Ain Shams University / Faculty of Engineering

CSE Department

Operating Systems CSE223

Banker's Algorithm

Submitted By

Omar Magdy Shaaban 16P6034

Group 2 Section 1

Submitted To

Dr. Gamal Abdel Shafy

Cairo, 2018

Abstract

This project implements Banker’s algorithm in choosing a sequence of certain processes that satisfies safety. An unsafe state could lead to a deadlock, which is the main purpose that Banker made his algorithm in order to avoid its occurrence. We will talk about some points concerning the code and provide some test cases and their outputs. The program is implemented in Java and developed using NetBeans IDE.

Contents

1. Code

```
package bankerv12;

import java.util.*;

public class Bankerv12 {

    private static int resourcesTypes;
    private static int numberOfProcesses;
    private static int[] instancesPerResource;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of resource types:");
        resourcesTypes = sc.nextInt();
        instancesPerResource = new int[resourcesTypes];

        System.out.println("Enter number of resources instances for each type:");
        for (int i = 0; i < instancesPerResource.length; i++) {
            instancesPerResource[i] = sc.nextInt();
        }
        System.out.println("Enter number of processes:");
        numberOfProcesses = sc.nextInt();
        System.out.println("");

        Process ps[] = new Process[numberOfProcesses];
        Random r = new Random();
        ArrayList<Process> processesList = new ArrayList();
```

```
boolean terminated;

int[] copyOfInstancesPerResource =  new int[resourcesTypes];
System.arraycopy(instancesPerResource, 0, copyOfInstancesPerResource, 0,
        resourcesTypes);

for (int i = 0; i < numberOfProcesses; i++) {
    ps[i] = createProcess(i, copyOfInstancesPerResource);
    processesList.add(ps[i]);
    System.out.println(ps[i]);
}

int numberOfArrayListElements = processesList.size();
System.out.println("Now available: ");
for (int i = 0; i < resourcesTypes - 1; i++) {
    System.out.print(instancesPerResource[i] + " ");
}
System.out.println(instancesPerResource[resourcesTypes - 1]);

do {
    Process p = selectProcess(processesList);
    if (p == null) {
        System.out.println("Unsafe state. All processes requests are denied. "
                + "Exiting now");
        System.exit(0);
    } else {
        System.out.println("Process P" + p.getNumberOfProcess() + " is executing");
        System.out.println(p);
        String s = "";
```

```
for (int i = 0; i < resourcesTypes - 1; i++) {
    s += instancesPerResource[i] + " ";
}
s += instancesPerResource[resourcesTypes - 1];
System.out.println("Now available: " + s);
int allocated[] = p.getAllocatedResourcesInstances();
int[] previousAllocation = new int[resourcesTypes];
int[] currentAllocation = new int [resourcesTypes];
int retrievedAllocation;
for (int i = 0; i < resourcesTypes; i++) {
    previousAllocation[i] = allocated[i];
    currentAllocation[i] = r.nextInt(allocated[i] + 1);
    retrievedAllocation = previousAllocation[i] - currentAllocation[i];
    instancesPerResource[i] += retrievedAllocation;
}
p.setAllocatedResourcesInstances(currentAllocation);
terminated = checkProcessNeeds(p);
if (terminated) {
    numberOfArrayListElements--;
    processesList.remove(p);
    for(int i = 0; i < resourcesTypes; i++){
        instancesPerResource[i] += currentAllocation[i];
    }
} else {
    processesList.remove(p);
    p = makeNewRequest(p);
    processesList.add(p);
}
System.out.println("New Request for P" + p.getNumberOfProcess() + "\t" + p);
```

```
            s = "";
            for (int i = 0; i < resourcesTypes - 1; i++) {
                s += instancesPerResource[i] + " ";
            }
            s += instancesPerResource[resourcesTypes - 1];
            System.out.println("After Release: " + s);
        }
    } while (numberOfArrayListElements > 0);

}

public static Process selectProcess(ArrayList al) {
    Process p = (Process) al.get(0);
    boolean safe = true;
    for (int i = 0; i < al.size(); i++) {
        p = (Process) al.get(i);
        int allocated[] = p.getAllocatedResourcesInstances();
        int needed[] = p.getNeededResourcesInstances();
        int requested[] = p.getRequestedResourcesInstances();
        for (int j = 0; j < resourcesTypes; j++) {
            if (requested[j] > instancesPerResource[j]) {
                safe = false;
                break;
            }
        }
        if (!safe) {
            System.out.println("Request for P" + p.getNumberOfProcess() + " is denied");
            if(i == al.size() - 1){
                safe = false;
            }
            else{
            safe = true;
            }
        } else {
            for (int k = 0; k < resourcesTypes; k++) {
                needed[k] -= requested[k];
                instancesPerResource[k] -= requested[k];
                allocated[k] += requested[k];
                requested[k] = 0;
            }
            p.setAllocatedResourcesInstances(allocated);
            p.setNeededResourcesInstances(needed);
            p.setRequestedResourcesInstances(requested);
            break;
        }
    }
    if (safe) {
        return p;
    } else {
        return null;
    }
}

public static boolean checkProcessNeeds(Process p) {
    int[] needed = p.getNeededResourcesInstances();
    for (int i = 0; i < needed.length; i++) {
        if (needed[i] != 0) {
            return false;
        }
    }
    return true;
}

public static Process makeNewRequest(Process p) {
    int[] requested = p.getRequestedResourcesInstances();
    int[] needed = p.getNeededResourcesInstances();
    Random r = new Random();
    for (int i = 0; i < needed.length; i++) {
        requested[i] = r.nextInt(needed[i] + 1);
    }
    p.setRequestedResourcesInstances(requested);
    return p;
}

public static Process createProcess(int i, int[] copyOfInstancesPerResource){
    Process p;
    int allocatedResources[] = new int[resourcesTypes];
    int maxResources[] = new int[resourcesTypes];
    int neededResources[] = new int[resourcesTypes];
    int requestedResources[] = new int[resourcesTypes];
    Random r = new Random();

    for (int j = 0; j < resourcesTypes; j++) {
            allocatedResources[j] = r.nextInt(instancesPerResource[j] / 2);
            maxResources[j] = allocatedResources[j]
                    + r.nextInt(copyOfInstancesPerResource[j] - 2 *
                            allocatedResources[j] + 1);
```

```
                neededResources[j] = maxResources[j] - allocatedResources[j];
                requestedResources[j] = r.nextInt(neededResources[j] + 1);
                instancesPerResource[j] -= allocatedResources[j];
            }
        p = new Process(i, allocatedResources, maxResources, neededResources,
                requestedResources);
        return p;
    }
}

class Process {

    private int numberOfProcess;
    private int[] allocatedResourcesInstances;
    private int[] maxResourcesInstances;
    private int[] neededResourcesInstances;
    private int[] requestedResourcesInstances;

    public Process(int numberOfProcess, int[] allocatedResourcesInstances,
            int[] maxResourcesInstances, int[] neededResourcesInstances,
            int[] requestedResourcesInstances) {
        this.numberOfProcess = numberOfProcess;
        this.allocatedResourcesInstances = allocatedResourcesInstances;
        this.maxResourcesInstances = maxResourcesInstances;
        this.neededResourcesInstances = neededResourcesInstances;
        this.requestedResourcesInstances = requestedResourcesInstances;
    }

    public int getNumberOfProcess() {
```

```
    return numberOfProcess;
}

public int[] getAllocatedResourcesInstances() {
    return allocatedResourcesInstances;
}

public int[] getMaxResourcesInstances() {
    return maxResourcesInstances;
}

public int[] getNeededResourcesInstances() {
    return neededResourcesInstances;
}

public int[] getRequestedResourcesInstances() {
    return requestedResourcesInstances;
}

public void setNumberOfProcess(int numberOfProcess) {
    this.numberOfProcess = numberOfProcess;
}

public void setAllocatedResourcesInstances(int[] allocatedResourcesInstances) {
    this.allocatedResourcesInstances = allocatedResourcesInstances;
}

public void setMaxResourcesInstances(int[] maxResourcesInstances) {
    this.maxResourcesInstances = maxResourcesInstances;
```

```
}

public void setNeededResourcesInstances(int[] neededResourcesInstances) {
   this.neededResourcesInstances = neededResourcesInstances;
}

public void setRequestedResourcesInstances(int[] requestedResourcesInstances) {
   this.requestedResourcesInstances = requestedResourcesInstances;
}

@Override
public String toString() {
   String s = "";
   for (int i = 0; i < allocatedResourcesInstances.length - 1; i++) {
      s = s + allocatedResourcesInstances[i] + " ";
   }
   s += allocatedResourcesInstances[allocatedResourcesInstances.length - 1];
   s += "\t";
   for (int i = 0; i < maxResourcesInstances.length - 1; i++) {
      s = s + maxResourcesInstances[i] + " ";
   }
   s += maxResourcesInstances[maxResourcesInstances.length - 1];
   s += "\t";
   for (int i = 0; i < neededResourcesInstances.length - 1; i++) {
      s = s + neededResourcesInstances[i] + " ";
   }
   s += neededResourcesInstances[neededResourcesInstances.length - 1];
   s += "\t";
   for (int i = 0; i < requestedResourcesInstances.length - 1; i++) {
```

```
            s = s + requestedResourcesInstances[i] + " ";
        }
        s += requestedResourcesInstances[requestedResourcesInstances.length - 1];
        return s;
    }
}
```

We made some attributes within the main class, to enable modifying it in other functions other than the main function (like “selectProcess”, “createProcess”).

2. Test Cases and Outputs

1) 3, 15 10 15, 5

```
Output: Enter number of resource types:
3
Enter number of resources instances for each type:
15
10
15
Enter number of processes:
5


1 3 4    9 6 7    8 3 3    4 3 1
0 2 3    4 4 4    4 2 1    4 1 0
0 0 0    5 8 7    5 8 7    2 6 7
4 0 1    4 0 2    0 0 1    0 0 0
2 1 0    10 7 13 8 6 13   6 1 9
Now available:
8 4 7
Process P0 is executing
```

```
5 6 5     9 6 7     4 0 2     0 0 0

Now available: 4 1 6

New Request for P0       3 4 5     9 6 7     4 0 2     1 0 2

After Release: 6 3 6

Process P1 is executing

4 3 3     4 4 4     0 1 1     0 0 0

Now available: 2 2 6

New Request for P1       4 3 0     4 4 4     0 1 1     0 1 0

After Release: 2 2 9

Request for P2 is denied

Process P3 is executing

4 0 1     4 0 2     0 0 1     0 0 0

Now available: 2 2 9

New Request for P3       0 0 1     4 0 2     0 0 1     0 0 1

After Release: 6 2 9

Request for P2 is denied

Process P4 is executing

8 2 9     10 7 13   2 5 4   0 0 0

Now available: 0 1 0

New Request for P4       1 0 5     10 7 13   2 5 4   0 3 3

After Release: 7 3 4

Request for P2 is denied

Process P0 is executing

4 4 7     9 6 7     3 0 0     0 0 0

Now available: 6 3 2

New Request for P0       3 4 7     9 6 7     3 0 0     0 0 0

After Release: 7 3 2

Request for P2 is denied

Process P1 is executing
```

```
4 4 0     4 4 4     0 0 1     0 0 0

Now available: 7 2 2

New Request for P1       3 2 0     4 4 4     0 0 1     0 0 0

After Release: 8 4 2

Request for P2 is denied

Process P3 is executing

0 0 2     4 0 2     0 0 0     0 0 0

Now available: 8 4 1

New Request for P3       0 0 2     4 0 2     0 0 0     0 0 0

After Release: 8 4 3

Request for P2 is denied

Process P4 is executing

1 3 8     10 7 13 2 2 1     0 0 0

Now available: 8 1 0

New Request for P4       1 0 0     10 7 13   2 2 1    2 2 0

After Release: 8 4 8

Request for P2 is denied

Process P0 is executing

3 4 7     9 6 7     3 0 0     0 0 0

Now available: 8 4 8

New Request for P0       3 0 0     9 6 7     3 0 0     1 0 0

After Release: 8 8 15

Process P2 is executing

2 6 7     5 8 7     3 2 0     0 0 0

Now available: 6 2 8

New Request for P2       1 5 1     5 8 7     3 2 0     3 1 0

After Release: 7 3 14

Process P1 is executing

3 2 0     4 4 4     0 0 1     0 0 0
```

```
Now available: 7 3 14

New Request for P1       3 2 0     4 4 4     0 0 1     0 0 0

After Release: 7 3 14

Process P4 is executing

3 2 0     10 7 13   0 0 1    0 0 0

Now available: 5 1 14

New Request for P4       2 0 0     10 7 13   0 0 1     0 0 1

After Release: 6 3 14

Process P0 is executing

4 0 0     9 6 7     2 0 0     0 0 0

Now available: 5 3 14

New Request for P0       0 0 0     9 6 7     2 0 0     2 0 0

After Release: 9 3 14

Process P2 is executing

4 6 1     5 8 7     0 1 0     0 0 0

Now available: 6 2 14

New Request for P2       4 4 0     5 8 7     0 1 0     0 1 0

After Release: 6 4 15

Process P1 is executing

3 2 0     4 4 4     0 0 1     0 0 0

Now available: 6 4 15

New Request for P1       0 1 0     4 4 4     0 0 1     0 0 1

After Release: 9 5 15

Process P4 is executing

2 0 1     10 7 13   0 0 0    0 0 0

Now available: 9 5 14

New Request for P4       1 0 1     10 7 13   0 0 0     0 0 0

After Release: 11 5 15

Process P0 is executing
```

```
2 0 0     9 6 7     0 0 0     0 0 0

Now available: 9 5 15

New Request for P0       1 0 0     9 6 7     0 0 0     0 0 0

After Release: 11 5 15

Process P2 is executing

4 5 0     5 8 7     0 0 0     0 0 0

Now available: 11 4 15

New Request for P2       4 2 0     5 8 7     0 0 0     0 0 0

After Release: 15 9 15

Process P1 is executing

0 1 1     4 4 4     0 0 0     0 0 0

Now available: 15 9 14

New Request for P1       0 0 1     4 4 4     0 0 0     0 0 0

After Release: 15 10 15
```

2) 3, 20 14 18, 4

```
Output: Enter number of resource types:

3

Enter number of resources instances for each type:

20

18

14

Enter number of processes:

4


9 2 6     11 16 6   2 14 0   1 1 0

3 2 2     3 2 10     0 0 8     0 0 8

3 1 0     9 6 12     6 5 12   0 2 5

1 0 1     11 18 9   10 18 8  7 4 5

Now available:
```

```
4 13 5

Process P0 is executing

10 3 6    11 16 6   1 13 0   0 0 0

Now available: 3 12 5

New Request for P0        5 1 3      11 16 6   1 13 0   1 3 0

After Release: 8 14 8

Process P1 is executing

3 2 10    3 2 10    0 0 0      0 0 0

Now available: 8 14 0

New Request for P1        2 2 0      3 2 10    0 0 0      0 0 0

After Release: 11 16 10

Process P2 is executing

3 3 5      9 6 12    6 3 7      0 0 0

Now available: 11 14 5

New Request for P2        1 0 1      9 6 12    6 3 7      0 3 2

After Release: 13 17 9

Process P3 is executing

8 4 6      11 18 9   3 14 3   0 0 0

Now available: 6 13 4

New Request for P3        6 3 5      11 18 9   3 14 3   2 5 1

After Release: 8 14 5

Process P0 is executing

6 4 3      11 16 6   0 10 0   0 0 0

Now available: 7 11 5

New Request for P0        6 4 1      11 16 6   0 10 0   0 8 0

After Release: 7 11 7

Process P2 is executing

1 3 3      9 6 12    6 0 5      0 0 0

Now available: 7 8 5
```

```
New Request for P2        1 2 1      9 6 12    6 0 5      0 0 1

After Release: 7 9 7

Process P3 is executing

8 8 6      11 18 9    1 9 2    0 0 0

Now available: 5 4 6

New Request for P3        8 5 2      11 18 9    1 9 2    1 4 1

After Release: 5 7 10

Request for P0 is denied

Process P2 is executing

1 2 2      9 6 12    6 0 4      0 0 0

Now available: 5 7 9

New Request for P2        0 0 1      9 6 12    6 0 4      0 0 4

After Release: 6 9 10

Process P0 is executing

6 12 1    11 16 6    0 2 0    0 0 0

Now available: 6 1 10

New Request for P0        0 10 1    11 16 6    0 2 0    0 1 0

After Release: 12 3 10

Request for P3 is denied

Process P2 is executing

0 0 5      9 6 12    6 0 0      0 0 0

Now available: 12 3 6

New Request for P2        0 0 3      9 6 12    6 0 0      0 0 0

After Release: 12 3 8

Request for P3 is denied

Process P0 is executing

0 11 1    11 16 6    0 1 0    0 0 0

Now available: 12 2 8

New Request for P0        0 10 0    11 16 6    0 1 0    0 1 0
```

```
After Release: 12 3 9

Request for P3 is denied

Process P2 is executing

0 0 3     9 6 12    6 0 0     0 0 0

Now available: 12 3 9

New Request for P2       0 0 0     9 6 12    6 0 0     0 0 0

After Release: 12 3 12

Request for P3 is denied

Process P0 is executing

0 11 0    11 16 6   0 0 0    0 0 0

Now available: 12 2 12

New Request for P0       0 10 0    11 16 6   0 0 0    0 0 0

After Release: 12 13 12

Process P3 is executing

9 9 3     11 18 9   0 5 1    0 0 0

Now available: 11 9 11

New Request for P3       6 4 2     11 18 9   0 5 1    0 2 1

After Release: 14 14 12

Process P2 is executing

0 0 0     9 6 12    6 0 0     0 0 0

Now available: 14 14 12

New Request for P2       0 0 0     9 6 12    6 0 0     6 0 0

After Release: 14 14 12

Process P3 is executing

6 6 3     11 18 9   0 3 0    0 0 0

Now available: 14 12 11

New Request for P3       0 3 2     11 18 9   0 3 0    0 3 0

After Release: 20 15 12

Process P2 is executing
```

```
6 0 0     9 6 12   0 0 0     0 0 0

Now available: 14 15 12

New Request for P2       3 0 0     9 6 12   0 0 0     0 0 0

After Release: 20 15 12

Process P3 is executing

0 6 2     11 18 9   0 0 0   0 0 0

Now available: 20 12 12

New Request for P3       0 4 2     11 18 9   0 0 0   0 0 0

After Release: 20 18 14
```

3) 3, 20 25 22, 4

```
Output: Enter number of resource types:

3

Enter number of resources instances for each type:

20

25

22

Enter number of processes:

4


1 8 9     13 11 11          12 3 2   5 0 2

3 1 4     17 17 18          14 16 14          13 4 3

6 7 2     14 10 15          8 3 13   8 1 6

1 0 0     12 16 22          11 16 22          4 5 8

Now available:

9 9 7

Process P0 is executing

6 8 11   13 11 11          7 3 0     0 0 0

Now available: 4 9 5

New Request for P0       6 0 8     13 11 11          7 3 0     2 2 0
```

```
After Release: 4 17 8

Request for P1 is denied

Request for P2 is denied

Process P3 is executing

5 5 8     12 16 22          7 11 14 0 0 0

Now available: 0 12 0

New Request for P3      4 2 0     12 16 22          7 11 14 3 6 1

After Release: 1 15 8

Request for P1 is denied

Request for P2 is denied

Request for P0 is denied

Request for P3 is denied

Unsafe state. All processes requests are denied. Exiting now
```

Note that: When the needed and requested resources for a certain process are zeroes, we assume it won't need the processor anymore as it won't request any new resources, and since we don't know how much additional time it needs to continue its execution (as Banker's algorithm doesn't include anything about time parameters), we just assume that this process finished its execution when it has no additional requests to make, and its allocated resources are de-allocated to be put in the available resources.