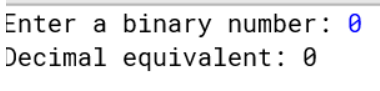
// Print the decimal equivalent

System.out.println("Decimal equivalent: " + decimal);

}

}

Output:



2.

import java.util.Scanner;

public class Q2\_DecimalToBinary {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Read the decimal number from the user

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

int decimal = input.nextInt();

// Convert the decimal number to binary

StringBuilder binary = new StringBuilder();

while (decimal > 0) {

int remainder = decimal % 2;

binary.append(remainder);

decimal /= 2;

}

// Reverse the binary string

binary.reverse();

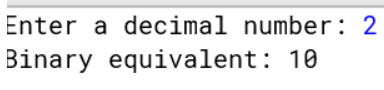
// Print the binary equivalent

System.out.println("Binary equivalent: " + binary);

}

}

Output:



3.

public class Q3\_BitwiseAndExample {

public static void main(String[] args) {

int num1 = 10; // binary representation: 1010

int num2 = 6; // binary representation: 0110

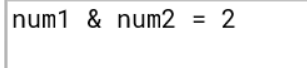
int result = num1 & num2;

System.out.println("num1 & num2 = " + result); // prints 2 (binary representation: 0010)

}

}

Output:



4.

public class Q4\_BitwiseComplementExample {

public static void main(String[] args) {

int num = 10; // binary representation: 0000 1010

int complement = ~num;

System.out.println("~num = " + complement); // prints -11 (binary representation: 1111 0101)

}

}

Output:



5. public class Q5\_LeftShiftExample {

public static void main(String[] args) {

int num = 5; // binary representation: 0000 0101

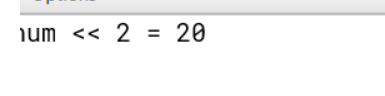
int result = num << 2;

System.out.println("num << 2 = " + result); // prints 20 (binary representation: 0001 0100)

}

}

Output:



6.

public class Q6\_SignedRightShiftExample {

public static void main(String[] args) {

int num = -15; // binary representation: 1111 0001

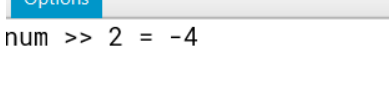
int result = num >> 2;

System.out.println("num >> 2 = " + result); // prints -4 (binary representation: 1111 1100)

}

}

Output:



7.

class GFG {

public static void main(String[] args) {

int[] arr = new int[5]; // Create a new int array with 5 elements

arr[0] = 10; // Set the value of the first element to 10

arr[1] = 20; // Set the value of the second element to 20

arr[2] = 30; // Set the value of the third element to 30

arr[3] = 40; // Set the value of the fourth element to 40

arr[4] = 50; // Set the value of the fifth element to 50

// Loop through each element of the array and print its index and value

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

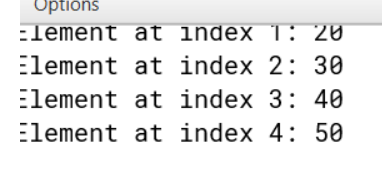
System.out.println("Element at index " + i + ": " + arr[i]);

}

}

}

Output:



8.

interface Square {

int calculate(int x);

}

class debug \_Q1\_interface {

public static void main(String args[]) {

// Create an instance of the Square interface using a lambda expression

Square s = (int x) -> x \* x;

int a = 5;

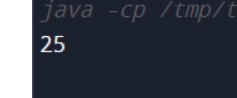
int ans = s.calculate(a); // Call the calculate method on the Square instance

System.out.println(ans);

}

}

Output:



9.

class parent {

void show() {}

}

class child extends parent {

@Override

void show() {}

}

public class Main {

public static void main(String[] args) {

// code to test parent and child classes

}

}

Output: