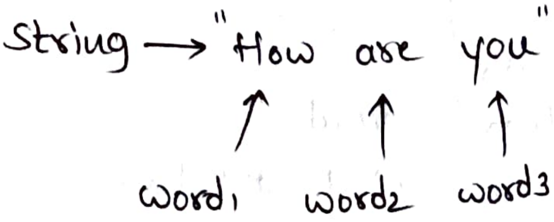
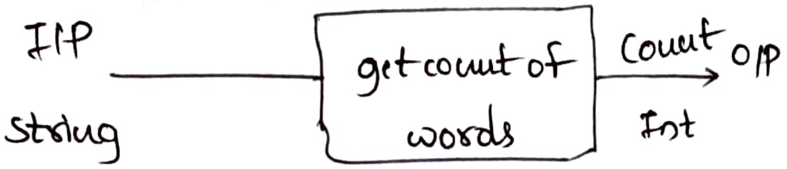
JAVA HANDS ON PRACTICE

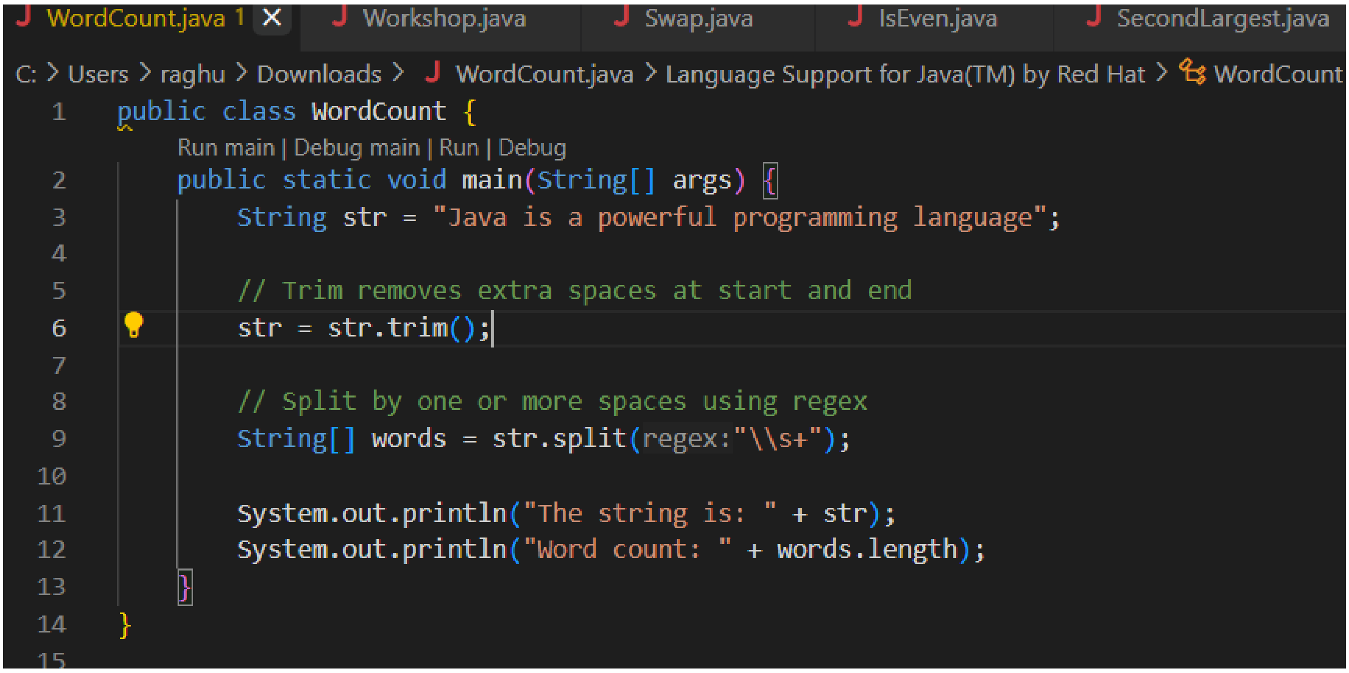
Q> Get count of words in a string



H	o	w		A	R	E		Y	o	u	

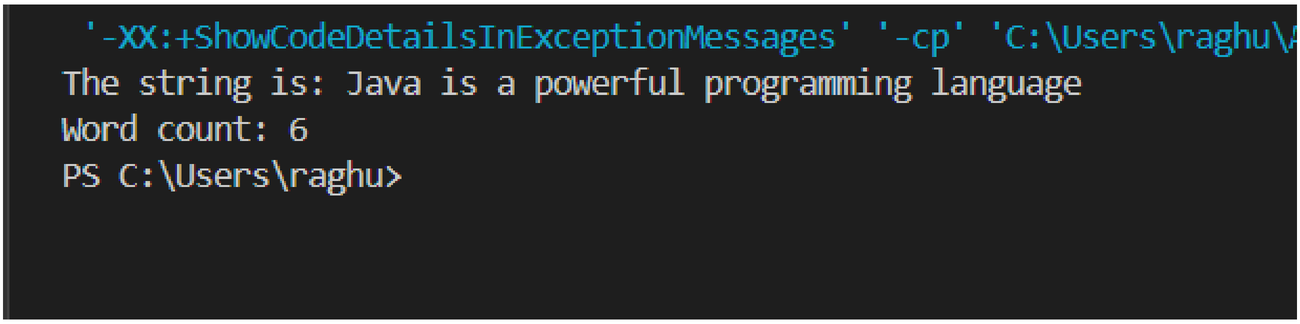
Counter → 1+1+1 = 3




WordCount.java 1
Workshop.java
Swap.java
IsEven.java
SecondLargest.java
C: > Users > raghu > Downloads > WordCount.java > Language Support for Java(TM) by Red Hat > WordCount
Run main | Debug main | Run | Debug

```
public class WordCount {
    public static void main(String[] args) {
        String str = "Java is a powerful programming language";

        // Trim removes extra spaces at start and end
        str = str.trim();

        // Split by one or more spaces using regex
        String[] words = str.split(regex:"\\s+");

        System.out.println("The string is: " + str);
        System.out.println("Word count: " + words.length);
    }
}
```

Output :



```
 '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\raghu\A
The string is: Java is a powerful programming language
Word count: 6
PS C:\Users\raghu>
```

Q7 Write a function to print Binary values of various input like Integers, also perform Shift operations on them.

1> Binary format

2> left, Right shift

Binary format printing

0/1	0/1	0/1	0/1
2^3	2^2	2^1	2^0
8	4	2	1

Ex: 10

1	0	1	0
2^3	2^2	2^1	2^0

1 Active

0 off

$2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^2 \times 0$

$8 + 0 + 2 + 0$

$8 + 2$

$= 10$

Binary "&" operation

Multiplication

```
 1 0 1 0
 1 0 0 0
 -------
 1 0 0 0
```

Bit Masking

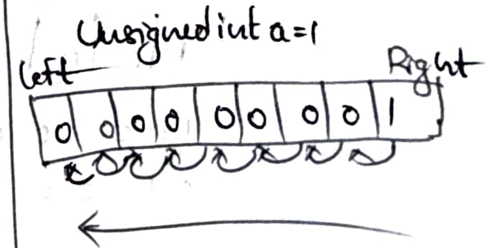
A	B	A&B
1	0	0
1	1	1
0	0	0
0	1	1

with the help of Bit masking we can find wheather the bit is active or not.

dec = 10   Binary $(1010)_2$

```
  1010       1010      1010      1010
& 1000     & 0100    & 0010      0001
  ----       ----      ----      ----
  1000       0000      0010      0000
```

```
1 0 0 0 → 1 output
0 1 0 0 → 0
0 0 1 0 → 1
0 0 0 1 → 0
```



Perform Shift operation

Step find Bit logic

Unsigned int n=1

n=n<<31

size of 4x8 = 32

32-1 = 31

for -ve integers (2's complement)

1's complement 1 0 1 0

0 1 0 1

Add 1 to 1's complement

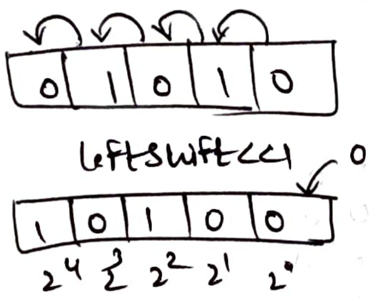
0 1 0 1

Add 1 1

0 1 1 0

0110 → -10

leftshift



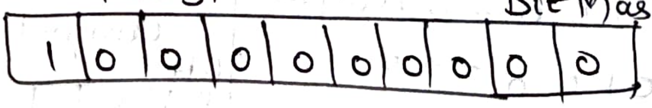
$2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 0$

$16 + 0 + 4 + 0 + 0 = 20$

left shift $\boxed{\text{Number} \times 2^i}$ → where i is number of shift

$\boxed{\text{Rightshift } \frac{\text{Number}}{2^i}}$

1<<31 Bit Masking



1 0 0 0 0 0 0 - - - - -

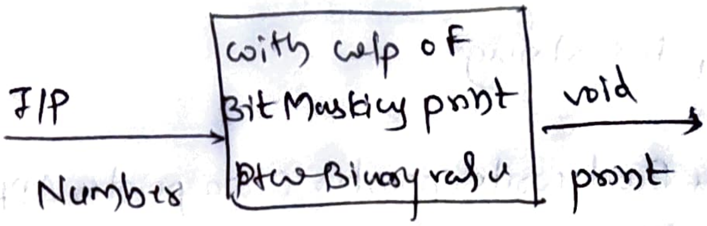
0 1 0 0 0 0 0 0 - - - - — No of bits can vary

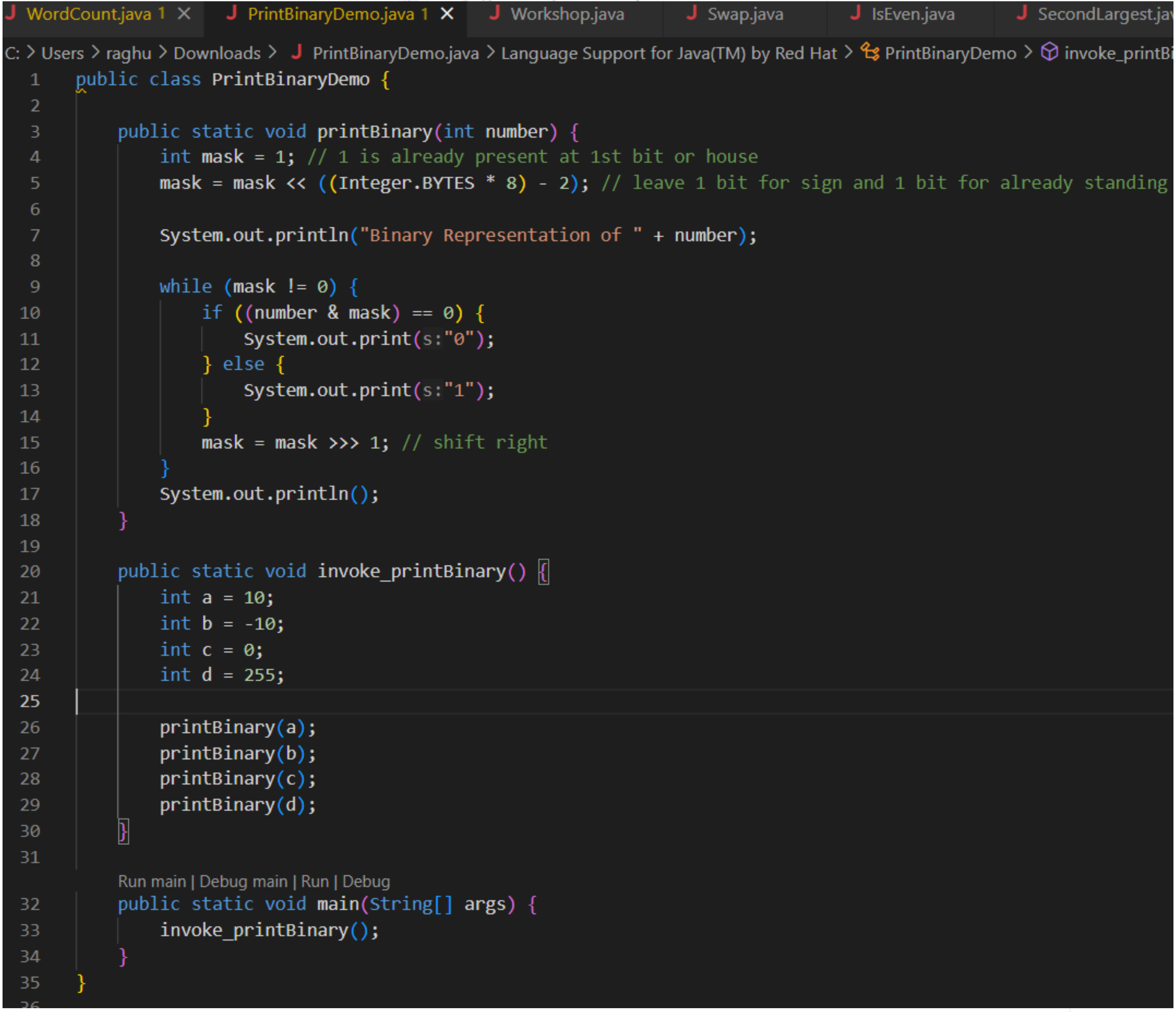
0 0 1 0 0 0 0 - - - — Based on Machine & compiler

0 0 0 1 0 0 0 — — —

0 0 0 0 1 0 0 — —

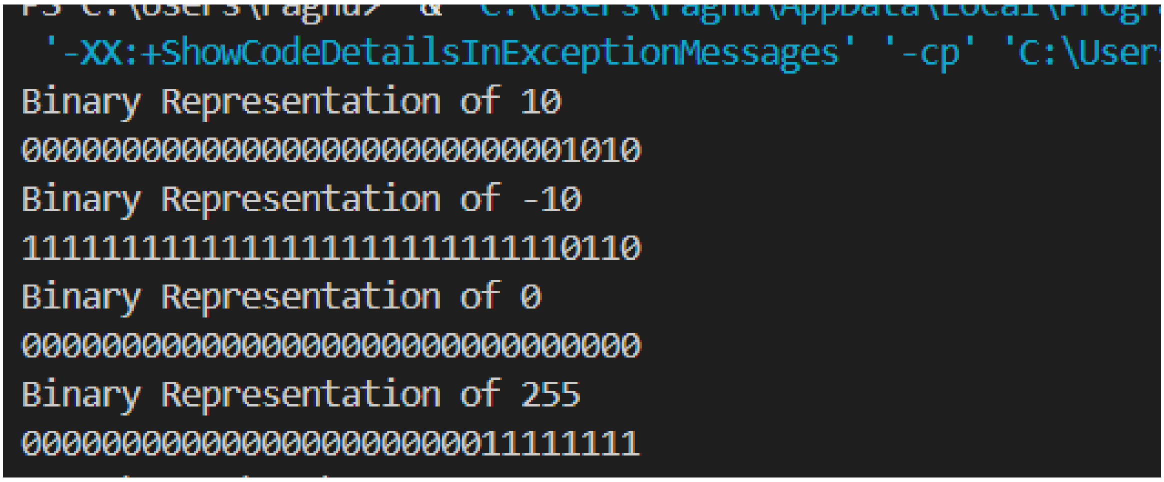
0 0 0 0 0 1 0 — —




WordCount.java 1
PrintBinaryDemo.java 1
Workshop.java
Swap.java
IsEven.java
SecondLargest.ja
C: > Users > raghu > Downloads > PrintBinaryDemo.java > Language Support for Java(TM) by Red Hat > PrintBinaryDemo > invoke_printBi
Run main | Debug main | Run | Debug

```java
public class PrintBinaryDemo {

    public static void printBinary(int number) {
        int mask = 1; // 1 is already present at 1st bit or house
        mask = mask << ((Integer.BYTES * 8) - 2); // leave 1 bit for sign and 1 bit for already standing

        System.out.println("Binary Representation of " + number);

        while (mask != 0) {
            if ((number & mask) == 0) {
                System.out.print(s:"0");
            } else {
                System.out.print(s:"1");
            }
            mask = mask >>> 1; // shift right
        }
        System.out.println();
    }

    public static void invoke_printBinary() {
        int a = 10;
        int b = -10;
        int c = 0;
        int d = 255;

        printBinary(a);
        printBinary(b);
        printBinary(c);
        printBinary(d);
    }

    public static void main(String[] args) {
        invoke_printBinary();
    }
}
```

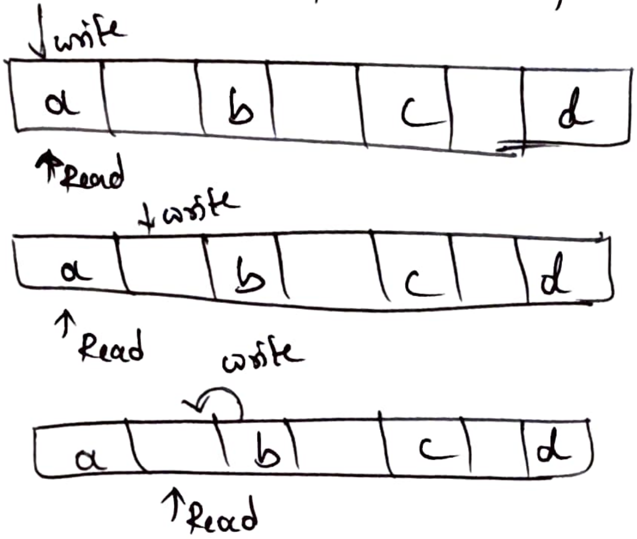
Output :



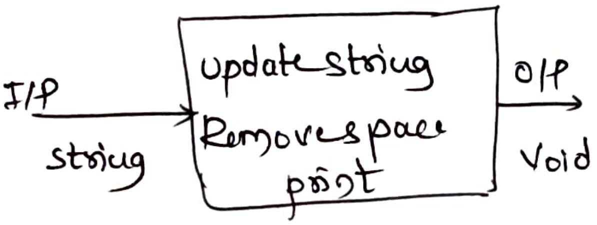
```
PS C:\Users\raghu> & 'C:\Users\raghu\AppData\Local\Progr
 '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\User
Binary Representation of 10
0000000000000000000000000001010
Binary Representation of -10
1111111111111111111111111110110
Binary Representation of 0
0000000000000000000000000000000
Binary Representation of 255
0000000000000000000000011111111
```

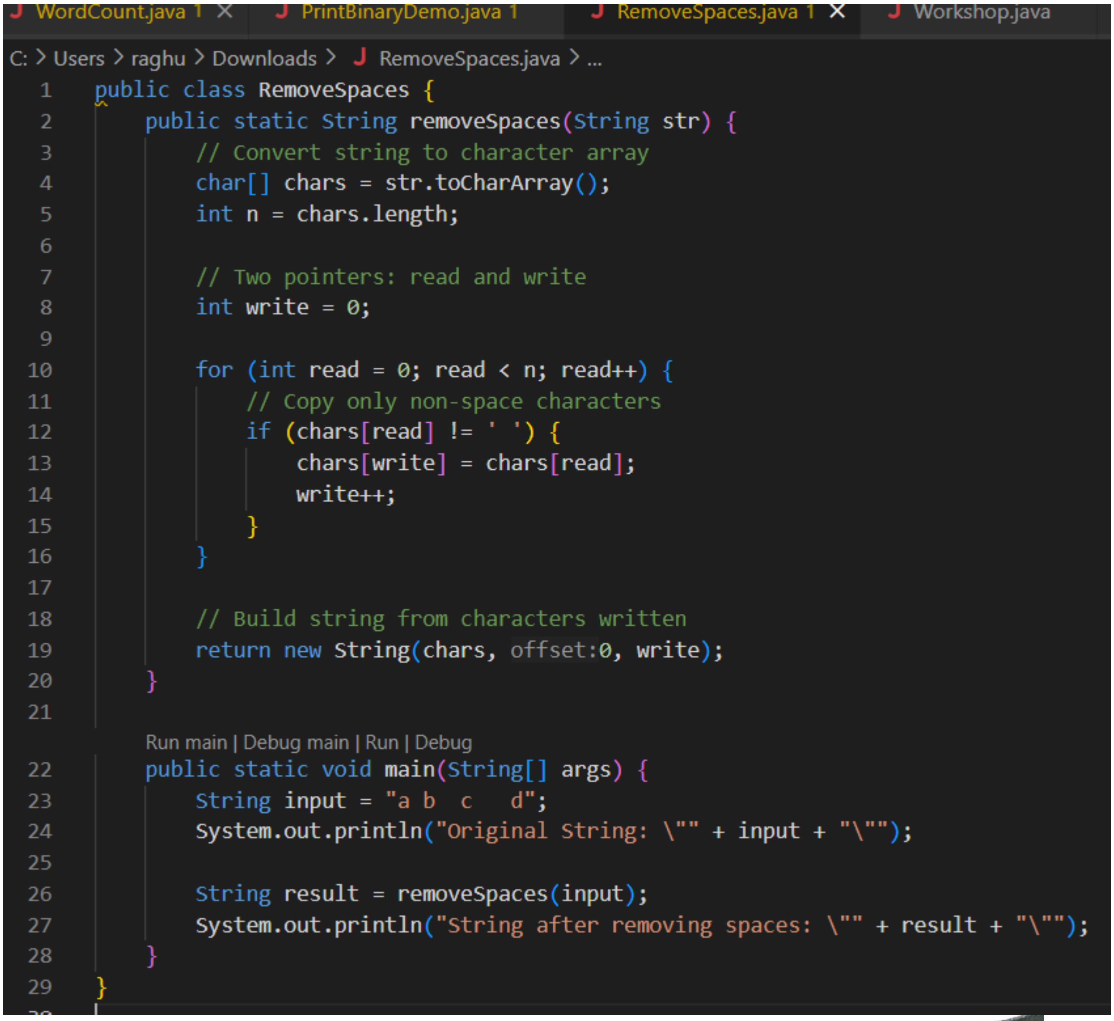
Q) Remove spaces in the string

Using ~~Spaces~~ Read write pointer (two pointer Approach)



Or we can use the New StringBuilder & append characters




WordCount.java 1
PrintBinaryDemo.java 1
RemoveSpaces.java 1
Workshop.java
C: > Users > raghu > Downloads > RemoveSpaces.java > ...
Run main | Debug main | Run | Debug

```java
public class RemoveSpaces {
    public static String removeSpaces(String str) {
        // Convert string to character array
        char[] chars = str.toCharArray();
        int n = chars.length;

        // Two pointers: read and write
        int write = 0;

        for (int read = 0; read < n; read++) {
            // Copy only non-space characters
            if (chars[read] != ' ') {
                chars[write] = chars[read];
                write++;
            }
        }

        // Build string from characters written
        return new String(chars, offset:0, write);
    }

    public static void main(String[] args) {
        String input = "a b  c   d";
        System.out.println("Original String: \"" + input + "\"");

        String result = removeSpaces(input);
        System.out.println("String after removing spaces: \"" + result + "\"");
    }
}
```