

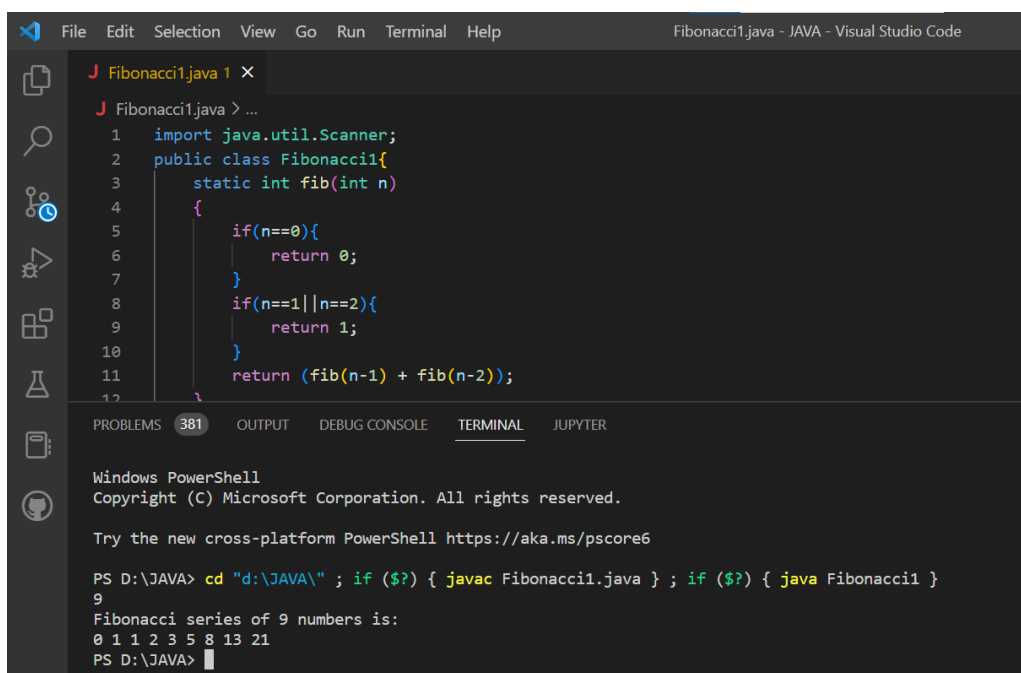
Assignment-1

Code:

i) With Recursion

```
import java.util.Scanner;
public class Fibonacci1{
    static int fib(int n)
    {
        if(n==0){
            return 0;
        }
        if(n==1 || n==2){
            return 1;
        }
        return (fib(n-1) + fib(n-2));
    }
    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        System.out.println("Fibonacci series of "+n+" numbers is:");
        for(int i=0; i<n; i++){
            System.out.print(fib(i)+" ");
        }
    }
}
```

Output:



The screenshot shows the Visual Studio Code editor with the file 'Fibonacci1.java' open. The code is as follows:

```
1 import java.util.Scanner;
2 public class Fibonacci1{
3     static int fib(int n)
4     {
5         if(n==0){
6             return 0;
7         }
8         if(n==1 || n==2){
9             return 1;
10        }
11        return (fib(n-1) + fib(n-2));
12    }
13 }
```

The terminal output shows the execution of the program:

```
PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac Fibonacci1.java } ; if ($?) { java Fibonacci1 }
9
Fibonacci series of 9 numbers is:
0 1 1 2 3 5 8 13 21
PS D:\JAVA>
```

ii) Without Recursion

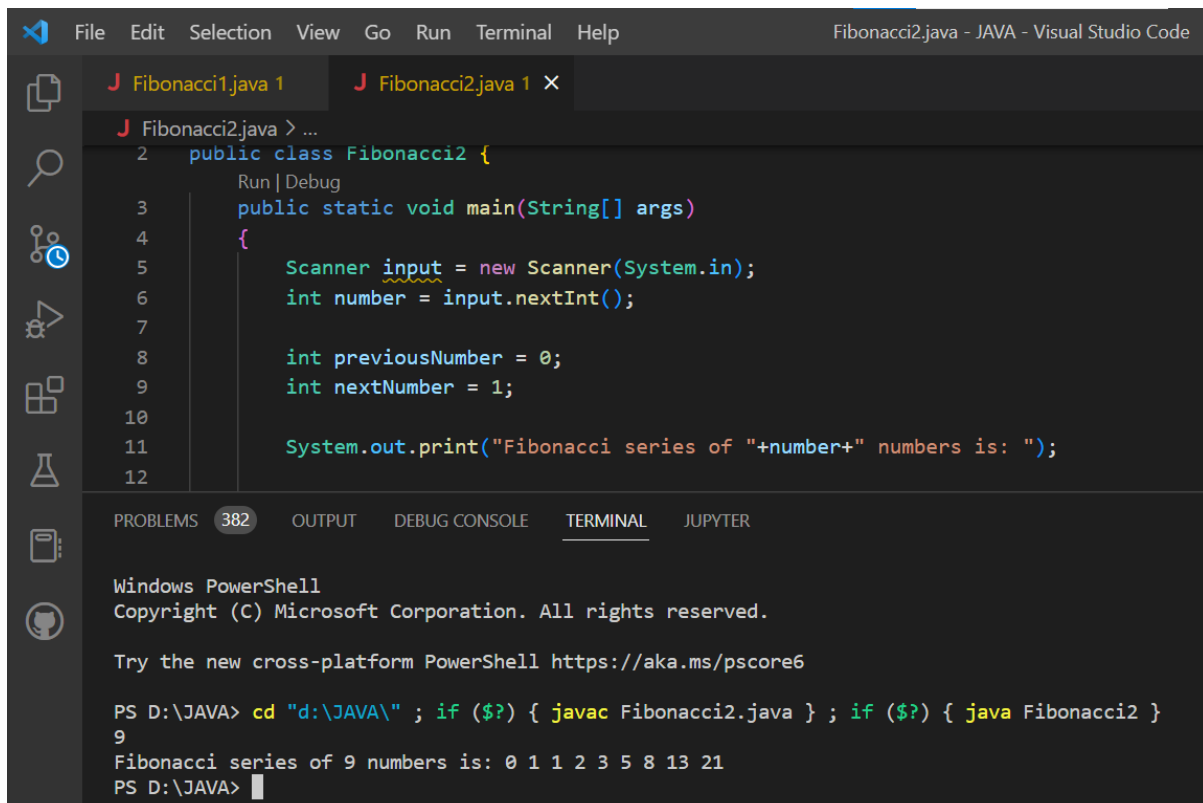
```
import java.util.Scanner;
public class Fibonacci2 {
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        int previousNumber = 0;
        int nextNumber = 1;

        System.out.print("Fibonacci series of "+number+" numbers is: ");

        for (int i = 1; i <= number; ++i){
            System.out.print(previousNumber+" ");
            int sum = previousNumber + nextNumber;
            previousNumber = nextNumber;
            nextNumber = sum;
        }
    }
}
```

Output:



The screenshot shows the Visual Studio Code editor with the file 'Fibonacci2.java' open. The code in the editor matches the provided code block. The bottom panel shows the 'TERMINAL' tab with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac Fibonacci2.java } ; if ($?) { java Fibonacci2 }
9
Fibonacci series of 9 numbers is: 0 1 1 2 3 5 8 13 21
PS D:\JAVA>
```

Assignment-5

Code:

```
import java.util.Scanner;

public class NQueen {

    //number of queens

    private static int N;

    //chessboard

    private static int[][] board = new int[100][100];

    //function to check if the cell is attacked or not

    private static boolean isAttack(int i,int j)

    {

        int k,l;

        //checking if there is a queen in row or column

        for(k=0;k<N;k++)

        {

            if((board[i][k] == 1) || (board[k][j] == 1))

                return true;

        }

        //checking for diagonals

        for(k=0;k<N;k++)

        {

            for(l=0;l<N;l++)

            {

                if(((k+l) == (i+j)) || ((k-l) == (i-j)))

                {

                    if(board[k][l] == 1)

                        return true;

                }

            }

        }

    }

}
```

```
        return false;
    }
    private static boolean nQueen(int n)
    {
        int i,j;
        //if n is 0, solution found
        if(n==0)
            return true;
        for(i=0;i<N;i++){
            for(j=0;j<N;j++){
                /* checking if we can place a queen here or not queen will not be placed
                if the place is being attacked or already occupied */
                if((!isAttack(i,j)) && (board[i][j]!=1)){
                    board[i][j] = 1;
                    if(nQueen(n-1)==true)
                    {
                        return true;
                    }
                    board[i][j] = 0;
                }
            }
        }
        return false;
    }

    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        //taking the value of N
        System.out.println("Enter the value of N for NxN chessboard: ");
        N = input.nextInt();
        int i,j;
        //calling the function
```

```

nQueen(N);

//printing the matix
for(i=0;i<N;i++)
{
    for(j=0;j<N;j++)

        System.out.print(board[i][j]+"t");

    System.out.print("\n");
}
}
}

```

Output:

The screenshot shows the Visual Studio Code editor with the file `NQueen.java` open. The code defines a class `NQueen` with a static integer `N` for the number of queens, a 2D array `board` of size `100x100`, and a static method `isAttack` to check if a cell is attacked. The terminal window at the bottom shows the command prompt where the user has entered `4` for the value of `N`. The output displays a 4x4 chessboard matrix where the queen is placed at (0,1), (1,0), (2,2), and (3,3).

```

NQueen.java > NQueen > isAttack(int, int)
1  import java.util.Scanner;
2  public class NQueen {
3      //number of queens
4      private static int N;
5      //chessboard
6      private static int[][] board = new int[100][100];
7
8      //function to check if the cell is attacked or not
9      private static boolean isAttack(int i,int j)
10     {
11         //if i < 1,

```

PROBLEMS 382 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```

PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac NQueen.java } ; if ($?) { java NQueen }
Enter the value of N for NxN chessboard:
4
0      1      0      0
0      0      0      1
1      0      0      0
0      0      1      0
PS D:\JAVA>

```

Assignment-3

Code:

```
import java.util.Scanner;

class Knapsack
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        int object,m;

        System.out.println("Enter the Total Objects: ");
        object=sc.nextInt();

        int weight[]=new int[object];
        int profit[]=new int[object];
        for(int i=0;i<object;i++)
        {
            System.out.println("Enter the Profit: ");
            profit[i]=sc.nextInt();

            System.out.println("Enter the Weight: ");
            weight[i]=sc.nextInt();
        }

        System.out.println("Enter the Knapsack capacity: ");
        m=sc.nextInt();

        double p_w[]=new double[object];
        for(int i=0;i<object;i++)
        {
            p_w[i]=(double)profit[i]/(double)weight[i];
        }

        System.out.println("");
        System.out.println("-----Dataset-----");
        System.out.println("");
```

```
System.out.print("Objects ");
for(int i=1;i<=object;i++)
{
    System.out.print(i+" ");
}
System.out.println();
System.out.print("Profit ");
for(int i=0;i<object;i++)
{
    System.out.print(profit[i]+" ");
}
System.out.println();
System.out.print("Weight ");
for(int i=0;i<object;i++)
{
    System.out.print(weight[i]+" ");
}
System.out.println();
System.out.print("P/W ");
for(int i=0;i<object;i++)
{
    System.out.print(p_w[i]+" ");
}
for(int i=0;i<object-1;i++)
{
    for(int j=i+1;j<object;j++)
    {
        if(p_w[i]<p_w[j])
        {
            double temp=p_w[j];
            p_w[j]=p_w[i];
```

```
p_w[i]=temp;

int temp1=profit[j];
profit[j]=profit[i];
profit[i]=temp1;

int temp2=weight[j];
weight[j]=weight[i];
weight[i]=temp2;
}
}
}

System.out.println("");
System.out.println("\n--After Arranging--");
System.out.println("");
System.out.print("Objects ");
for(int i=1;i<=object;i++)
{
System.out.print(i+" ");
}

System.out.println();
System.out.print("Profit ");
for(int i=0;i<object;i++)
{
System.out.print(profit[i]+" ");
}

System.out.println();
System.out.print("Weight ");
for(int i=0;i<object;i++)
{
System.out.print(weight[i]+" ");
```



```
}  
System.out.println();  
System.out.print("P/W  ");  
for(int i=0;i<object;i++)  
{  
System.out.print(p_w[i]+" ");  
}  
int k=0;  
double sum=0;  
System.out.println();  
while(m>0)  
{  
if(weight[k]<m)  
{  
sum+=1*profit[k];  
m=m-weight[k];  
}  
else  
{  
double x4=m*profit[k];  
double x5=weight[k];  
double x6=x4/x5;  
sum=sum+x6;  
m=0;  
}  
k++;  
}  
System.out.println("Final Profit = "+sum);  
}  
}
```

Output:

Output Clear

```
java -cp /tmp/8GIB6PJT2 Knapsack
Enter the Total Objects:
3
Enter the Profit:
60
Enter the Weight:
10
Enter the Profit:
100
Enter the Weight:
20
Enter the Profit:
120
Enter the Weight:
30
Enter the Knapsack capacity:
50
-----Dataset-----
Objects 1    2    3
Profit 60   100  120   Weight 10   20   30
P/W    6.0  5.0  4.0

--After Arranging--

Objects 1    2    3
Profit 60   100  120
Weight 10   20   30
P/W    6.0  5.0  4.0
▼ Final Profit = 240.0
```

Assignment-4

Code:

```
class Knapsack {  
    static int max(int a, int b)  
{ return (a > b) ? a : b; }  
  
    // Returns the maximum value that can be put in a knapsack of capacity W  
    static int knapSack(int W, int wt[], int val[], int n)  
{  
        int i, w;  
        int K[][] = new int[n + 1][W + 1];  
  
        // Build table K[][] in bottom up manner  
        for (i = 0; i <= n; i++) {  
            for (w = 0; w <= W; w++) {  
                if (i == 0 || w == 0)  
                    K[i][w] = 0;  
                else if (wt[i - 1] <= w)  
                    K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]);  
                else  
                    K[i][w] = K[i - 1][w];  
            }  
        }  
        return K[n][W];  
    }  
  
    public static void main(String args[])  
    {  
        int val[] = new int[] { 60, 100, 120 };  
        int wt[] = new int[] { 10, 20, 30 };  
        int W = 50;
```

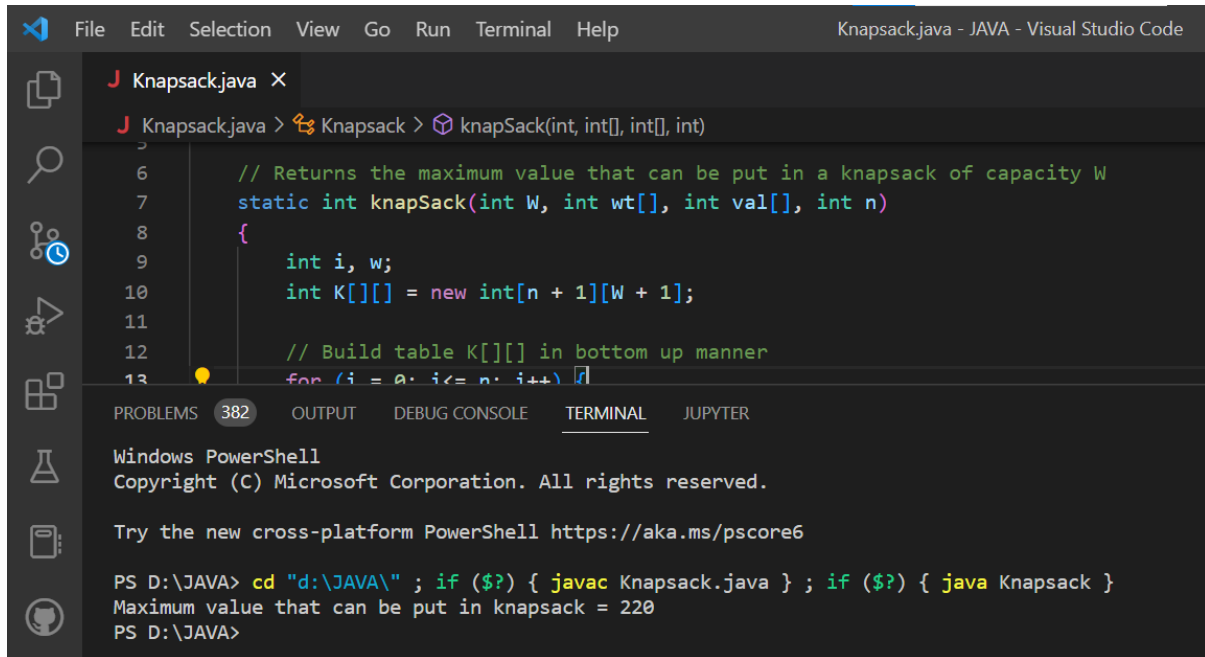
```
int n = val.length;

System.out.println("Maximum value that can be put in knapsack = "+knapSack(W, wt, val, n));

}

}
```

Output:



The screenshot displays the Visual Studio Code interface with the 'Knapsack.java' file open. The code defines a static method `knapSack` that calculates the maximum value for a knapsack of capacity `W` given weights `wt` and values `val` of size `n`. The method uses a dynamic programming table `K` of size `(n+1) x (W+1)` and builds it in a bottom-up manner. The terminal output shows the command `javac Knapsack.java` and `java Knapsack` being executed, resulting in the output: 'Maximum value that can be put in knapsack = 220'.

```
File Edit Selection View Go Run Terminal Help Knapsack.java - JAVA - Visual Studio Code

J Knapsack.java X
J Knapsack.java > Knapsack > knapSack(int, int[], int[], int)

6 // Returns the maximum value that can be put in a knapsack of capacity W
7 static int knapSack(int W, int wt[], int val[], int n)
8 {
9     int i, w;
10    int K[][] = new int[n + 1][W + 1];
11
12    // Build table K[][] in bottom up manner
13    for (i = 0; i <= n; i++) {
```

PROBLEMS 382 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\JAVA> cd "d:\JAVA\" ; if (\$?) { javac Knapsack.java } ; if (\$?) { java Knapsack }
Maximum value that can be put in knapsack = 220
PS D:\JAVA>

Assignment-2

Code:

```
class Node {  
    private String data;  
    private int frequency;  
    private Node left;  
    private Node right;  
    public Node(String element, int freq){  
        data = element;  
        frequency = freq;  
        left = null;  
        right = null;  
    }  
    public void setRightChild(Node n)  
    {  
        right = n;  
    }  
    public void setLeftChild(Node n){  
        left = n;  
    }  
    public Node getRightChild(){  
        return right;  
    }  
    public Node getLeftChild(){  
        return left;  
    }  
    public String getData(){  
        return data;  
    }  
}
```

```
public int getFrequency(){
    return frequency;
}

public static int getLeftChildIndex(int index) {
    if(((2*index) <= MinHeap.heapSize) && (index >= 1)) {
        return 2*index;
    }
    return -1;
}

public static int getRightChildIndex(int index) {
    if((((2*index)+1) <= MinHeap.heapSize) && (index >= 1)) {
        return (2*index)+1;
    }
    return -1;
}

public static int getParentIndex(int index){
    if((index > 1 && (index <= MinHeap.heapSize))) {
        return index/2;
    }
    return -1;
}

public static void inorder(Node root) {
    if(root != null) {
        inorder(root.getLeftChild());
        System.out.print(" "+root.getFrequency()+" ");
        inorder(root.getRightChild());
    }
}

class MinHeap {
    public static int heapSize = 0;
```

```
public static final int heapArraySize = 100;

public static final int INF = 100000;

public static void minHeapify(Node A[], int index) {
    int leftChildIndex = Node.getLeftChildIndex(index);
    int rightChildIndex = Node.getRightChildIndex(index);
    int smallest = index;

    if ((leftChildIndex <= MinHeap.heapSize) && (leftChildIndex>0)) {
        if (A[leftChildIndex].getFrequency() < A[smallest].getFrequency()) {
            smallest = leftChildIndex;
        }
    }

    if ((rightChildIndex <= MinHeap.heapSize) && (rightChildIndex>0)) {
        if (A[rightChildIndex].getFrequency() < A[smallest].getFrequency()) {
            smallest = rightChildIndex;
        }
    }

    // smallest is not the node, node is not a heap
    if (smallest != index) {
        Node temp;
        temp = A[index];
        A[index] = A[smallest];
        A[smallest] = temp;
        minHeapify(A, smallest);
    }
}

class MinQueue {
    public static void insert(Node A[], Node a, int key) {
        MinHeap.heapSize++;
        A[MinHeap.heapSize] = a;
        int index = MinHeap.heapSize;
```

```

while((index>1) && (A[Node.getParentIndex(index)].getFrequency() > a.getFrequency())) {
    Node temp;
    temp = A[index];
    A[index] = A[Node.getParentIndex(index)];
    A[Node.getParentIndex(index)] = temp;
    index = Node.getParentIndex(index);
}
}

public static Node[] buildQueue(Node c[], int size) {
    Node[] a = new Node[MinHeap.heapArraySize];
    for(int i=0; i<size; i++) {
        MinQueue.insert(a, c[i], c[i].getFrequency());
    }
    return a;
}

public static Node extractMin(Node A[]) {
    Node minm = A[1];
    A[1] = A[MinHeap.heapSize];
    MinHeap.heapSize--;
    MinHeap.minHeapify(A, 1);
    return minm;
}
}

class Huffman {
    public static Node greedyHuffmanCode(Node C[]) {
        Node[] minQueue = MinQueue.buildQueue(C, 6);
        while(MinHeap.heapSize > 1) {
            Node h = MinQueue.extractMin(minQueue);
            Node i = MinQueue.extractMin(minQueue);
            Node z = new Node("NONE", h.getFrequency()+i.getFrequency());
            z.setLeftChild(h);

```



```

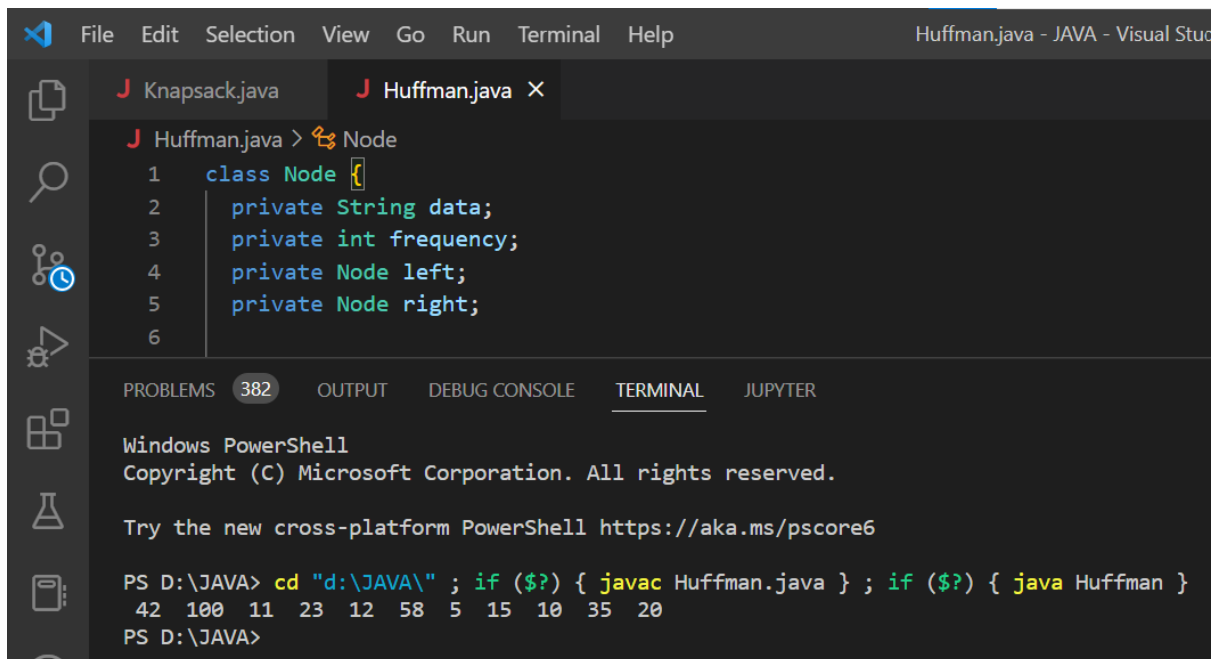
        z.setRightChild(i);

        MinQueue.insert(minQueue, z, z.getFrequency());
    }

    return MinQueue.extractMin(minQueue);
}

public static void main(String[] args) {
    Node a = new Node("a", 42);
    Node b = new Node("b", 20);
    Node c = new Node("c", 5);
    Node d = new Node("d", 10);
    Node e = new Node("e", 11);
    Node f = new Node("f", 12);
    Node[] C = {a, b, c, d, e, f};
    Node z = Huffman.greedyHuffmanCode(C);
    Node.inorder(z);
    System.out.println("");
}
}

```

Output:


The screenshot shows the Visual Studio Code editor with the file `Huffman.java` open. The code defines a `Node` class with attributes `data`, `frequency`, `left`, and `right`. The terminal window at the bottom shows the execution of the program, displaying the frequency of each character in a Huffman tree.

```

class Node {
    private String data;
    private int frequency;
    private Node left;
    private Node right;
}

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

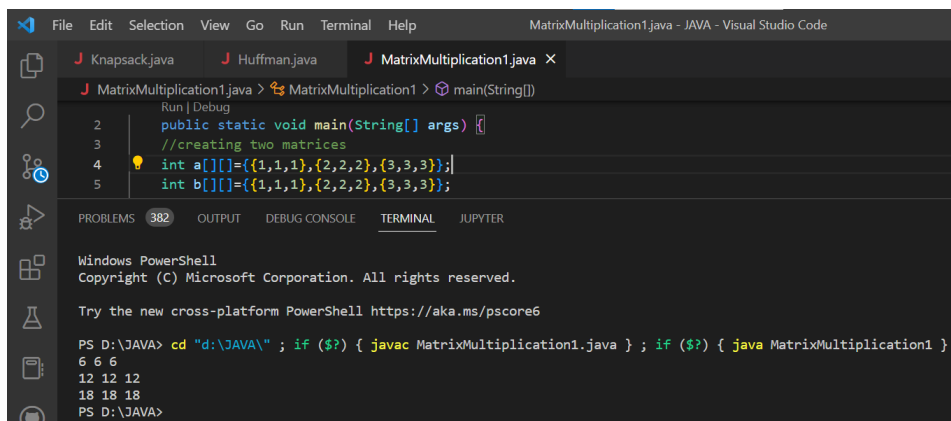
PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac Huffman.java } ; if ($?) { java Huffman }
42 100 11 23 12 58 5 15 10 35 20
PS D:\JAVA>

```

Mini Project

Matrix Multiplication**Code:**

```
public class MatrixMultiplication1{  
    public static void main(String[] args) {  
        //creating two matrices  
        int a[][]={{1,1,1},{2,2,2},{3,3,3}};  
        int b[][]={{1,1,1},{2,2,2},{3,3,3}};  
        int c[][]=new int[3][3];  
        for(int i=0;i<3;i++){  
            for(int j=0;j<3;j++){  
                c[i][j]=0;  
                for(int k=0;k<3;k++){  
                    {  
                        c[i][j]+=a[i][k]*b[k][j];  
                    }  
                System.out.print(c[i][j]+" ");  
            }  
            System.out.println();  
        }  
    }  
}
```

Output:

```
File Edit Selection View Go Run Terminal Help  
MatrixMultiplication1.java - JAVA - Visual Studio Code  
J Knapsack.java J Huffman.java J MatrixMultiplication1.java X  
J MatrixMultiplication1.java > MatrixMultiplication1 > main(String[])  
Run | Debug  
2 public static void main(String[] args) {  
3 //creating two matrices  
4 int a[][]={{1,1,1},{2,2,2},{3,3,3}};  
5 int b[][]={{1,1,1},{2,2,2},{3,3,3}};  
PROBLEMS (382) OUTPUT DEBUG CONSOLE TERMINAL JUPYTER  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac MatrixMultiplication1.java } ; if ($?) { java MatrixMultiplication1 }  
6 6 6  
12 12 12  
18 18 18  
PS D:\JAVA>
```

Multithreading Matrix Multiplication

Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class MatrixMultiplication2 extends Thread{

    static int in1[][];
    static int in2[][];
    static int out[][];
    static int n=2;

    int row;

    MatrixMultiplication2(int i)
    {
        row=i;
        this.start();
    }

    public void run()
    {
        int i,j;
        for(i=0;i<n;i++)
        {
            out[row][i]=0;
            for(j=0;j<n;j++)
            out[row][i]=out[row][i]+in1[row][j]*in2[j][i];
        }
    }

    public static void main(String args[])
    {
        int i,j;

        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

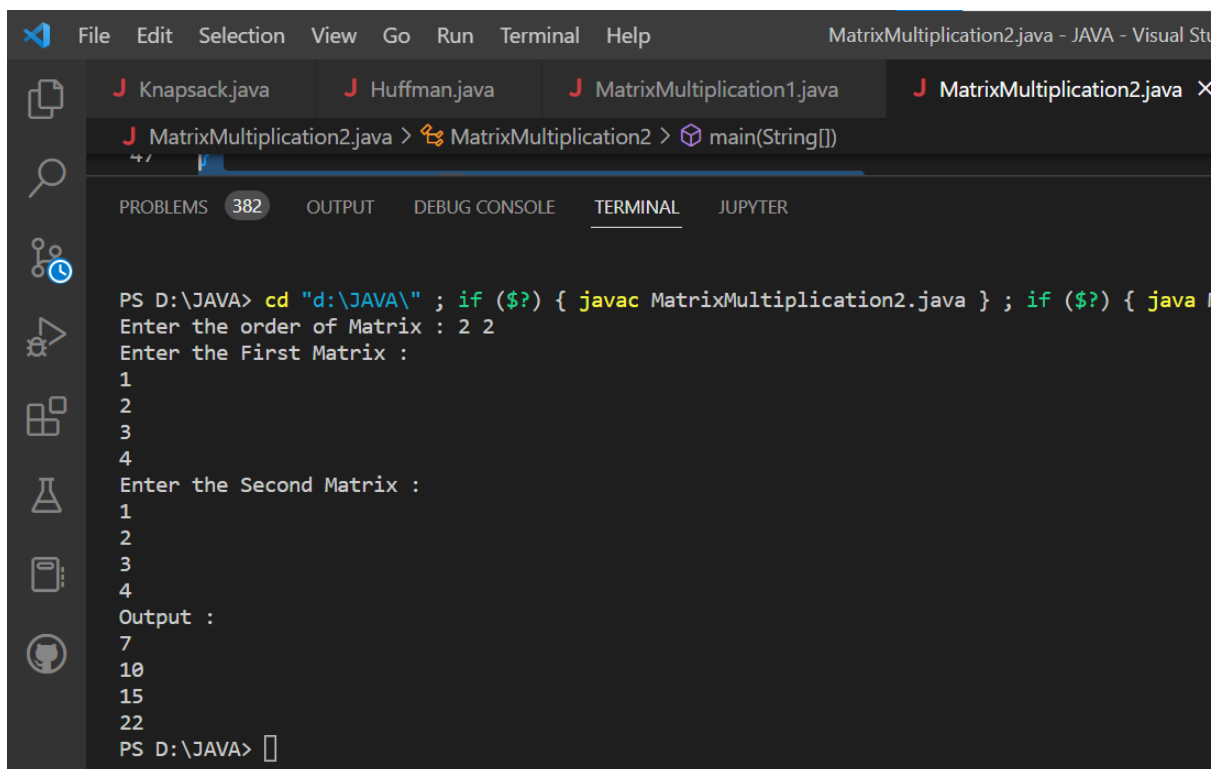
        System.out.print("Enter the order of Matrix : ");

        try
```

```
{
n=Integer.parseInt(br.readLine());
}catch(Exception e){}
in1=new int[n][n];
in2=new int[n][n];
out=new int[n][n];
System.out.println("Enter the First Matrix : ");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
try
{
in1[i][j]=Integer.parseInt(br.readLine());
}catch(Exception e){}
}
}
System.out.println("Enter the Second Matrix : ");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
try
{
in2[i][j]=Integer.parseInt(br.readLine());
}catch(Exception e){}
}
}
MatrixMultiplication2 mat[]=new MatrixMultiplication2[n];
for(i=0;i<n;i++)
mat[i]=new MatrixMultiplication2(i);
```

```
try
{
for(i=0;i<n;i++)
mat[i].join();
}catch(Exception e){}
System.out.println("Output :");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
System.out.println(out[i][j]);
}
}
```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal displays the following text:

```
PS D:\JAVA> cd "d:\JAVA\" ; if ($?) { javac MatrixMultiplication2.java } ; if ($?) { java
Enter the order of Matrix : 2 2
Enter the First Matrix :
1
2
3
4
Enter the Second Matrix :
1
2
3
4
Output :
7
10
15
22
PS D:\JAVA>
```

The terminal output shows the program successfully compiled and executed. It prompts the user to enter the order of the matrix (2 2), then the elements of the first matrix (1, 2, 3, 4), and the elements of the second matrix (1, 2, 3, 4). The resulting output is a single column of numbers: 7, 10, 15, and 22.