```
package a7;


import java.util.*;

public class MemoryAllocation {

    // Method to allocate memory to blocks as per Next Fit algorithm
    static void nextFit(int blockSize[], int m, int processSize[], int n) {
        int allocation[] = new int[n];
        int j = 0, t = m - 1;
        Arrays.fill(allocation, -1);

        for (int i = 0; i < n; i++) {
            while (j < m) {
                if (blockSize[j] >= processSize[i]) {
                    allocation[i] = j;
                    blockSize[j] -= processSize[i];
                    t = (j - 1) % m;
                    break;
                }
                if (t == j) {
                    t = (j - 1) % m;
                    break;
                }
                j = (j + 1) % m;
            }
        }

        System.out.println("Sakshi Malusare TACO22150");
        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();
        }
    }

    // Method to allocate memory to blocks as per Worst Fit algorithm
    static void worstFit(int blockSize[], int m, int processSize[], int n) {
        int allocation[] = new int[n];
        Arrays.fill(allocation, -1);

        for (int i = 0; i < n; i++) {
            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 (wstIdx != -1) {
            allocation[i] = wstIdx;
            blockSize[wstIdx] -= processSize[i];
        }
    }

    System.out.println("Sakshi Malusare TACO22150");
    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 with menu for user choice
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Define memory blocks and processes
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = blockSize.length;
    int n = processSize.length;

    // Display menu
    System.out.println("Select the memory allocation method:");
    System.out.println("1. Next Fit");
    System.out.println("2. Worst Fit");
    System.out.println("Enter your choice (1/2):");

    int choice = scanner.nextInt();

    // Run selected memory allocation method
    switch (choice) {
        case 1:
            nextFit(blockSize, m, processSize, n);
            break;
        case 2:
            worstFit(blockSize, m, processSize, n);
```

```
                break;
            default:
                System.out.println("Invalid choice. Please select a valid option.");
                break;
        }

        scanner.close();
    }
}
```