

OPERATING SYSTEM PROJECT

BANKER’S ALGORITHM

**Submitted by:**

1. **Sherif Ashraf Ahmed 16P9033**
2. **Ahmed Hesham Al-Saey 16P6000**

**Submitted to:**

* **Dr. Gamal A. Ebrahim**

**Banker’s Algorithm**

* **Brief Description:**

The Banker algorithm, sometimes referred to as the detection algorithm, is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

* **What is provided in the program?**

program that implements the Banker’s algorithm. Several processes request and release resources and the algorithm will grant a request only if it leaves the system in a safe state. A request that leaves the system in an unsafe state will be denied. The algorithm will consider requests from n customers for m resources types. Processes will continually loop requesting and releasing resources from the system. The processes request and then release random numbers of resources, which are bounded by their respective values in the need array.

* **How the Program works?**

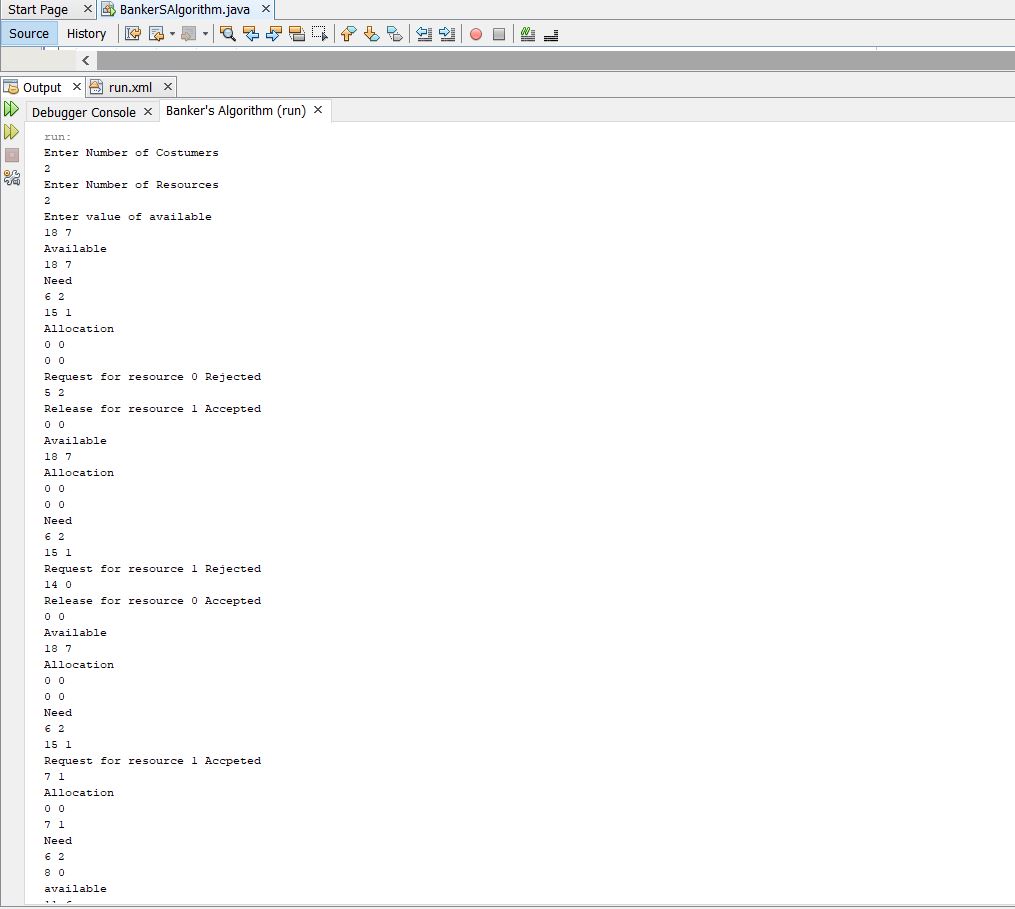
1. The System asks the user to enter number of Costumers (Processes), number of Resources and the available system resources.
2. The System creates 3 arrays:
   1. Need array for each costumer 🡪Randomly.
   2. Allocation array for each costumer 🡪Initially filled with zeros.
   3. Available array 🡪filled with entered values from user.
3. The System starts to generate randomly Requests:
   1. Generates which process will make a request.
   2. Generates the value of this request.
4. Each Request is Generated:
   1. Checked if it is less than both the available for this process and the need.
      1. If it is less than both of them:
         1. Check if this request will keep the system in safe state or not by knowing if this request is given to this process, will the other processes take also all what they want or not
            1. If check is safe, Accept this request.

If this process had all its needs: then move its allocation to available and free it’s allocation.

* + - * 1. Else, Reject this request.
    1. If it is not less than both of them:
       1. Reject this Request.
    2. If the system is finished (when each process takes all its needs), then terminate the program.

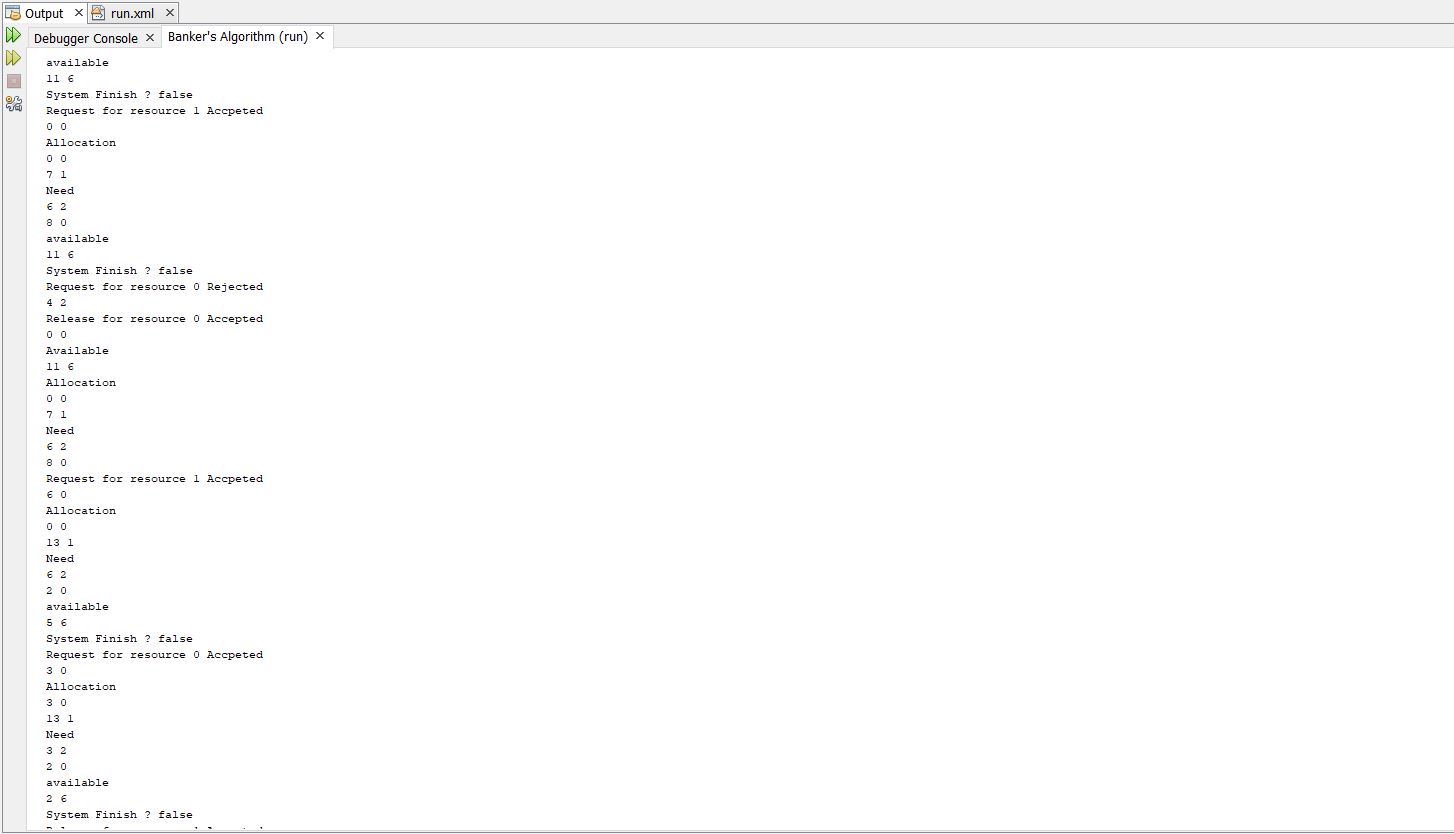
1. The System starts to generate Releases Randomly:
   1. If the system, choose to release:
      1. Generates which process will be released from its allocation.
      2. Generates the value of this release.
      3. The System releases from this process and makes changes to its allocation and the available.

* **Screenshots for a random output from the Program.**



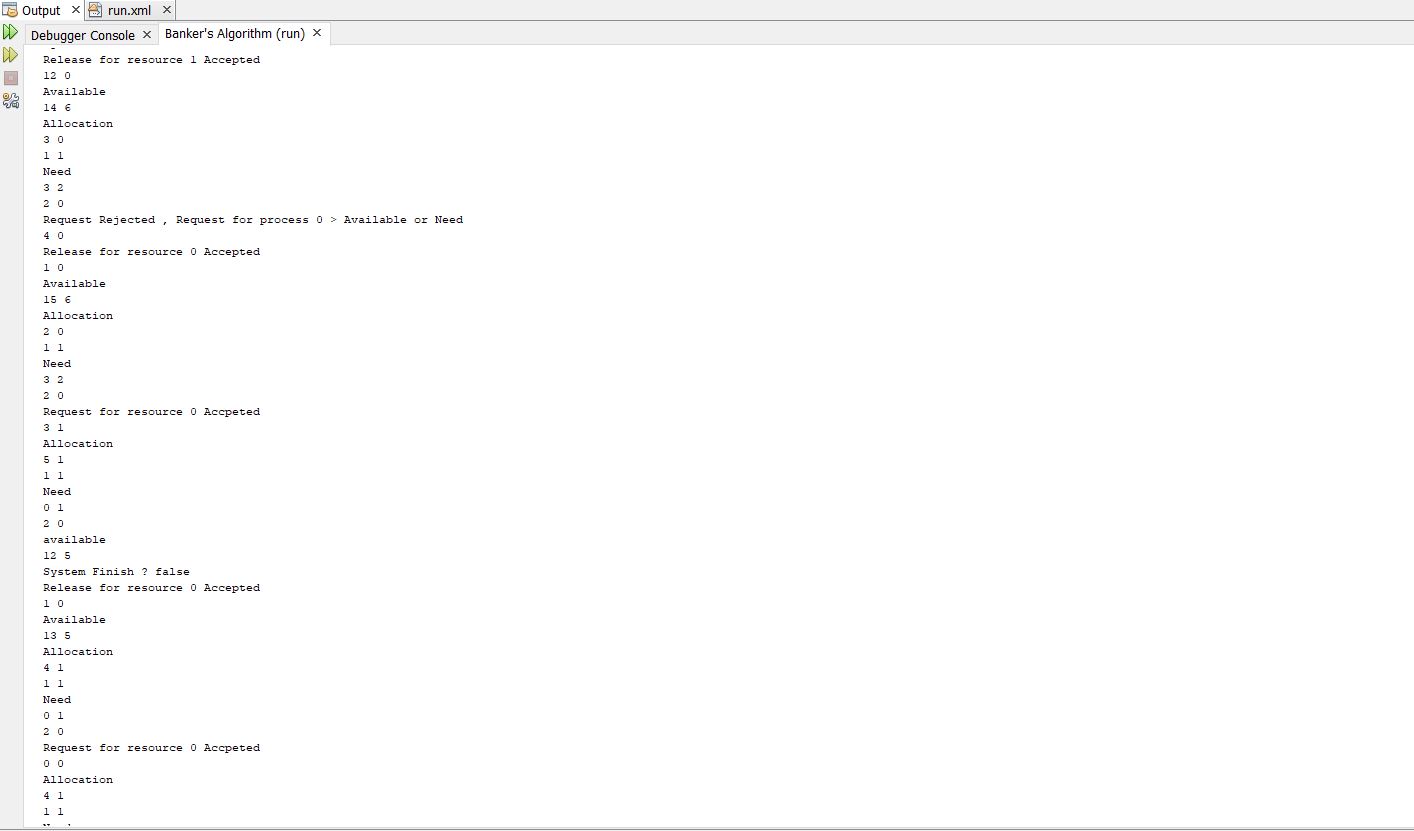
The generated Need Randomly and the Allocation initialized with zeros

A Request was rejected for making the system in unsafe state



Changes in available, allocation and need after the Request is accepted

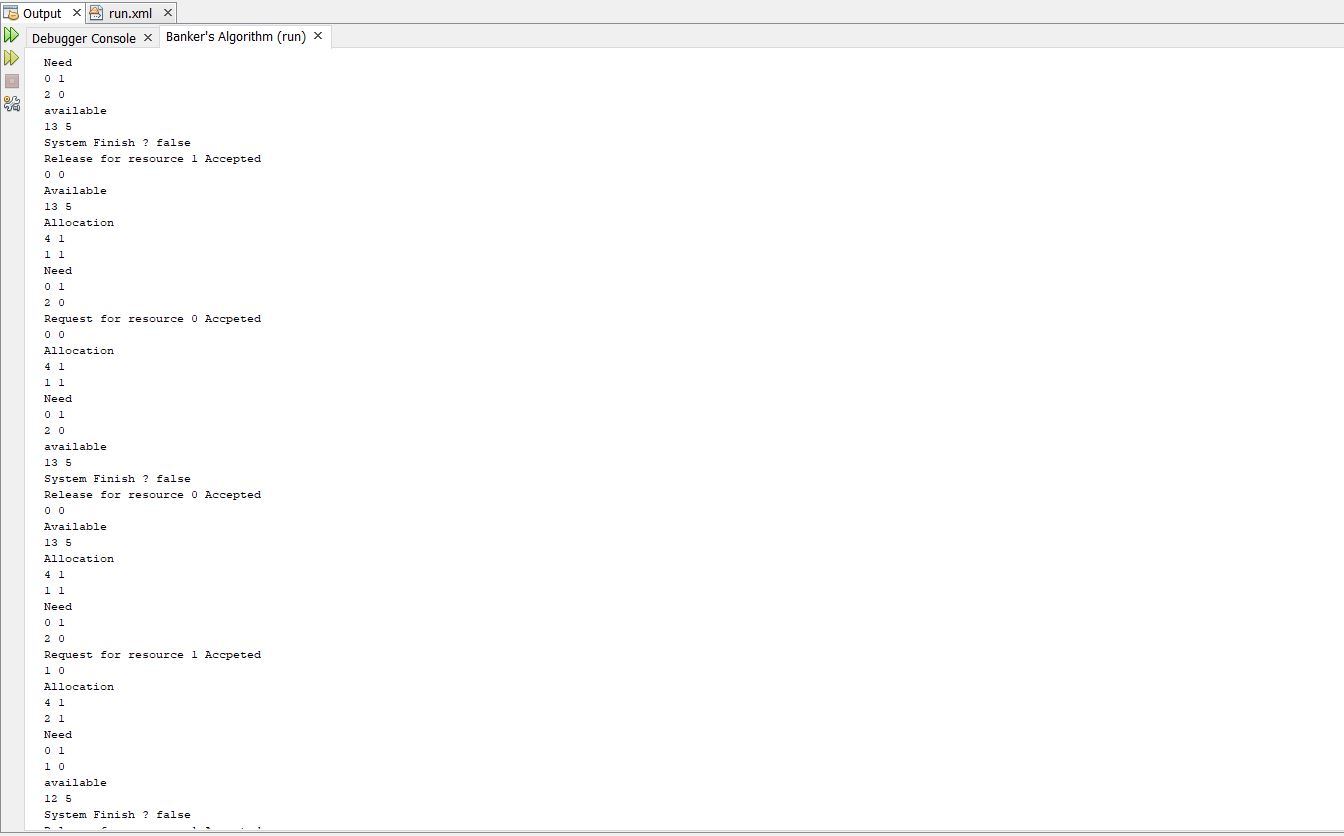
A requested accepted as it keeps the system in safe state

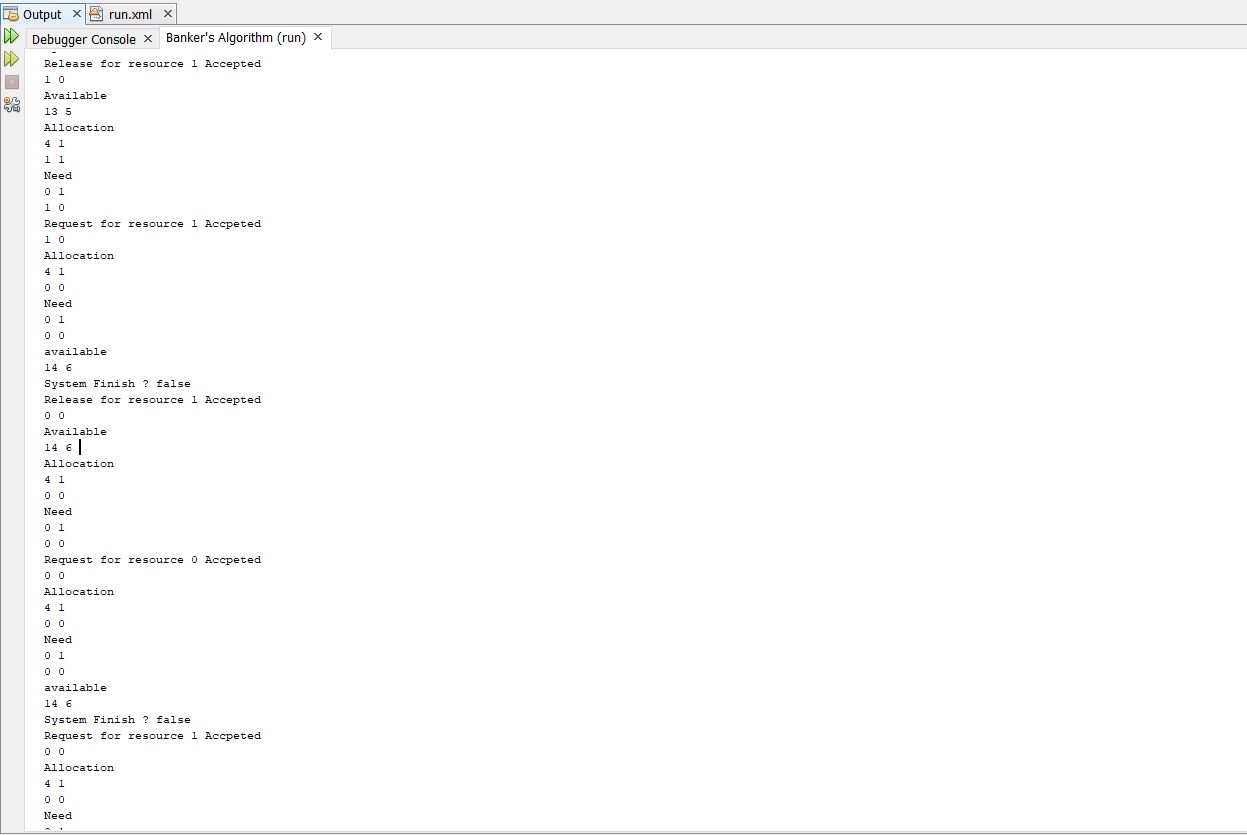


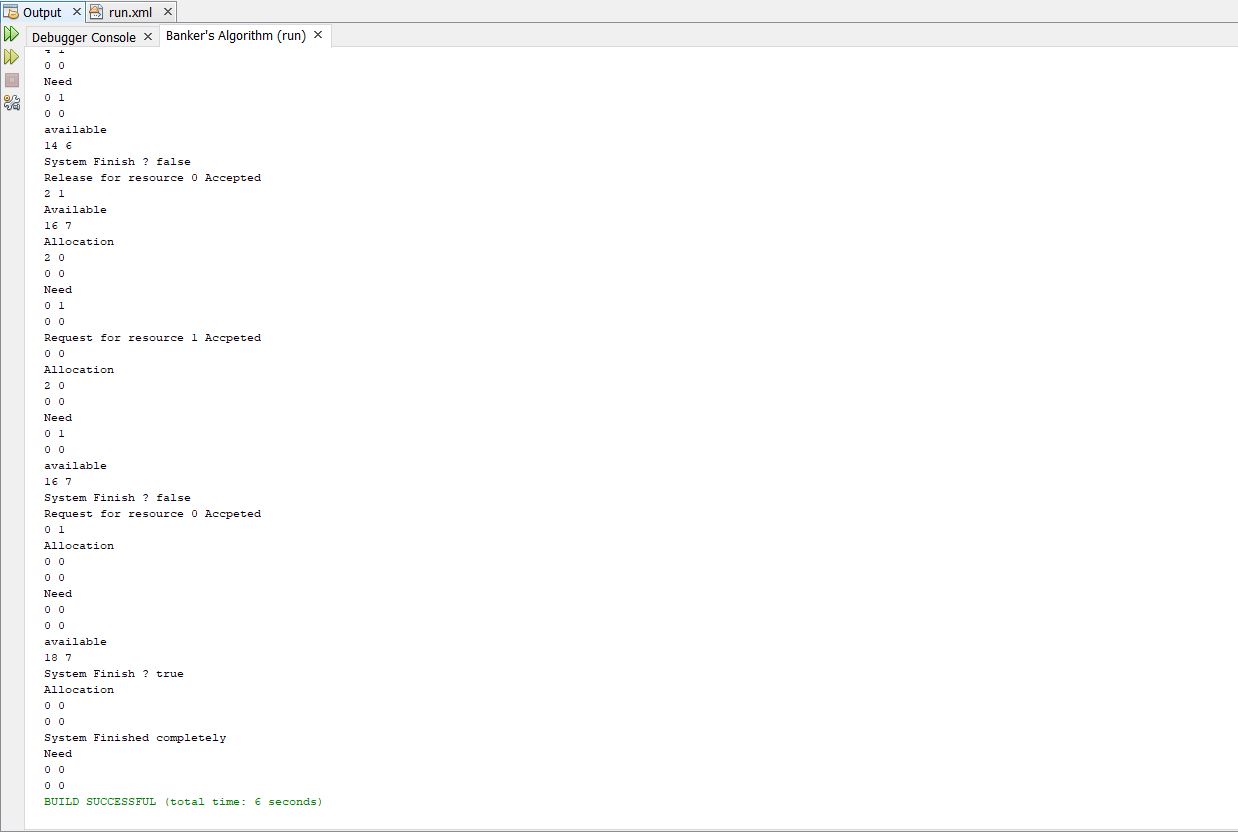
A Request Rejected for being larger than the Need or Available

Changes in Available, Allocation and Need after an Accepted Release

A Release was Accepted







System finished after the Allocation and Need for all Resources are Empty