ALBUKHARY INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTING & INFORMATICS

Project: Bank Queue Simulation

Course Code	**CCC2113**	Course Name	**Data Structures and Algorithm Analysis**
Group Members	**Rania Kherba : AIU22102285** **Malika Amiri : AIU21102298** **Thinley Yeshey Choden : AIU22102188** **Dema Yuden : AIU22102234**		

Assessment

No.	**Criteria**	**Weight**	**Marks**
1.			
2.			
	Total Marks		

Comments

Lecturer's Name: Dr. Mozaherul Hoque Abul Hasanat

Table of Content

1.0 Introduction

In today's competitive world, the success of organizations hinges significantly on enhancing customer satisfaction through the improvement of service quality. This is particularly true for banks, where service quality constitutes a crucial aspect of their core competencies. Managers in the banking sector are especially concerned with the time customers spend waiting for service, as queue length and waiting time are critical factors influencing customer perception of service quality.

Queuing, also known as waiting in line, is a common occurrence in everyday life; for example, banks have customers in line to receive teller assistance, automobiles queue up for re-filling, and employees queue up to use a machine to complete their tasks. Samanta et al. (2007) states that queues occur when consumers (humans or objects) in need of service must wait because there are more of them than there are servers available; or the facility is inefficient, or it takes longer than necessary to serve a client.

To address these concerns, banks strive to achieve an optimal service configuration that satisfies both customers and service providers. Among various evaluation approaches, simulation has demonstrated exceptional capability in modeling and evaluating service scenarios, making it an invaluable tool for this purpose. This project aims to implement and analyze data structures and algorithms related to queues for visualizing customer queues in a bank environment. By simulating and visualizing customer arrivals and service times using queue data structures, the project seeks to provide insights into queue management strategies. The simulation uses queues to manage customers and assigns them to the shortest queue available. The goal is to evaluate different scenarios to optimize queue performance, minimize waiting times, and enhance the overall efficiency and customer satisfaction in a banking setting.

2.0 Code Implementation

Link to Code

Prerequisite:

Please ensure Visual Studio or Apache Netbeans with Java support is installed to run the Java application as it includes GUI components.

3.0 Code Description

Overview

This bank simulation program models the operation of a bank with multiple service counters, handling customer arrivals and services. It uses a graphical user interface (GUI) for interactive input collection and displays the simulation results. The program is written in Java and employs queues to manage customers at each service counter.

Classes and Their Roles

1. **Customer Class**

```java
// This class represents a customer with arrivalTime and serviceTime.
class Customer {
    int arrivalTime; // When the customer arrives.
    int serviceTime; // How long the customer needs to be served.

// Constructor: Initializes the Customer object with arrival and service times.
    Customer(int arrivalTime, int serviceTime) {
        this.arrivalTime = arrivalTime;
        this.serviceTime = serviceTime;
    }
}
```

- **Attributes**:
- arrivalTime: The time at which the customer arrives at the bank.
- serviceTime: The time required to serve the customer.
- **Constructor**: Initializes the arrivalTime and serviceTime for a customer.

2. **Counter Class**

```
// Counter Class
// This class represents a service counter.
class Counter {
    Queue<Customer> queue = new LinkedList<>(); // Queue to hold customers.
    int endTime = 0; // Time when the counter finishes serving all customers.
    int servedCount = 0; // Number of customers served.
    int totalServiceTime = 0; // Total service time of all served customers.

    // Adds a customer to the queue.
    void addCustomer(Customer customer) {
        queue.add(customer);
    }

    // Serves a customer, returns the wait time.
    int serveCustomer() {
        if (queue.isEmpty()) {
            return 0;
        }
        Customer customer = queue.poll();
        int waitTime = Math.max(0, endTime - customer.arrivalTime);
        endTime = Math.max(endTime, customer.arrivalTime) + customer.serviceTime;
        totalServiceTime += customer.serviceTime;
        servedCount++;
        return waitTime;
    }

    // Checks if the counter is idle.
    boolean isIdle() {
        return queue.isEmpty();
    }
// Getter methods for respective fields.
    int getEndTime() {
        return endTime;
    }

    int getQueueSize() {
        return queue.size();
    }

    int getServedCount() {
        return servedCount;
    }

    int getTotalServiceTime() {
        return totalServiceTime;
    }
}
```

- **Attributes**:

- queue: A queue to hold customers waiting for service.
- endTime: The time at which the last customer finished being served.
- servedCount: The total number of customers served by this counter.
- totalServiceTime: The total time spent serving customers.

- **Methods**:

- addCustomer(Customer customer): Adds a customer to the queue.
- serveCustomer(): Serves the next customer in the queue, updates waiting and service times.
- isIdle(): Checks if the counter is idle (no customers in the queue).
- getEndTime(), getQueueSize(), getServedCount(), getTotalServiceTime(): Getter methods for various attributes.

3. **BankSimulationGUI Class**

```
// BankSimulationGUI Class
// This class extends JFrame and represents the main GUI for the simulation.
public class BankSimulationGUI extends JFrame {

    private Counter[] counters; // Array of counters
    private int totalWaitTime = 0; // Total wait time
private int numberOfCustomers; // Number of customers
private int serviceTime; // Service time per customer
private int arrivalRate; // Arrival rate of customers

// Constructor: Initializes the GUI components and sets up the layout.
public BankSimulationGUI() {
    setTitle("Bank Simulation");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());
    setPreferredSize(new Dimension(600, 400));
    setLocationRelativeTo(null);

    // Starts the input collection process.
    collectInputs();
}
```

- **Attributes**:
- counters: An array of Counter objects.
- totalWaitTime: The total waiting time for all customers.
- numberOfCustomers, serviceTime, arrivalRate: Parameters for the simulation.
- **Constructor**: Sets up the GUI and initiates the input collection process.
- **Methods**:
- collectInputs(): Collects user inputs using dialog boxes.
- getUserInput(): Gathers and validates the number of counters, customers, service time, and arrival rate.
- simulateCustomerArrival(): Simulates the arrival of customers and assigns them to the counters with the shortest queue.
- processCustomers(): Processes the customers at each counter.
- displayResults(): Displays the simulation results in a dialog box.
- main(): Entry point of the program.

How the Code Works

1. **Initialization:**

    - When the program starts, the BankSimulationGUI class initializes the GUI and starts collecting user inputs.

2. **Collecting Inputs:**

    - The program uses a series of dialog boxes to collect:
        - The number of service counters.
        - The number of customers.
        - The service time per customer.
        - The arrival rate of customers (number of customers arriving every 2 units of time).

3. **Simulating Customer Arrival:**

    - Customers are created with their arrival times and service times.
    - They are assigned to the counter with the shortest queue.
    - The arrival rate dictates the time intervals at which customers arrive.

4. **Processing Customers:**

    - Each counter serves its customers in the order they arrive.
    - The waiting time for each customer is calculated as the difference between the counter's end time and the customer's arrival time. Specifically:

      Wait Time = max(0,Counter End Time−Customer Arrival Time)

      where the counter's end time is the time when the counter becomes available after serving the previous customer.
    - The end time for each counter is updated to reflect the service time for the current customer:

Counter End Time = max(Counter End Time,Customer Arrival Time)+Service Time

- The total service time and number of customers served are tracked for each counter.

5. **Displaying Results:**

- After all customers have been served, the program calculates:
    - The total time taken (the maximum end time across all counters).
    - The total wait time (sum of all individual wait times).
    - The average waiting time per customer:

$$\text{Average Waiting Time} = \frac{\text{Total Wait Time}}{\text{Number of Customers}}$$

- The results are displayed in a resizable dialog box.
- The user is then prompted whether they want to run another simulation.

4.0 Test Scenarios

1. **Test Scenario 1 - Normal Operating Conditions**

Description: Simulate a typical day with moderate customer arrivals and standard service times.

Input:

- Number of service counters: 3
- Number of customers: 10
- Service time per customer: 6 units
- Arrival rate (customers per 2 units of time): 3

Output:

- Total time taken: 24 units
- Counter #1:
  - Number of customers served: 4
  - Total service time: 24 units
  - End time: 24 units
- Counter #2:
  - Number of customers served: 3
  - Total service time: 18 units
  - End time: 18 units
- Counter #3:
  - Number of customers served: 3
  - Total service time: 18 units
  - End time: 18 units

Overall Statistics:

- Total wait time: 48 units
- Average waiting time per customer: 4.80 units

Description:

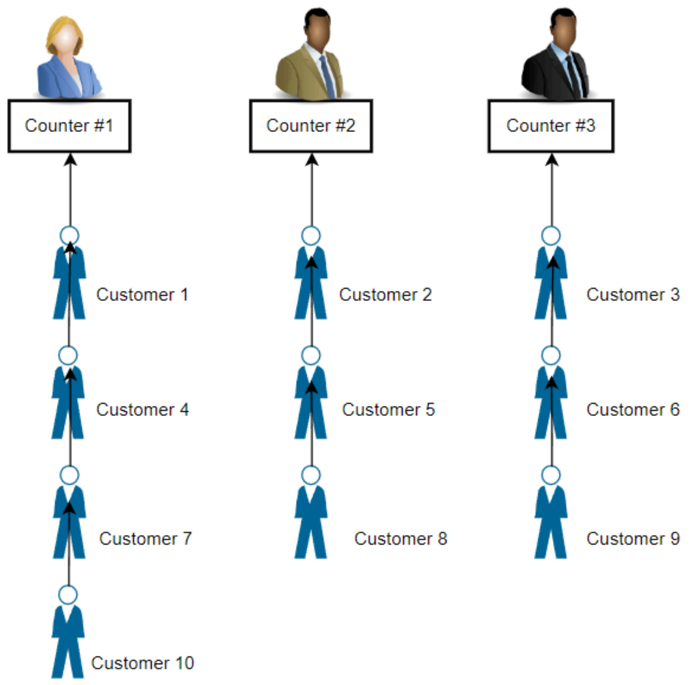


Figure 1 : Customer Distribution Across Service Counters in Normal Operating Condition

The diagram (Figure 1) represents the distribution of customers across three service counters during a typical day at the bank. The simulation considers moderate customer arrivals and standard service times. Each customer is assigned to the counter with the shortest queue upon arrival. The simulation demonstrates effective handling of customer service with three counters, accommodating a total of 10 customers with a service time of 6 units per customer.

The simulation in Figure 1 reveals that the bank's total operation time is 24 units, with Counter #1 handling one more customer than Counters #2 and #3. Despite a cumulative waiting time of 48 units, the average waiting time per customer remained manageable at 4.80 units. This indicates that the bank's queuing system, given the parameters, efficiently distributed the customer load among the counters, ensuring timely service and minimal customer wait times.

The results suggest that the bank's current setup can handle similar traffic volumes effectively, but continuous monitoring and adjustments could further enhance customer satisfaction and operational efficiency.

2. **Test Scenario 2 - Peak Hour Conditions**

Description: Simulate peak banking hours with a high influx of customers.

Input:

- Number of service counters: 3
- Number of customers: 100
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 10

Output:

- Total time taken: 170 units
- Counter #1:
  - Number of customers served: 34
  - Total service time: 170 units
  - End time: 170 units
- Counter #2:
  - Number of customers served: 33
  - Total service time: 165 units
  - End time: 165 units
- Counter #3:
  - Number of customers served: 33
  - Total service time: 165 units
  - End time: 165 units

Overall Statistics:

- Total wait time: 7185 units
- Average waiting time per customer: 71.85 units

Description :

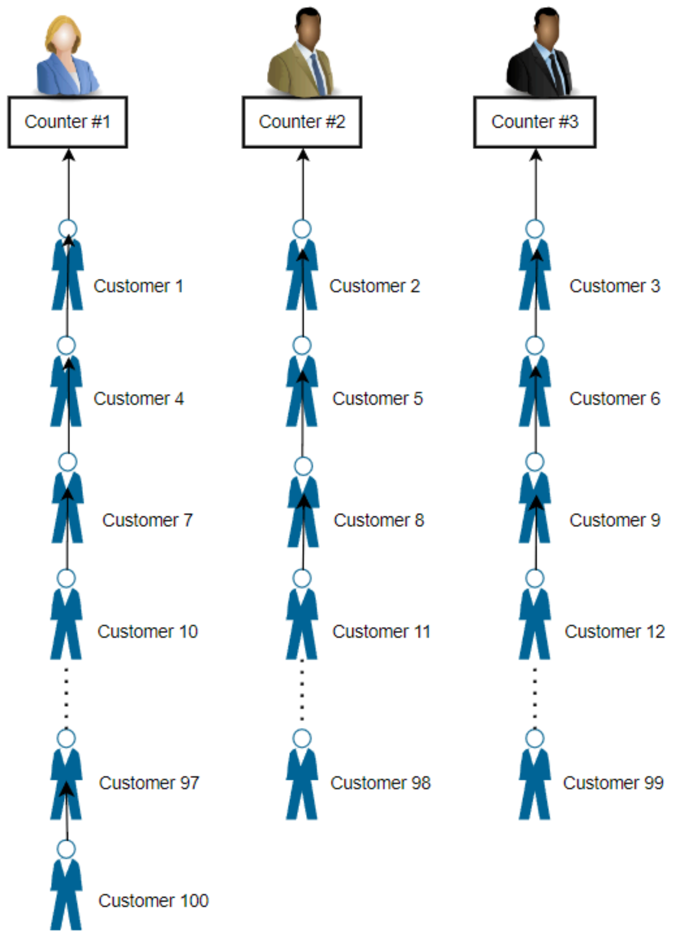


Figure 2 : Customer Distribution Across Service Counters in Peak Hour Condition

The diagram (Figure 2) represents the distribution of customers across three service counters during peak banking hours at the bank.

The simulation in Figure 2 replicates peak conditions with a large number of clients, putting the system's performance under stress. The load was evenly divided among the three service counters, assuming 100 customers, a service time of 5 units per client, and an arrival rate of 10 customers every 2 units of time. Counter #1 serviced 34 clients in a total of 170 units, while Counters #2 and #3 served 33 customers in 165 units. The total time required was 170 units, with a cumulative wait time of 7185 units and an average wait time of 71.85 units per client. This demonstrates the system's balanced load distribution.

3. **Test Scenario 3 - Low Traffic Conditions**

Description: Simulate a slow day with few customer arrivals.

Input:

- Number of service counters: 3
- Number of customers: 3
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 1

Output:

- Total time taken: 9 units
- Counter #1:
  - Number of customers served: 1
  - Total service time: 5 units
  - End time: 5 units
- Counter #2:
  - Number of customers served: 1
  - Total service time: 5 units
  - End time: 7 units
- Counter #3:
  - Number of customers served: 1
  - Total service time: 5 units
  - End time: 9 units

Overall Statistics

- Total wait time: 0 units
- Average waiting time per customer: 0.00 units

Description :

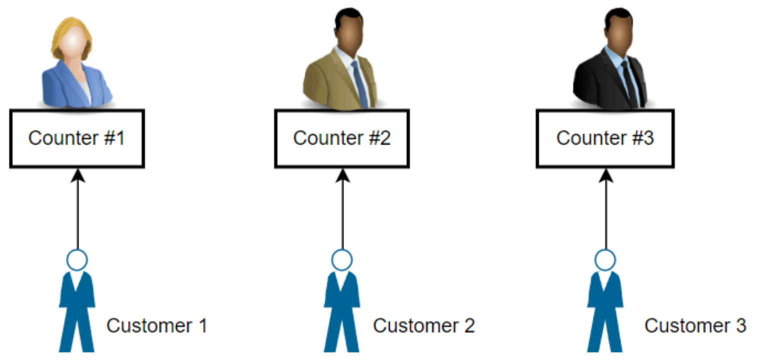


Figure 3 : Customer Distribution Across Service Counters in Low Traffic Condition

The diagram (Figure 3) represents the distribution of customers across three service counters on a slow day with minimal customer arrivals.

The simulation in Figure 3 evaluates system performance under light load conditions, with input parameters including three service counters, three customers, a service time of 5 units per customer, and an arrival rate of 1 customer every 2 units of time. Each counter served one customer: Counter #1 completed its service in 5 units, Counter #2 in 7 units, and Counter #3 in 9 units. The total time taken was 9 units, with no waiting time, resulting in an average waiting time of 0.00 units per customer. This demonstrates the system's efficiency and effectiveness in managing customer service during low traffic conditions.

4. Test Scenario 4 - Single Counter

Description: Simulate the bank operation with only one service counter.

Input:

- Number of service counters: 1
- Number of customers: 10
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 2

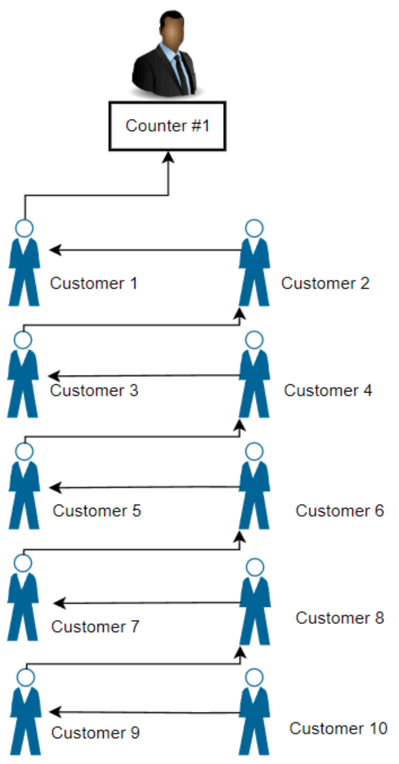
Output:

- Total time taken: 50 units
- Counter #1:
  - Number of customers served: 10
  - Total service time: 50 units
  - End time: 50 units

Overall Statistics:

- Total wait time: 185 units
- Average waiting time per customer: 18.50 units

Description:



Customers arrive in pairs every 2 units of time

Figure 4 : Customer Distribution Across Single Service Counter

The diagram (Figure 4) represents the distribution of customers across a single service counter during a typical day at the bank under minimal resource conditions. The simulation considers moderate customer arrivals and standard service times. All 10 customers are assigned to the single counter as they arrive, resulting in a sequential queuing system. The simulation demonstrates the bank's capability to handle customer service with one counter, accommodating a total of 10 customers, each with a service time of 5 units.

The simulation in Figure 4 reveals that the bank's total operation time is 50 units, with the single counter handling all 10 customers. Despite a cumulative waiting time of 185 units, the average waiting time per customer is 18.50 units. This indicates that, given the parameters, the bank's queuing system under minimal counter conditions leads to significantly higher wait times for customers. The results suggest that while the bank can manage customer flow with one counter, the efficiency and customer satisfaction are greatly compromised. Adding more service counters would be necessary to distribute the customer load more effectively, reduce waiting times, and enhance operational efficiency.

5. Test Scenario 5 - Empty Field and Invalid Input Handling

Input :

- Empty input fields:
- Non-numeric inputs: abc
- Negative numbers : -4

Output

The system displays an error message and prompts the user to re-enter valid inputs.

Description:

This simulation demonstrates the system's response to invalid input scenarios, including empty fields, non-numeric inputs like "abc," and negative numbers such as -4. When any of these invalid inputs are detected, the system promptly displays an error message and asks the user to re-enter valid information. This validation ensures the robustness and user-friendliness of the system by preventing crashes and guiding users to provide correct data.

5.0 Conclusion & Future Work

The Bank Queue Simulation project effectively demonstrates the application of data structures and algorithms in optimizing bank service operations. By simulating customer arrivals and service processes, the model successfully minimizes waiting times and balances the load across multiple counters. The insights gained from this simulation can help bank managers enhance service efficiency and customer satisfaction, highlighting the importance of computational techniques in operational improvements.

Additionally, this project helped us learn valuable skills in designing and implementing queues in a real world scenario, gaining practical experience in problem-solving and algorithm analysis.

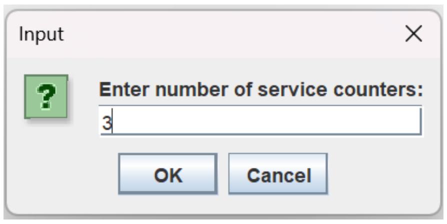
For future works, features like saving/loading configurations, exporting results, and more sophisticated styling could further improve the application. These improvements could include using modern UI frameworks for a visually appealing interface, offering custom themes, incorporating interactive graphs and animations, and adding tooltips and a help section. These enhancements would improve usability, aesthetics, and functionality.

6.0 References

1. Samanta, S., Gupta, U., & Sharma, R. (2007). Analyzing discrete-time D-BMAP/G/1/N queue with single and multiple vacations. *European Journal of Operational Research*, *182*(1), 321–339. https://doi.org/10.1016/j.ejor.2006.09.031
2. Visual Studio (https://visualstudio.microsoft.com/) [For coding]

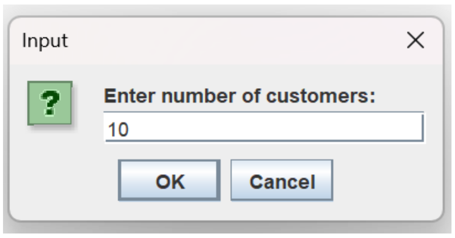
7.0 Appendices

Test Scenario 1 - Normal Operating Conditions


Input

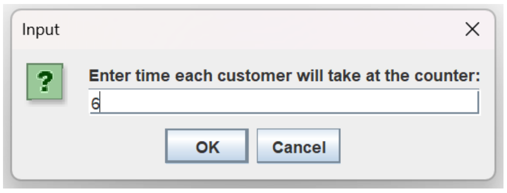
Enter number of service counters:

3


Input

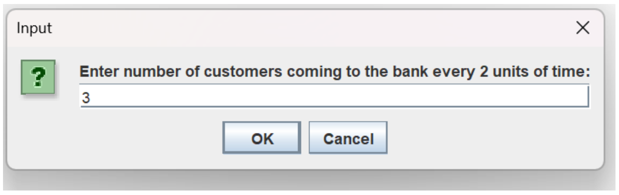
Enter number of customers:

10


Input

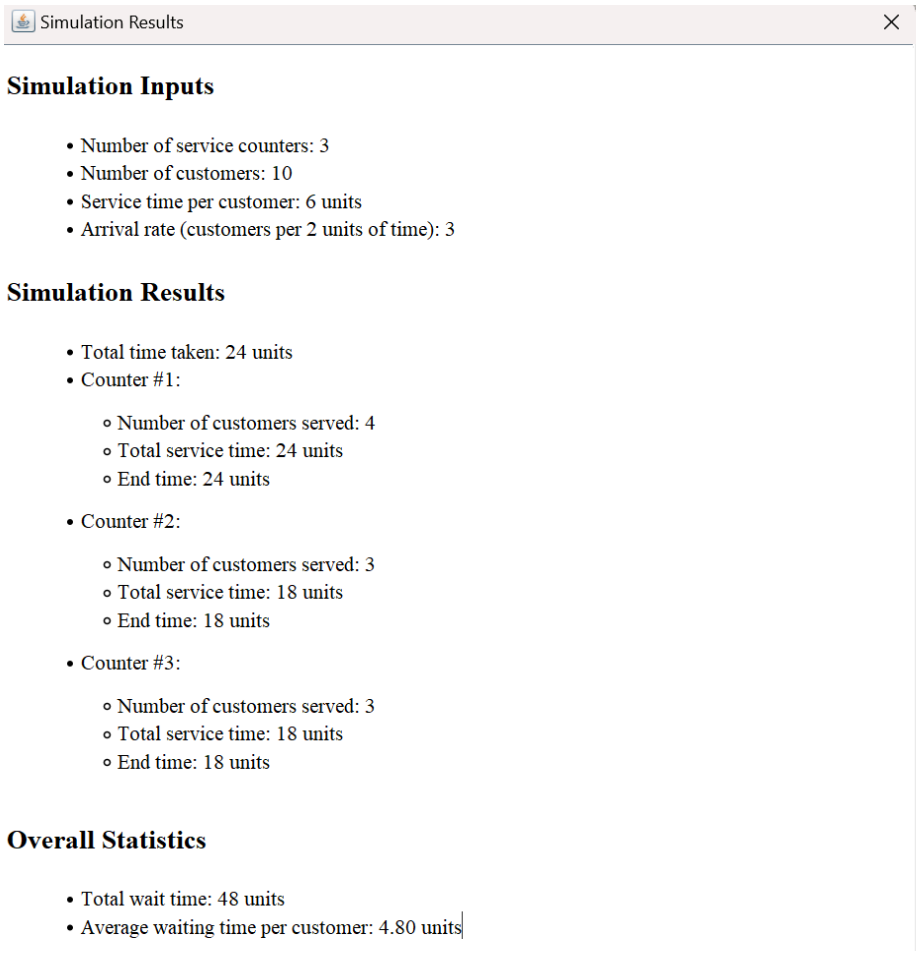
Enter time each customer will take at the counter:

6


Input

Enter number of customers coming to the bank every 2 units of time:

3


Simulation Results

Simulation Inputs

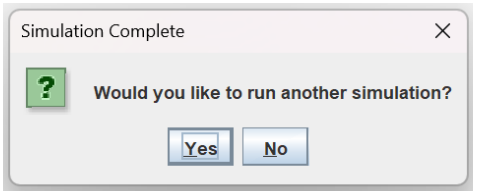
- Number of service counters: 3
- Number of customers: 10
- Service time per customer: 6 units
- Arrival rate (customers per 2 units of time): 3

Simulation Results

- Total time taken: 24 units
- Counter #1:
  - Number of customers served: 4
  - Total service time: 24 units
  - End time: 24 units
- Counter #2:
  - Number of customers served: 3
  - Total service time: 18 units
  - End time: 18 units
- Counter #3:
  - Number of customers served: 3
  - Total service time: 18 units
  - End time: 18 units

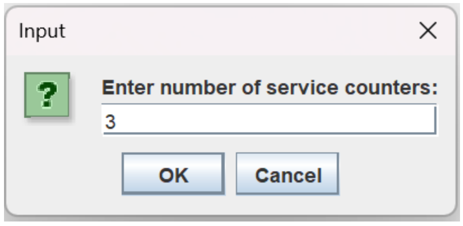
Overall Statistics

- Total wait time: 48 units
- Average waiting time per customer: 4.80 units


Simulation Complete

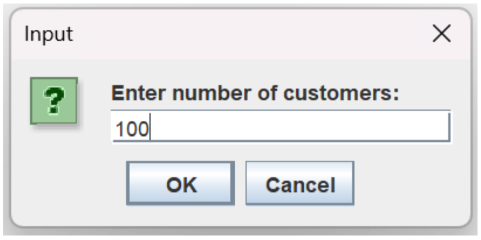
Would you like to run another simulation?

Test Scenario 2 - Peak Hour Conditions

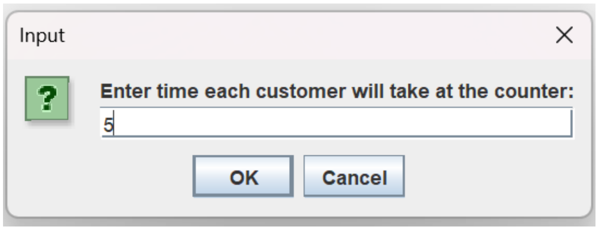

Input

Enter number of service counters:

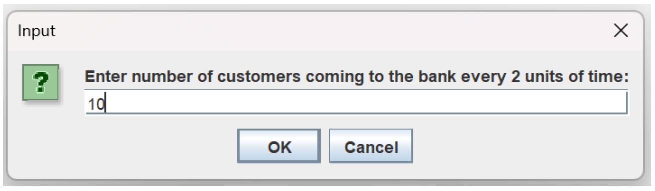
3



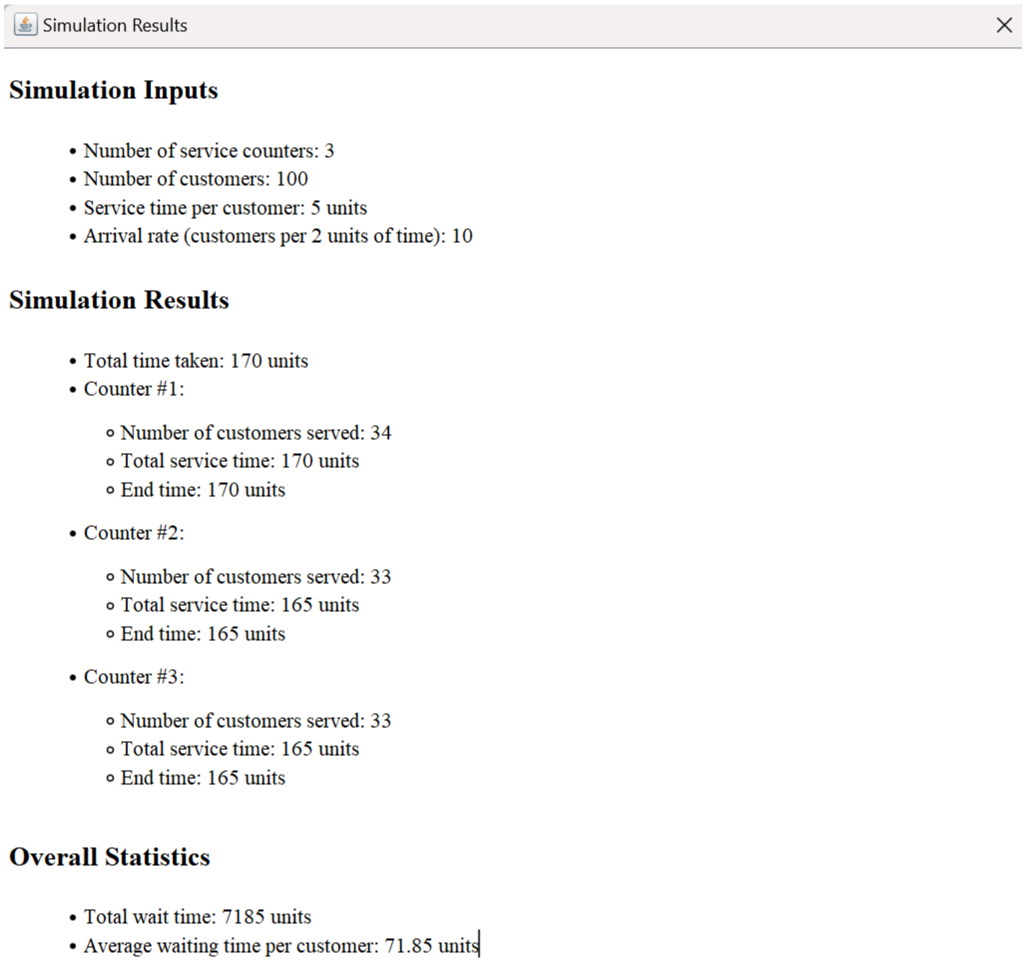
Enter number of customers:
100



Enter time each customer will take at the counter:
5



Enter number of customers coming to the bank every 2 units of time:
10


Simulation Results

Simulation Inputs

- Number of service counters: 3
- Number of customers: 100
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 10

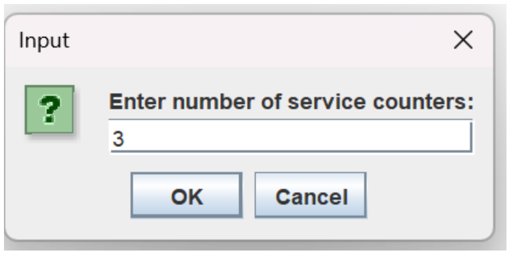
Simulation Results

- Total time taken: 170 units
- Counter #1:
  - Number of customers served: 34
  - Total service time: 170 units
  - End time: 170 units
- Counter #2:
  - Number of customers served: 33
  - Total service time: 165 units
  - End time: 165 units
- Counter #3:
  - Number of customers served: 33
  - Total service time: 165 units
  - End time: 165 units

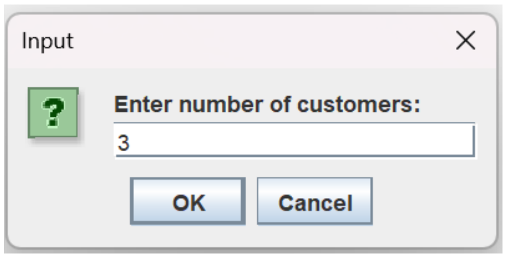
Overall Statistics

- Total wait time: 7185 units
- Average waiting time per customer: 71.85 units

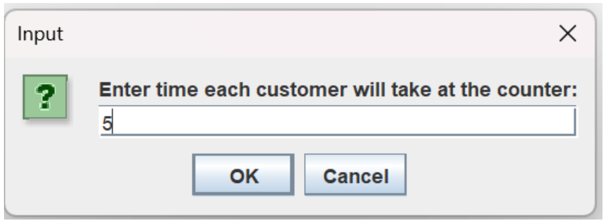
Test Scenario 3 - Low Traffic Conditions

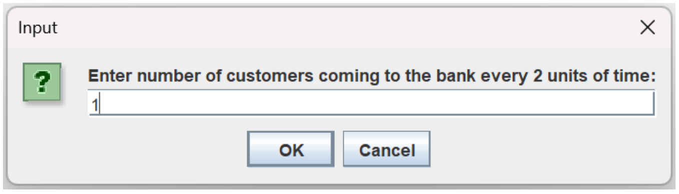


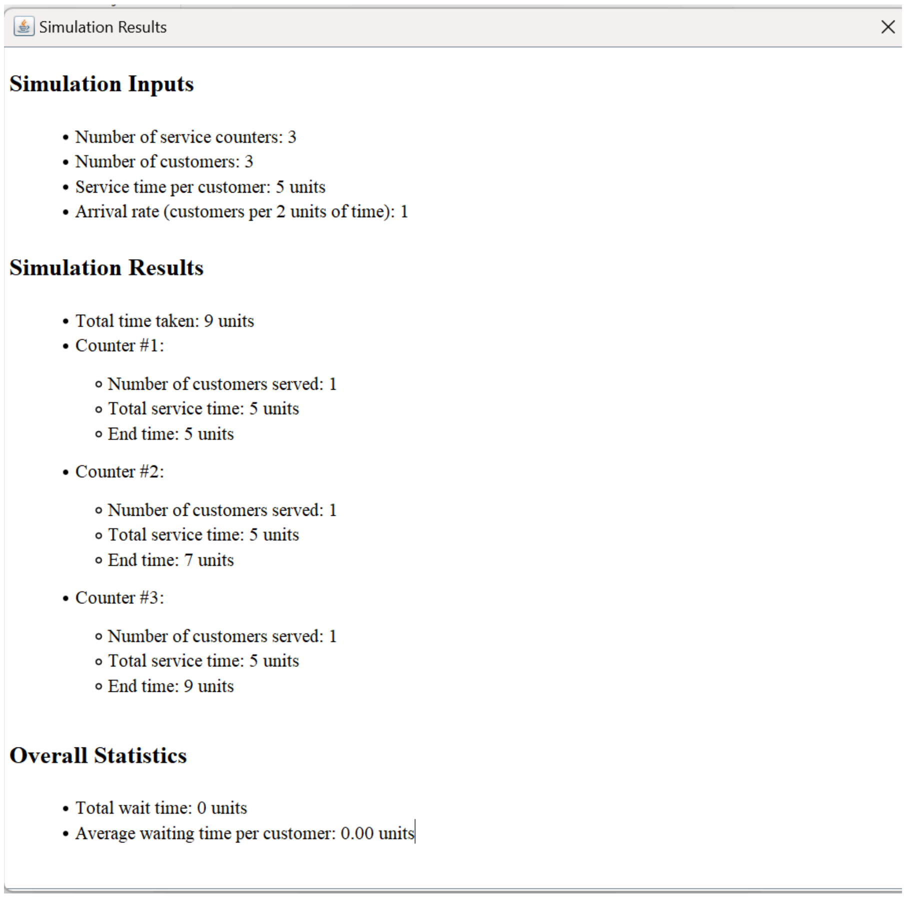
Enter number of service counters: 3



Enter number of customers: 3




Input
Enter number of customers coming to the bank every 2 units of time:
1
OK
Cancel



Simulation Inputs

- Number of service counters: 3
- Number of customers: 3
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 1

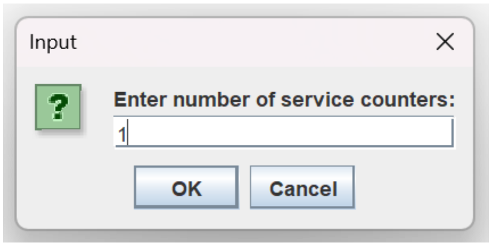
Simulation Results

- Total time taken: 9 units
- Counter #1:
    - Number of customers served: 1
    - Total service time: 5 units
    - End time: 5 units
- Counter #2:
    - Number of customers served: 1
    - Total service time: 5 units
    - End time: 7 units
- Counter #3:
    - Number of customers served: 1
    - Total service time: 5 units
    - End time: 9 units

Overall Statistics

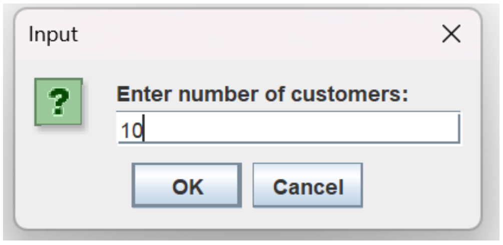
- Total wait time: 0 units
- Average waiting time per customer: 0.00 units

Test Scenario 4 - Single Counter


Input

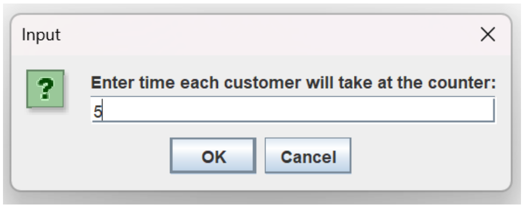
Enter number of service counters:

1


Input

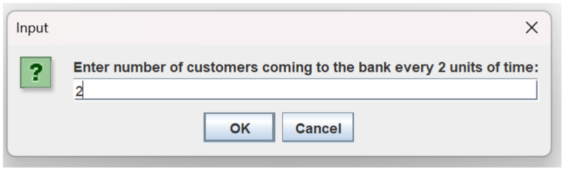
Enter number of customers:

10


Input

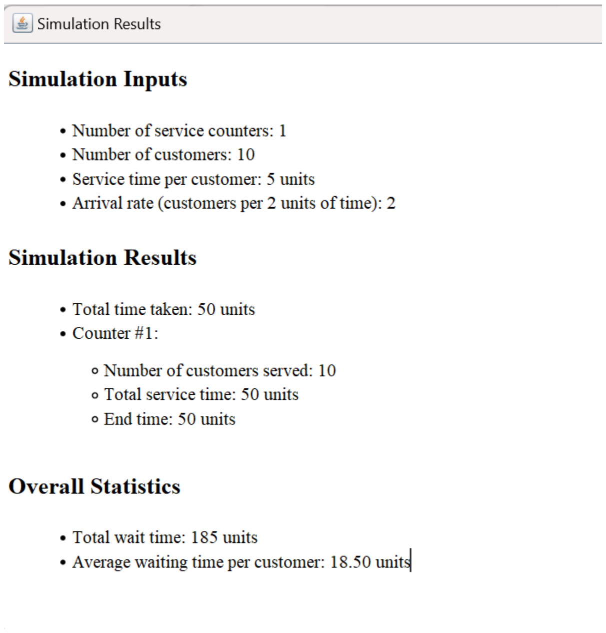
Enter time each customer will take at the counter:

5


Input

Enter number of customers coming to the bank every 2 units of time:

2


Simulation Results

Simulation Inputs

- Number of service counters: 1
- Number of customers: 10
- Service time per customer: 5 units
- Arrival rate (customers per 2 units of time): 2

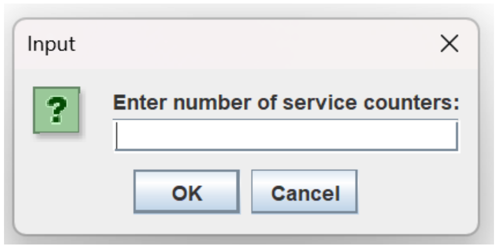
Simulation Results

- Total time taken: 50 units
- Counter #1:
  - Number of customers served: 10
  - Total service time: 50 units
  - End time: 50 units

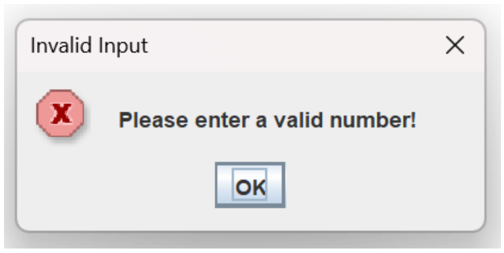
Overall Statistics

- Total wait time: 185 units
- Average waiting time per customer: 18.50 units

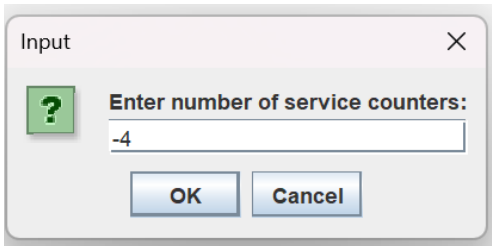
Test Scenario 5 - Empty Field and Invalid Input Handling


Input

Enter number of service counters:

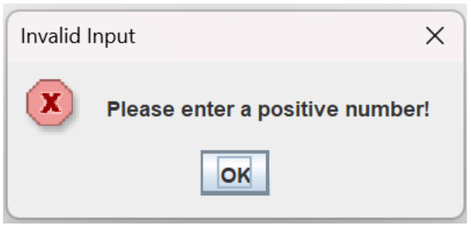

Invalid Input

Please enter a valid number!

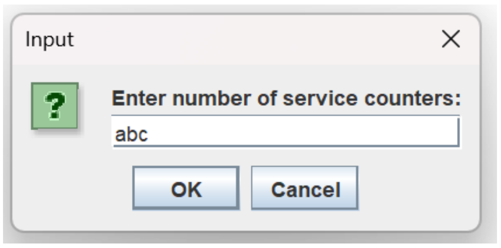

Input

Enter number of service counters:

-4

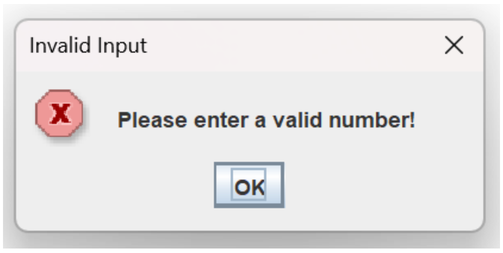


Please enter a positive number!



Enter number of service counters:

abc



Please enter a valid number!