**NORTH SOUTH UNIVERSITY**

**Spring 2023**

**CSE323**: **OPERATING SYSTEMS DESIGN**



**REPORT TITLE**: IMPLEMENTATION OF BANKER’S ALGORITHM

Section 06

Date of Submission: 15/06/23

Submitted to: Rashed Mazumder

|  |  |
| --- | --- |
| **TEAM MEMBER NAME** | **TEAM MEMBER ID** |
| RAFIFA BINTE JAHIR | 2014315042 |
| MM SHOVON HOSSAIN JUNAK | 2011396042 |
| ASIF RAHMAN | 2012691042 |
| MD NAZMUL ISLAM SRABON | 2021227042 |

**Introduction**

An essential component of every computer system or organization is the effective management of resources. Making sure that resources are allocated without leading to deadlock or resource starvation becomes a major challenge in complex systems where several processes compete for limited resources. This problem is addressed by the well-known Banker's Algorithm, which offers a secure and effective way of resource distribution. In this project, we will be implementing the Banker's Algorithm and demonstrate its effectiveness in preventing deadlock and ensuring the maximum utilization of available resources.

**Objectives**

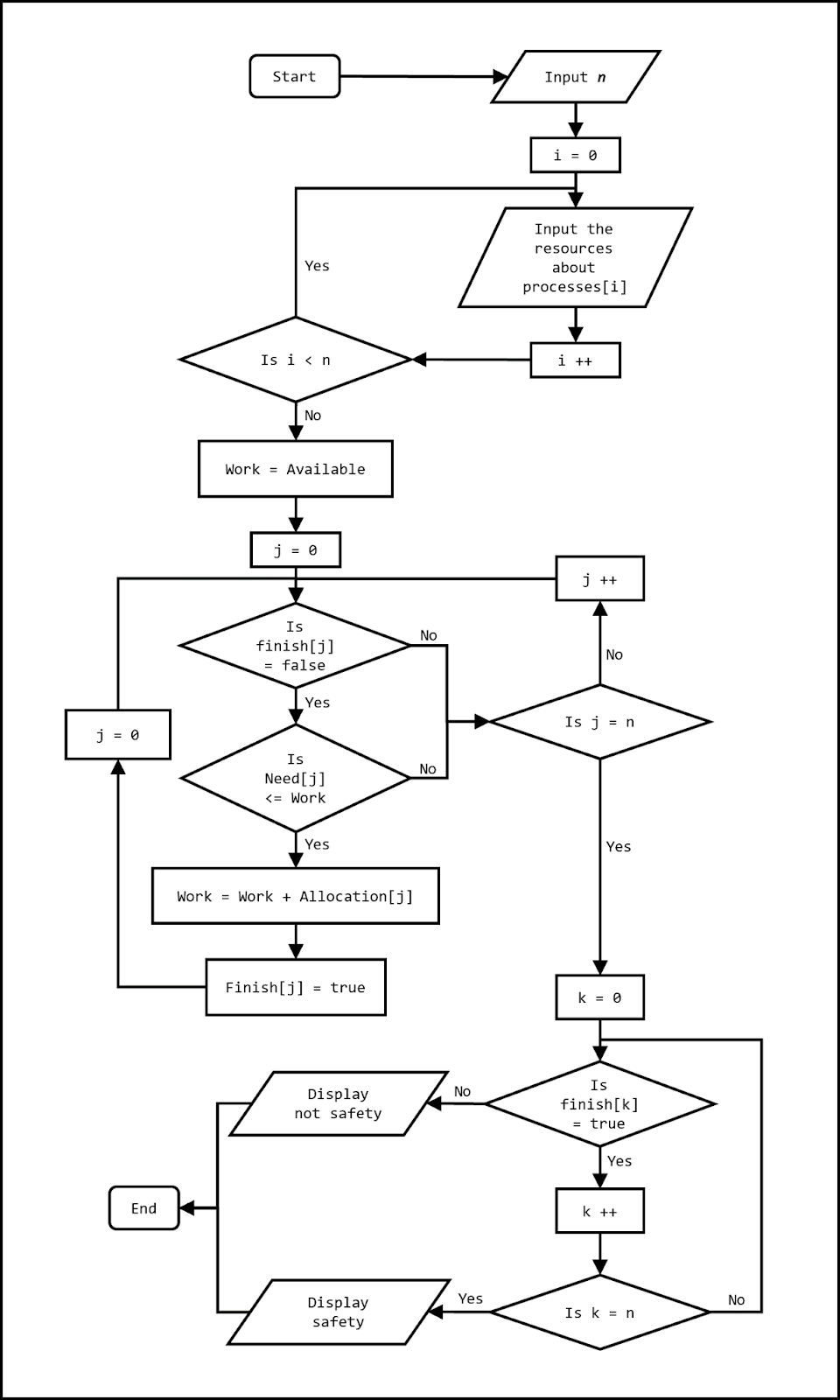
* An application designed to detect and avoid deadlock in the operating system.
* Implementation of Resource Request algorithm and Banker’s algorithm.
* Visual representation of the resource allocation using Need Matrix.
* Visual representation of the resource availability.

**Languages used**

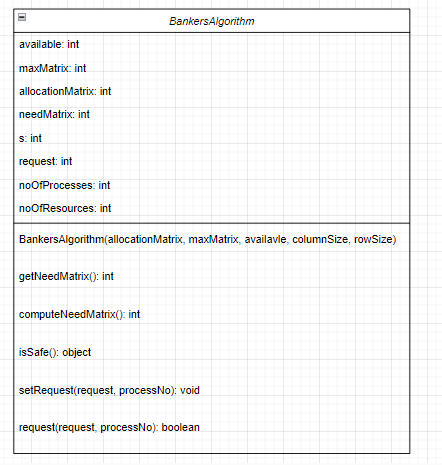
* Java
* Java Swing

**Design**

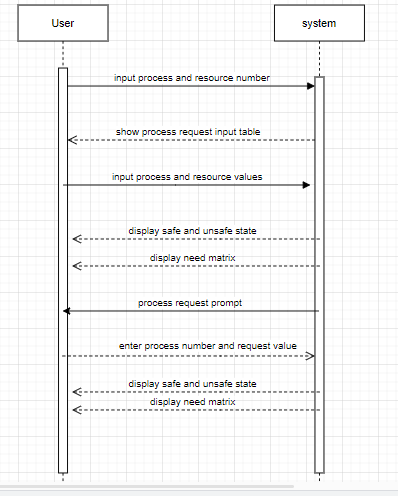
1. **Flow Chart:**

****

1. **Class Diagram:**

****

1. **Sequence Diagram:**

****

**Implementation:**

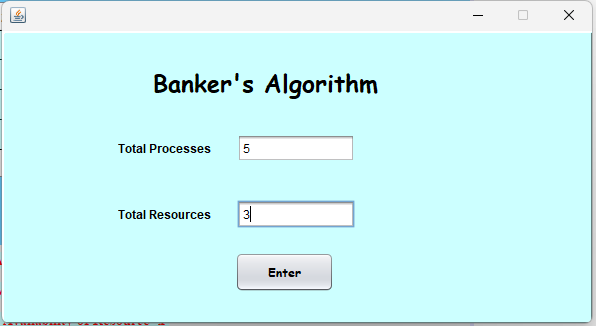
****

Figure: User Input of Processes and Resources

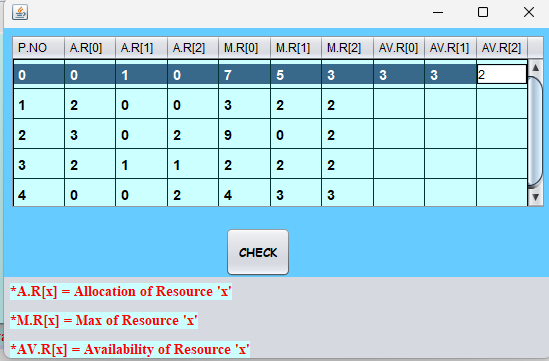
****

Figure: User Input of Process and Resource Value

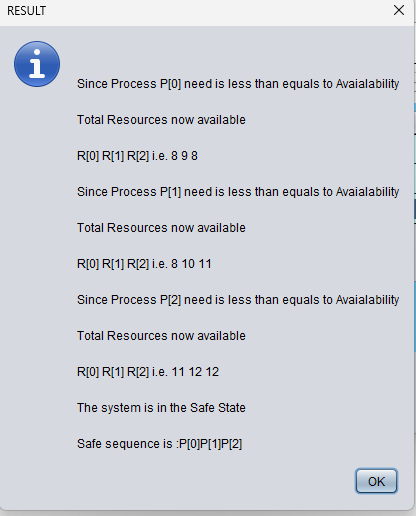
****

Figure: Result

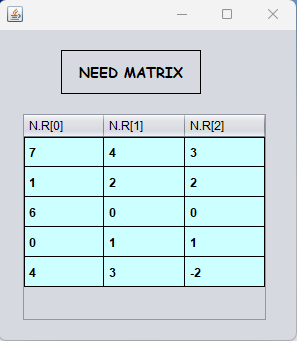


Figure: Need Matrix

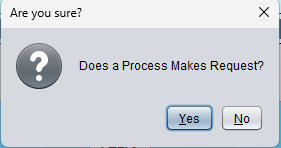
****

Figure: Process Request Prompt

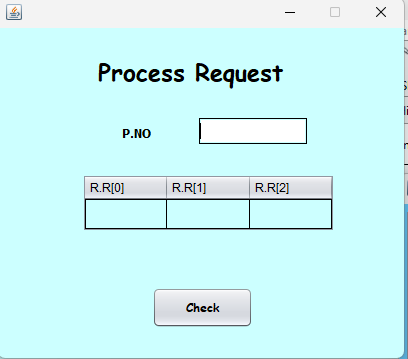
****

Figure: Process Request Input

**Description**

In this project we just Java and java swing to implement the GUI. Using the concept of Banker’s algorithm and Resource Request algorithm we can provide the user with a concise representation of the number of processes assigned and number of resources available after each process. Banker algorithm is used to avoid deadlock and allocate resources safely to each process in the computer system. It helps the operating system to share all the resources amongst the processes efficiently. When a user requests for resources it determines if the system will be in a safe state after the resources are allocated to the requested processes. If the required resources are not available then the processes have to wait until some processes have been carried out releasing the occupied resources.

Let,

m= number of resources

n= number of processes

**Available:** A vector of length m indicates the number of available resources. If Available[j] =k, then k instances of resources are available.

**Max:** n x m matrix. If Max [i, j] =k then process Pi can request at most k instances of resources Rj.

**Allocation:** n x m matrix. If Allocation [i, j] =k then process Pi, is currently allocated k instances of resources Rj.

**Need:** n x m matrix. If Need [i, j] =k then process Pi, may need k more instances of resource Rj to complete its task.

Where,

Need[i][j]= Max[i][j] - Allocation[i][j]

Resource Request Algorithm is used to determine whether requests can be safely granted.

Let,

Request i be the request vector for process Pi.

If Request i [j] == k, then process Pi wants k instances of resource Rj.  
When a request is made for a resource by process Pi,

Then any one of the following things occur:

1. If Request i <= Need i, go to step 2. Otherwise, error is shown as the process request is greater than the max.
2. If request i <= Available, go to step 3. Otherwise, error is shown as the resources are not available.
3. It follows these equations:

         Available= Available - Request’

        Allocation’ = Allocation’ + Request’

        Need’ = Need’ - Request’

**Conclusion:**

Through the implementation of Banker’s Algorithm and Resource-Request algorithm and application like this is used to make it easier for the user to understand how the process requesting for resources and allocation of those resources works. By detecting and avoiding deadlock it allows the system to function efficiently. The requests for the processes which require more resources than available wait for being executed. Once the previous processes are executed the occupied resources are released for the process in waiting to be used. This keeps on going until all the process requests have been fulfilled.