**BEST FIT**

package BestFit;

public class BestFit

{

// Method to allocate memory to blocks as per Best fit

// algorithm

static void bestFit(int blockSize[], int m, int processSize[], int n)

{

 // Stores block id of the block allocated to a

 // process

 int allocation[] = new int[n];

 // Initially no block is assigned to any process

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

 allocation[i] = -1;

 // pick each process and find suitable blocks

 // according to its size ad assign to it

 for (int i=0; i<n; i++)

 {

 // Find the best fit block for current process

 int bestIdx = -1;

 for (int j=0; j<m; j++)

 {

 if (blockSize[j] >= processSize[i])

 {

 if (bestIdx == -1)

 bestIdx = j;

 else if (blockSize[bestIdx] > blockSize[j])

     bestIdx = j;

 }

 }

 // If we could find a block for current process

 if (bestIdx != -1)

 {

 // allocate block j to p[i] process

 allocation[i] = bestIdx;

 // Reduce available memory in this block.

 blockSize[bestIdx] -= processSize[i];

 }

 }

 System.out.println("Name :Ketan Devraj");

 System.out.println("Roll No.: TACO22122");

 System.out.println("\nProcess No.\tProcess Size\tBlock no.");

 for (int i = 0; i < n; i++)

 {

 System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t");

 if (allocation[i] != -1)

 System.out.print(allocation[i] + 1);

 else

 System.out.print("Not Allocated");

 System.out.println();

 }

}

// Driver Method

public static void main(String[] args)

{

    int blockSize[] = {100, 500, 200, 300, 600};

     int processSize[] = {212, 417, 112, 426};

     int m = blockSize.length;

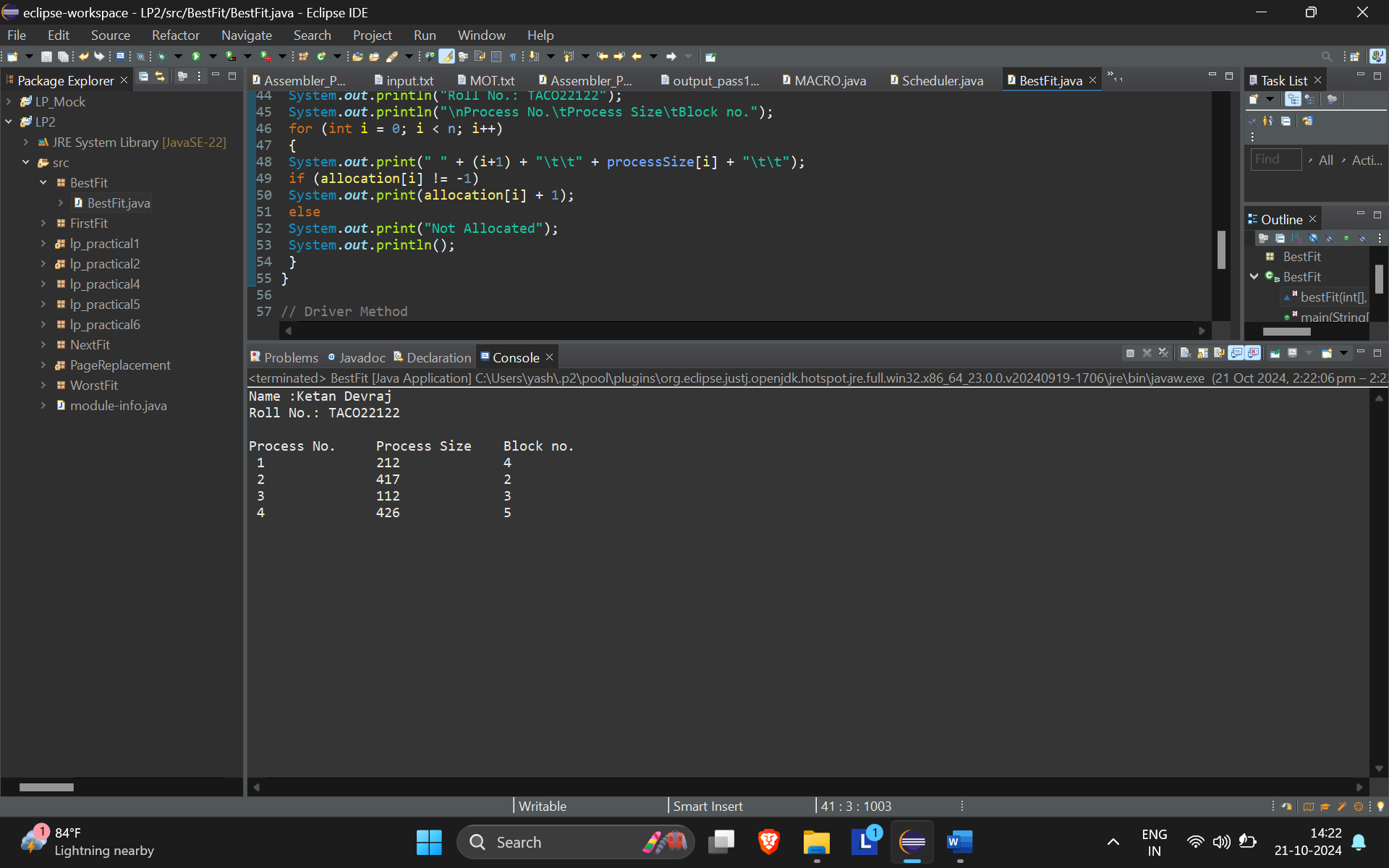
     int n = processSize.length;

     bestFit(blockSize, m, processSize, n);

    }

    }

**OUTPUT**



**FIRST FIT**

package FirstFit;

public class FirstFit

{

 // Method to allocate memory to

 // blocks as per First fit algorithm

 static void firstFit(int blockSize[], int m,

 int processSize[], int n)

 {

 // Stores block id of the

 // block allocated to a process

 int allocation[] = new int[n];

 // Initially no block is assigned to any process

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

 allocation[i] = -1;

 // pick each process and find suitable blocks

 // according to its size ad assign to it

 for (int i = 0; i < n; i++)

 {

 for (int j = 0; j < m; j++)

 {

 if (blockSize[j] >= processSize[i])

 {

 // allocate block j to p[i] process

 allocation[i] = j;

 // Reduce available memory in this block.

 blockSize[j] -= processSize[i];

 break;

 }

 }

 }

 System.out.println("Name :Ketan Devraj ");

 System.out.println("Roll No.: TACO22122 ");

 System.out.println("\nProcess No.\tProcess Size\tBlock no.");

 for (int i = 0; i < n; i++)

 {

 System.out.print(" " + (i+1) + "\t\t" +

 processSize[i] + "\t\t");

 if (allocation[i] != -1)

 System.out.print(allocation[i] + 1);

 else

 System.out.print("Not Allocated");

 System.out.println();

 }

 }

 // Driver Code

 public static void main(String[] args)

 {

 int blockSize[] = {100, 500, 200, 300, 600};

 int processSize[] = {212, 417, 112, 426};

 int m = blockSize.length;

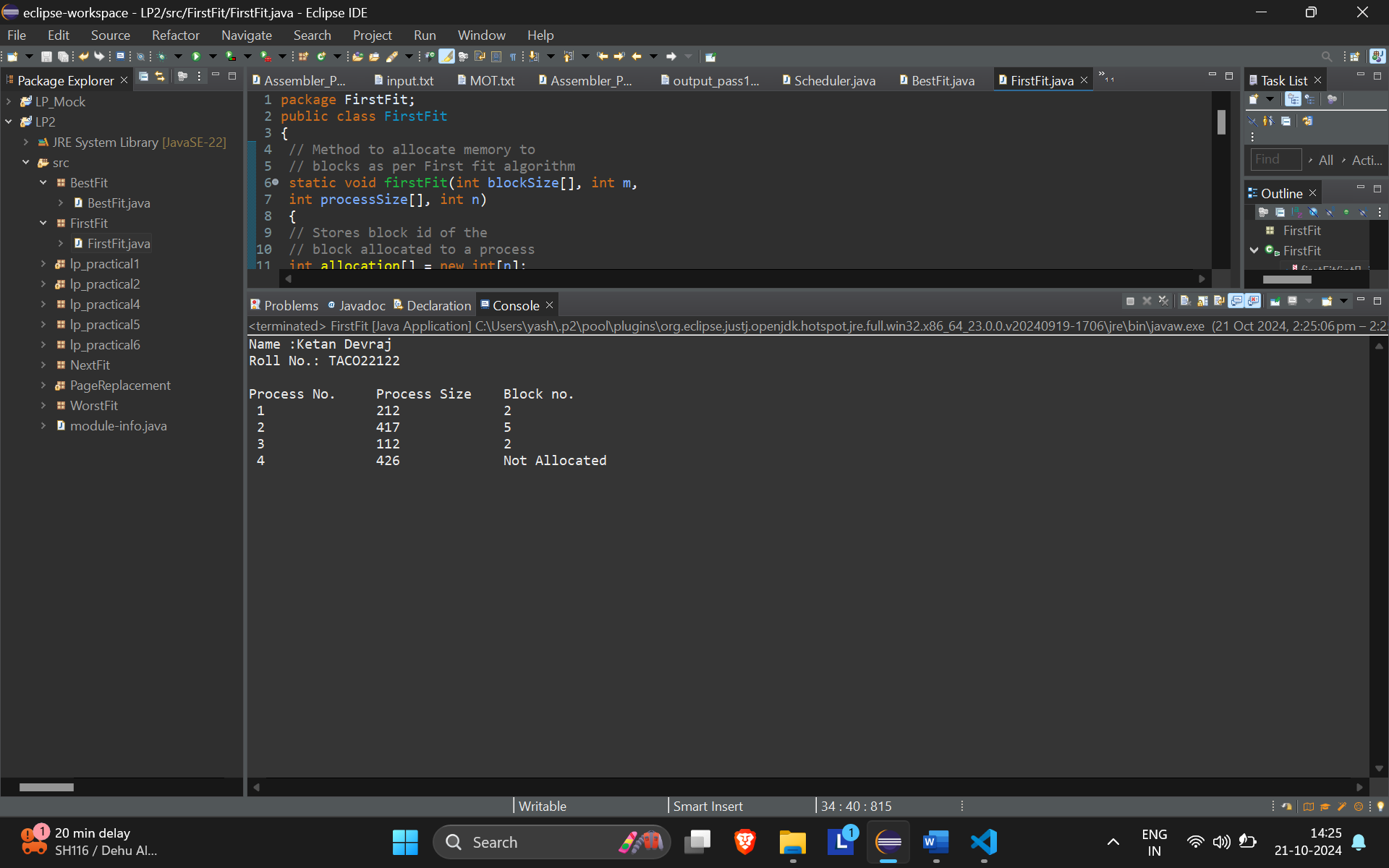
 int n = processSize.length;

 firstFit(blockSize, m, processSize, n);

 }

}

**OUTPUT**



**NEXT FIT**

package NextFit;

import java.util.\*;

public class Next\_fit {

//Function to allocate memory to blocks as per Next fit

//algorithm

static void NextFit(int blockSize[], int m, int processSize[], int n) {

 // Stores block id of the block allocated to a

 // process

 int allocation[] = new int[n], j = 0, t = m - 1;

 // Initially no block is assigned to any process

 Arrays.fill(allocation, -1);

 // pick each process and find suitable blocks

 // according to its size ad assign to it

// pick each process and find sui                    table blocks

//according to its size ad assign to it

for(int i = 0; i < n; i++){

 // Do not start from beginning

 while (j < m){

     if(blockSize[j] >= processSize[i]){

         // allocate block j to p[i] process

         allocation[i] = j;

         // Reduce available memory in this block.

         blockSize[j] -= processSize[i];

         // sets a new end point

         t = (j - 1) % m;

         break;

     }

     if (t == j){

         // sets a new end point

         t = (j - 1) % m;

         // breaks the loop after going through all memory block

         break;

     }

     // mod m will help in traversing the

     // blocks from starting block after

     // we reach the end.

     j = (j + 1) % m;

 }

}

 System.out.println("Ketan Devraj - 22");

 System.out.print("\nProcess No.\tProcess Size\tBlock no.\n");

 for (int i = 0; i < n; i++) {

     System.out.print( i + 1 + "\t\t\t\t" + processSize[i]

             + "\t\t\t\t");

     if (allocation[i] != -1) {

         System.out.print(allocation[i] + 1);

     } else {

         System.out.print("Not Allocated");

     }

     System.out.println("");

 }

}

//Driver program

public static void main(String[] args) {

 int blockSize[] = {5, 10, 20};

 int processSize[] = {10, 20, 5};

 int m = blockSize.length;

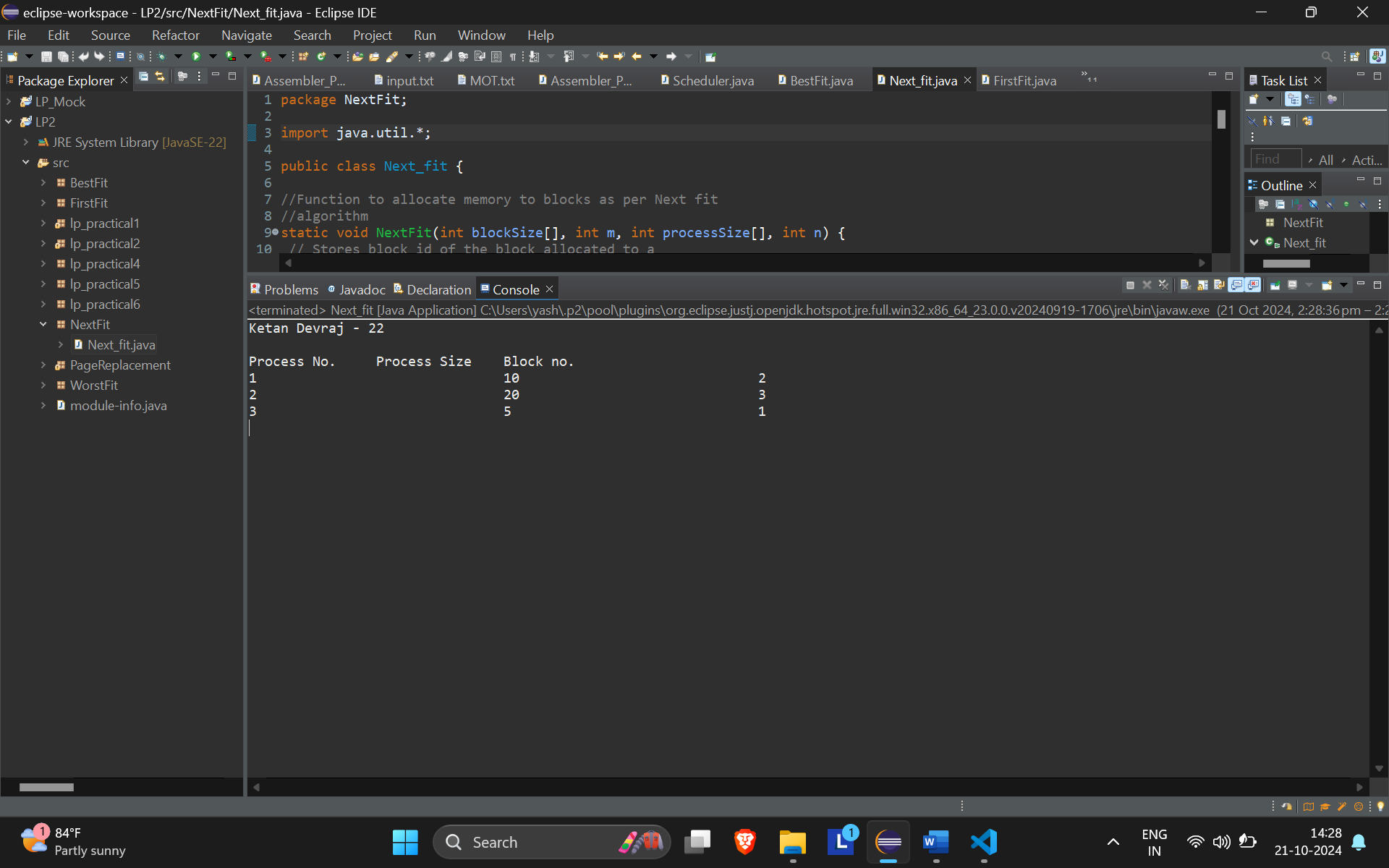
 int n = processSize.length;

 NextFit(blockSize, m, processSize, n);

}

}

**OUTPUT**



**WORST FIT**

package WorstFit;

public class GFG2

{

//Method to allocate memory to blocks as per worst fit

//algorithm

static void worstFit(int blockSize[], int m, int processSize[],

int n)

{

// Stores block id of the block allocated to a

// process

int allocation[] = new int[n];

// Initially no block is assigned to any process

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

allocation[i] = -1;

// pick each process and find suitable blocks

// according to its size ad assign to it

for (int i=0; i<n; i++)

{

// Find the best fit block for current process

int wstIdx = -1;

for (int j=0; j<m; j++)

{

if (blockSize[j] >= processSize[i])

{

if (wstIdx == -1)

    wstIdx = j;

else if (blockSize[wstIdx] < blockSize[j])

wstIdx = j;

}

}

// If we could find a block for current process

if (wstIdx != -1)

{

// allocate block j to p[i] process

allocation[i] = wstIdx;

// Reduce available memory in this block.

blockSize[wstIdx] -= processSize[i];

}

}

System.out.println("Name :KEtan Devraj ");

System.out.println("Roll No.: TACO-22122 ");

System.out.println("\nProcess No.\tProcess Size\tBlock no.");

for (int i = 0; i < n; i++)

{

System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t");

if (allocation[i] != -1)

System.out.print(allocation[i] + 1);

else

System.out.print("Not Allocated");

System.out.println();

}

}

//Driver Method

public static void main(String[] args)

{

 int blockSize[] = {100, 500, 200, 300, 600};

 int processSize[] = {212, 417, 112, 426};

 int m = blockSize.length;

 int n = processSize.length;

 worstFit(blockSize, m, processSize, n);

}

}

**OUTPUT**

