**Memory Placement Stratergies**

1. **First Fit -**

import java.util.Scanner;

public class FirstFit{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of processes: ");

int numP = sc.nextInt();

int arr[] = new int[numP];

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

System.out.println("Enter memory requirement of process " + (i+1)+": " );

arr[i] = sc.nextInt();

}

System.out.println("Enter number of memory blocks: ");

int numB = sc.nextInt();

int memory[] = new int[numB];

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

System.out.println("Enter size of memory block " + (i+1)+": " );

memory[i] = sc.nextInt();

}

System.out.println();

System.out.println("Initial available memory blocks: ");

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

System.out.println(memory[i] + " ");

}

//allocating memory

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

boolean allocated = false;

int allocatedWhere = -1;

for(int j=0; j<numB; j++){

if(memory[j] >= arr[i]){

memory[j] -= arr[i];

allocated = true;

allocatedWhere = j;

break;

}

}

if(allocated){

System.out.println("Memory allocation for Process P" + (i+1) + " is successful in memory block: " + (allocatedWhere + 1));

}

else{

System.out.println("Memory allocation for Process P" + (i + 1 ) + " is unsuccessful ");

}

}

System.out.println();

System.out.println("Fragmented memory blocks: ");

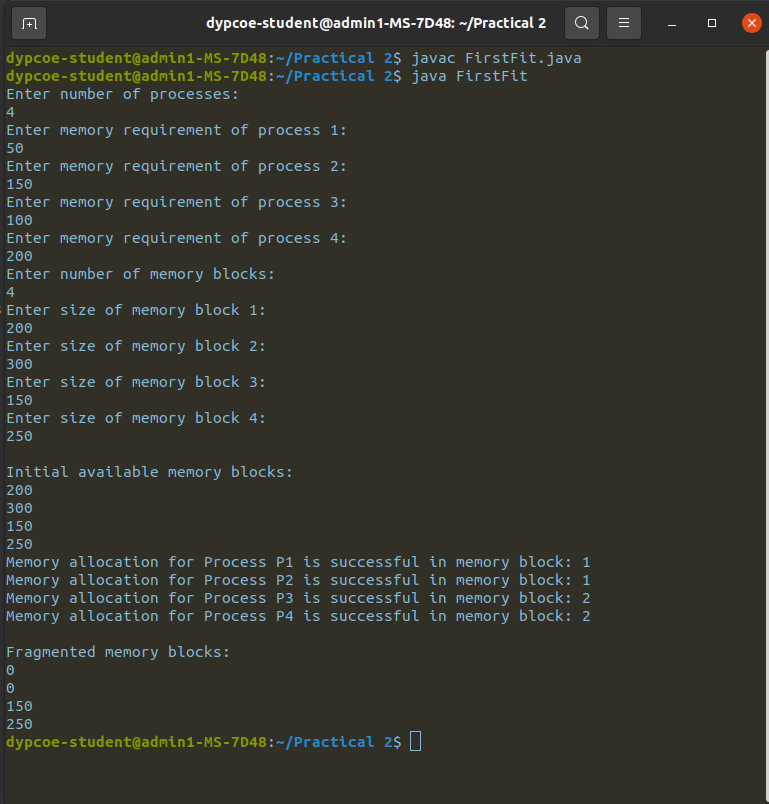
for(int i=0; i<numB; i++){

System.out.println(memory[i] + " ");

}

}

}



1. **Best Fit -**

import java.util.\*;

public class BestFit{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of processes: ");

int numP = sc.nextInt();

int arr[] = new int[numP];

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

System.out.println("Enter memory requirement of process " + (i+1)+": " );

arr[i] = sc.nextInt();

}

//sorting

Arrays.sort(arr);

System.out.println("Enter number of memory blocks: ");

int numB = sc.nextInt();

int memory[] = new int[numB];

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

System.out.println("Enter size of memory block " + (i+1)+": " );

memory[i] = sc.nextInt();

}

System.out.println();

System.out.println("Initial available memory blocks: ");

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

System.out.println(memory[i] + " ");

}

//allocating memory

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

boolean allocated = false;

int allocatedWhere = -1;

for(int j=0; j<numB; j++){

if(memory[j] >= arr[i]){

memory[j] -= arr[i];

allocated = true;

allocatedWhere = j;

break;

}

}

if(allocated){

System.out.println("Memory allocation for Process P" + (i+1) + " is successful in memory block: " + (allocatedWhere + 1));

}

else{

System.out.println("Memory allocation for Process P" + (i + 1 ) + " is unsuccessful ");

}

}

System.out.println();

System.out.println("Fragmented memory blocks: ");

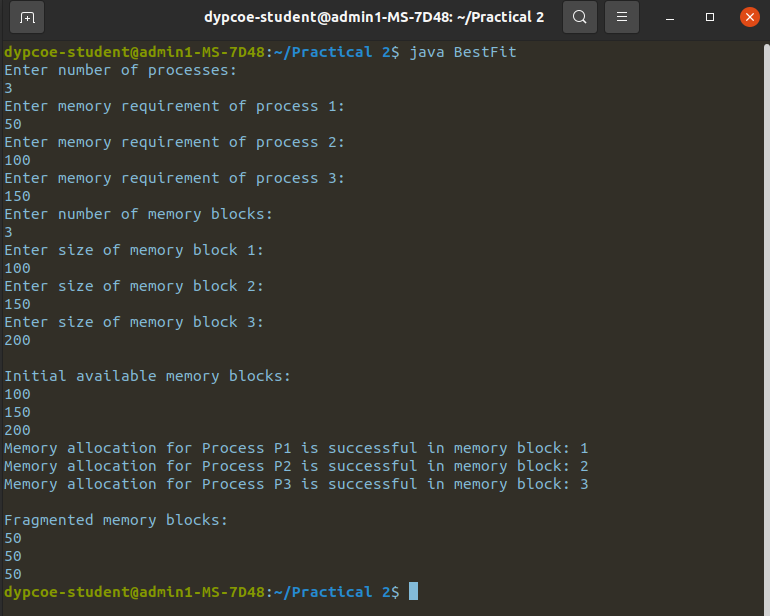
for(int i=0; i<numB; i++){

System.out.println(memory[i] + " ");

}

}

}



1. **Next Fit -**

import java.util.Scanner;

public class NextFit {

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

int[] allocation = new int[n];

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

allocation[i] = -1;

}

int j = 0;

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

int count = 0;

while (count < m) {

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

allocation[i] = j;

blockSize[j] -= processSize[i];

break;

}

j = (j + 1) % m;

count++;

}

}

System.out.println("Process 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.println(allocation[i] + 1);

}

else{

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

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of memory blocks: ");

int m = sc.nextInt();

int[] blockSize = new int[m];

System.out.println("Enter size of each memory block:");

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

blockSize[i] = sc.nextInt();

}

System.out.print("Enter number of processes: ");

int n = sc.nextInt();

int[] processSize = new int[n];

System.out.println("Enter size of each process:");

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

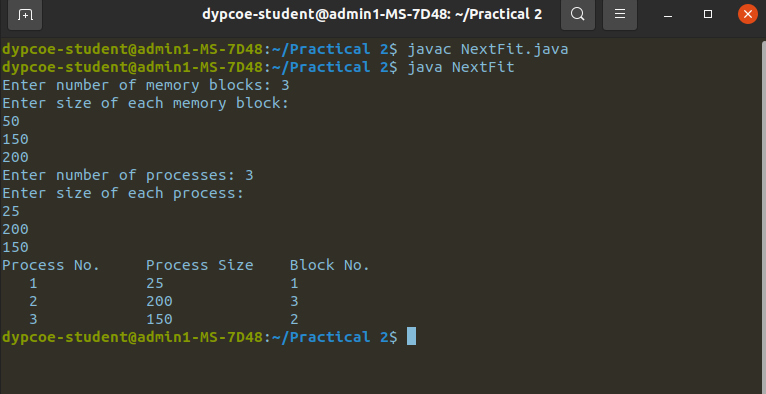
processSize[i] = sc.nextInt();

}

nextFit(blockSize, m, processSize, n);

}

}



1. **Worst Fit -**

import java.util.Scanner;

public class WorstFit{

public static int maxBlock(int[] memory, int sizeReq){

int maxIndex = -1;

int maxSize = -1;

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

if(memory[i] > maxSize && memory[i] >= sizeReq){

maxIndex = i;

maxSize = memory[i];

}

}

return maxIndex;

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of processes: ");

int numP = sc.nextInt();

int arr[] = new int[numP];

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

System.out.println("Enter memory requirement of process " + (i+1)+": " );

arr[i] = sc.nextInt();

}

System.out.println("Enter number of memory blocks: ");

int numB = sc.nextInt();

int memory[] = new int[numB];

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

System.out.println("Enter size of memory block " + (i+1)+": " );

memory[i] = sc.nextInt();

}

System.out.println();

System.out.println("Initial available memory blocks: ");

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

System.out.println(memory[i] + " ");

}

//allocating memory

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

int index = maxBlock(memory, arr[i]);

if(index != -1){

memory[index] -= arr[i];

System.out.println("Memory allocation for Process P" + (i+1) + " is successful in memory block: " + (index+1));

}

else{

System.out.println("Memory allocation for Process P" + (i + 1 ) + " is unsuccessful ");

}

}

System.out.println();

System.out.println("Fragmented memory blocks: ");

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

System.out.println(memory[i] + " ");

}

}

}

