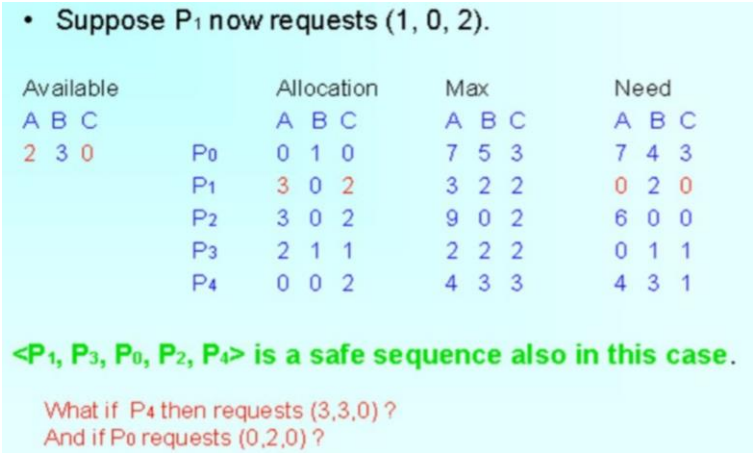
# Operating Systems

## Assignment 8- Banker's Algorithm

1.




**Code:**

BankerAlgo.java

```
import java.util.Scanner;

class BankerAlgo {
    1 usage
    static void findNeedValue(int needArray[][], int maxArray[][], int allocationArray[][],
                              int totalProcess, int totalResources) {
        for (int i = 0; i < totalProcess; i++) {
            for (int j = 0; j < totalResources; j++) {
                needArray[i][j] = maxArray[i][j] - allocationArray[i][j];
            }
        }
    }
    1 usage
    static boolean checkSafeSystem(int processes[], int availableArray[], int maxArray[][],
                                   int allocationArray[][], int totalProcess, int totalResources) {
        int[][] needArray = new int[totalProcess][totalResources];
        findNeedValue(needArray, maxArray, allocationArray, totalProcess, totalResources);
        boolean[] finishProcesses = new boolean[totalProcess];
        int[] safeSequenceArray = new int[totalProcess];
        int[] workArray = new int[totalResources];
        for (int i = 0; i < totalResources; i++)
            workArray[i] = availableArray[i];
        int counter = 0;
        while (counter < totalProcess) {
            boolean foundSafeSystem = false;
            for (int m = 0; m < totalProcess; m++) {
                if (finishProcesses[m] == false) {
                    int j;
                    for (j = 0; j < totalResources; j++)
                        if (needArray[m][j] > workArray[j])
                            break;
```

```
                if (j == totalResources) {
                    for (int k = 0; k < totalResources; k++)
                        workArray[k] += allocationArray[m][k];
                    safeSequenceArray[counter++] = m;
                    finishProcesses[m] = true;
                    foundSafeSystem = true;
                }
            }
        }
        if (foundSafeSystem == false) {
            System.out.print("The system is not in the safe state because lack of resources");
            return false;
        }
    }
    System.out.print("The system is in safe sequence and the sequence is as follows: ");
    for (int i = 0; i < totalProcess; i++)
        System.out.print("P" + safeSequenceArray[i] + " ");
    return true;
}

// main() method start
public static void main(String[] args) {
    int numberOfProcesses, numberOfResources;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter total number of processes");
    numberOfProcesses = sc.nextInt();
    System.out.println("Enter total number of resources");
    numberOfResources = sc.nextInt();
    int processes[] = new int[numberOfProcesses];
    for (int i = 0; i < numberOfProcesses; i++) {
        processes[i] = i;
    }
```

```
    int availableArray[] = new int[numberOfResources];
    for (int i = 0; i < numberOfResources; i++) {
        System.out.println("Enter the availability of resource" + i + ": ");
        availableArray[i] = sc.nextInt();
    }
    int maxArray[][] = new int[numberOfProcesses][numberOfResources];
    for (int i = 0; i < numberOfProcesses; i++) {
        for (int j = 0; j < numberOfResources; j++) {
            System.out.println("Enter the maximum resource" + j + " that can be allocated to process" + i + ": ")
            maxArray[i][j] = sc.nextInt();
        }
    }
    int allocationArray[][] = new int[numberOfProcesses][numberOfResources];
    for (int i = 0; i < numberOfProcesses; i++) {
        for ( int j = 0; j <numberOfResources; j++){
            System.out.println("How many instances of resource"+ j +" are allocated to process"+ i +"? ");
            allocationArray[i][j] = sc.nextInt();
        }
    }
    checkSafeSystem(processes, availableArray, maxArray, allocationArray, numberOfProcesses, numberOfResources);
}
}
```

**Output:**

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program
Enter total number of processes
5
Enter total number of resources
3
Enter the availability of resource0:
3
Enter the availability of resource1:
3
Enter the availability of resource2:
2
Enter the maximum resource0 that can be allocated to process0:
7
Enter the maximum resource1 that can be allocated to process0:
5
Enter the maximum resource2 that can be allocated to process0:
3
Enter the maximum resource0 that can be allocated to process1:
3
Enter the maximum resource1 that can be allocated to process1:
2
Enter the maximum resource2 that can be allocated to process1:
2
Enter the maximum resource0 that can be allocated to process2:
9
Enter the maximum resource1 that can be allocated to process2:
0
Enter the maximum resource2 that can be allocated to process2:
2
Enter the maximum resource0 that can be allocated to process3:
2
Enter the maximum resource1 that can be allocated to process3:
2
```

```
Enter the maximum resource2 that can be allocated to process3:
2
Enter the maximum resource0 that can be allocated to process4:
4
Enter the maximum resource1 that can be allocated to process4:
3
Enter the maximum resource2 that can be allocated to process4:
3
How many instances of resource0 are allocated to process0?
0
How many instances of resource1 are allocated to process0?
1
How many instances of resource2 are allocated to process0?
0
How many instances of resource0 are allocated to process1?
2
How many instances of resource1 are allocated to process1?
0
How many instances of resource2 are allocated to process1?
0
How many instances of resource0 are allocated to process2?
3
How many instances of resource1 are allocated to process2?
0
How many instances of resource2 are allocated to process2?
2
How many instances of resource0 are allocated to process3?
2
How many instances of resource1 are allocated to process3?
1
How many instances of resource2 are allocated to process3?
1
```

```
How many instances of resource0 are allocated to process4?
0
How many instances of resource1 are allocated to process4?
0
How many instances of resource2 are allocated to process4?
2
The system is in safe sequence and the sequence is as follows: P1 P3 P4 P0 P2
Process finished with exit code 0
```

**Submitted By:**

Harshita Pasupuleti

21BCE8421