PA05 - Bank Simulation

Generated by Doxygen 1.8.11

Contents

1 Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ArrayQueue	3
ArrayQueueBank	8
Client	13
CountingSort	18
Event	21
LinkQueue	21
LinkQueueBank	25
Node	30
NodeBank	34
Simulation1A	40

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

PA05/ArrayQueue.cpp **This is the implementation of the ArrayQueue class**	44
PA05/ArrayQueue.h **This is the header of the ArrayQueue class**	45
PA05/ArrayQueueBank.cpp **This is the implementation of the ArrayQueueBank class**	45
PA05/ArrayQueueBank.h **This is the header of the ArrayQueueBank class**	46
PA05/Client.cpp **This is the implementation of the Client class**	46
PA05/Client.h **This is the header of the Client class**	47
PA05/CountingSort.cpp **This is the implementation of the CountingSort class**	47

3 Class Documentation

3.1 ArrayQueue Class Reference

Public Member Functions

- ArrayQueue ()

  The default constructor of an ArrayQueue object.
- ArrayQueue (int sentCap)

  The parameterized constructor of an ArrayQueue object.
- ∼ArrayQueue ()

  The destructor of an ArrayQueue object.
- bool IsEmpty ()

  Checks if the ArrrayQueue is empty.
- bool Enqueue (int entry)

  Adds an item to the queue.

- bool Dequeue ()

    Removes the item at the front of the queue.
- int Peek ()

    Gets the item at the front of the queue.
- void Print ()

    Prints the contents of the queue.

Private Attributes

- int **front**
- int **back**
- int **count**
- int **capacity**
- int ∗ **data**

3.1.1 Constructor & Destructor Documentation

3.1.1.1 ArrayQueue::ArrayQueue ( )

The default constructor of an ArrayQueue object.

This constructor initializes values of a ArrayQueue object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.1.1.2 ArrayQueue::ArrayQueue (int *sentCap*)

The parameterized constructor of an ArrayQueue object.

This constructor initializes values of a ArrayQueue object to sent values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.1.1.3 ArrayQueue::∼ArrayQueue ( )

The destructor of an ArrayQueue object.

This safely removes an ArrayQueue object from memory

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.1.2 Member Function Documentation

3.1.2.1 bool ArrayQueue::Dequeue ( )

Removes the item at the front of the queue.

Removes the item at the front of the queue, the value is not returned.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns true if an item could be removed, false if it was already empty.

Note

This function can be modified to return the dequeued value.

3.1.2.2 bool ArrayQueue::Enqueue (int *entry*)

Adds an item to the queue.

Adds an item to the back of the queue, this implementation is a circular array.

Algorithm None.

Parameters

in	*entry*	The integer value to insert into the queue
out	*None.*	

Returns

Returns true if there is space for the new value, false if the queue is full.

Note

None.

3.1.2.3 bool ArrayQueue::IsEmpty ( )

Checks if the ArrrayQueue is empty.

Checks the count of the queue to see if it is empty

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns true if the queue is empty, false if it contains at least one item.

Note

None.

3.1.2.4 int ArrayQueue::Peek ( )

Gets the item at the front of the queue.

Checks the front of the queue and returns the value if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.1.2.5 void ArrayQueue::Print ( )

Prints the contents of the queue.

Runs through the queue and prints the contents.

Algorithm If the queue wraps around then the function just prints the entire queue, if it does not wrap then it just prints those values in scope.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/ArrayQueue.h
- PA05/ArrayQueue.cpp

3.2 ArrayQueueBank Class Reference

Public Member Functions

- ArrayQueueBank ()

  The default constructor of an ArrayQueueBank object.
- ArrayQueueBank (int sentCap)

  The parameterized constructor of an ArrayQueueBank object.
- ∼ArrayQueueBank ()

  The destructor of an ArrayQueue object.
- bool IsEmpty ()

  Checks if the queue is empty.
- bool Enqueue (int sentArr, int sentTrans)

  Adds an item to the queue.
- bool Dequeue ()

  Removes the item at the front of the queue.
- int PeekArrival ()

  Gets the arrival time of the client at the front of the queue.
- int PeekTransaction ()

  Gets the transaction time of the client at the front of the queue.
- Client ∗ PeekFront ()

  Gets the item at the front of the queue.
- void Print ()

  Prints the contents of the queue.

Private Attributes

- int **front**
- int **back**
- int **count**
- int **capacity**
- Client ∗ **data**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 ArrayQueueBank::ArrayQueueBank ( )

The default constructor of an ArrayQueueBank object.

This constructor initializes values of a ArrayQueueBank object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.2.1.2 ArrayQueueBank::ArrayQueueBank (int *sentCap*)

The parameterized constructor of an ArrayQueueBank object.

This constructor initializes values of a ArrayQueueBank object to sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.2.1.3 ArrayQueueBank::∼ArrayQueueBank ( )

The destructor of an ArrayQueue object.

This safely removes an ArrayQueue object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.2.2 Member Function Documentation

3.2.2.1 bool ArrayQueueBank::Dequeue ( )

Removes the item at the front of the queue.

Removes the item at the front of the queue, the value is not returned.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns true if an item could be removed, false if it was already empty.

Note

This function can be modified to return the dequeued value.

3.2.2.2 bool ArrayQueueBank::Enqueue (int *sentArr,* int *sentTrans*)

Adds an item to the queue.

Adds an item to the back of the queue, this implementation is a circular array.

Algorithm None.

Parameters

`in`	*entry*	The integer value to insert into the queue
`out`	*None.*	

Returns

Returns true if there is space for the new value, false if the queue is full.

Note

None.

3.2.2.3 bool ArrayQueueBank::IsEmpty ( )

Checks if the queue is empty.

Checks the count of the queue to see if it is empty

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns true if the queue is empty, false if it contains at least one item.

Note

None.

3.2.2.4 int ArrayQueueBank::PeekArrival ( )

Gets the arrival time of the client at the front of the queue.

Checks the front of the queue and returns the arrival time of the client if there is one.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.2.2.5 Client ∗ ArrayQueueBank::PeekFront ( )

Gets the item at the front of the queue.

Checks the front of the queue and returns the address if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, NULL if there isn't

Note

None.

3.2.2.6 int ArrayQueueBank::PeekTransaction ( )

Gets the transaction time of the client at the front of the queue.

Checks the front of the queue and returns the transaction time of the client if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.2.2.7 void ArrayQueueBank::Print ( )

Prints the contents of the queue.

Runs through the queue and prints the contents.

Algorithm If the queue wraps around then the function just prints the entire queue, if it does not wrap then it just prints those values in scope.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/ArrayQueueBank.h
- PA05/ArrayQueueBank.cpp

3.3 Client Class Reference

Public Member Functions

- Client ()

  The default constructor of a Client object.
- Client (int sentArr, int sentTrans)

  The default constructor of a Client object.
- ∼Client ()

  The destructor of a Client object.
- int GetArrival ()

  Gets the arrival time of this Client.
- int GetTransaction ()

  Gets the transaction time of this Client.
- void SetArrival (int sentVal)

  Sets the arrival time of this Client.
- void SetTransaction (int sentVal)

  Sets the transaction time of this Client.
- void Print ()

  Prints the values of this Client.
- void operator= (Client &sentClient)

  The assignment operator for a Client.

Private Attributes

- int **arrival**
- int **transaction**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Client::Client ( )

The default constructor of a Client object.

This constructor initializes values of a Client object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.3.1.2 Client::Client (int *sentArr*, int *sentTrans*)

The default constructor of a Client object.

This constructor initializes values of a Client object to sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.3.1.3 Client::∼Client ( )

The destructor of a Client object.

Safely removes a Client object from memory

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.3.2 Member Function Documentation

3.3.2.1 int Client::GetArrival ( )

Gets the arrival time of this Client.

Returns the arrival value of this Client object

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns the integer value of the arrival time of this Client.

Note

None.

3.3.2.2 int Client::GetTransaction ( )

Gets the transaction time of this Client.

Returns the transaction value of this Client object

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns the integer value of the transaction time of this Client.

Note

None.

3.3.2.3 void Client::operator= (**Client** & *sentClient*)

The assignment operator for a Client.

Overloads the assignment operator for a Client object

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.3.2.4 void Client::Print ( )

Prints the values of this Client.

Prints the arrival and transaction times of the client in the formal "[ARRIVALTIME - TRANSTIME]"

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.3.2.5 void Client::SetArrival (int *sentVal*)

Sets the arrival time of this Client.

Modifies the arrival value of this Client object

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.3.2.6 void Client::SetTransaction (int *sentVal*)

Sets the transaction time of this Client.

Modifies the transaction value of this Client object

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/Client.h
- PA05/Client.cpp

3.4 CountingSort Class Reference

Public Member Functions

- CountingSort ()

  The default constructor of a CountingSort object.
- CountingSort (Client ∗data, int size, int max)

  The parameterized constructor of a CountingSort object.
- ∼CountingSort ()

  The destructor of a CountingSort object.
- Client ∗ DoSort ()

  This function runs the sorting algorithm.
- Client ∗ DoSort (Client ∗data, int size, int max)

  This function runs the sorting algorithm.

Private Attributes

- int **size**
- Client ∗ **data**
- int **max**

3.4.1 Constructor & Destructor Documentation

3.4.1.1 CountingSort::CountingSort ( )

The default constructor of a CountingSort object.

This constructor initializes values of a CountingSort object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.4.1.2 CountingSort::CountingSort (Client ∗ *sentData*, int *sentSize*, int *sentMax*)

The parameterized constructor of a CountingSort object.

This constructor initializes values of a CountingSort object to the sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.4.1.3 CountingSort::∼CountingSort ( )

The destructor of a CountingSort object.

This safely removes a CountingSort object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.4.2 Member Function Documentation

3.4.2.1 Client ∗ CountingSort::DoSort ( )

This function runs the sorting algorithm.

The function takes the data and sorts it with the counting sort algorithm

Algorithm Counts the frequency of each value in the data, then sorts it by the count

Parameters

in	*None.*	
out	*The*	array pointed at by data is now sorted

Returns

None.

Note

None.

3.4.2.2 Client ∗ CountingSort::DoSort (Client ∗ *sentData,* int *sentSize,* int *sentMax*)

This function runs the sorting algorithm.

The function takes the data and sorts it with the counting sort algorithm with the sent values

Algorithm Counts the frequency of each value in the data, then sorts it by the count

Parameters

in	*sentData*	Pointer to the integer array to be sorted
in	*sentSize*	The size of the array
in	*sentMax*	The maximum value in the data, constantly 1M for this project
out	*The*	array pointed at by data is now sorted

Returns

Returns a pointer to the sorted integer array

Note

None.

The documentation for this class was generated from the following files:

- PA05/CountingSort.h
- PA05/CountingSort.cpp

3.5 Event Class Reference

Public Member Functions

- **Event** (bool type, int time)
- int **GetTime** ()
- bool **IsArrival** ()

Private Attributes

- bool **isArrival**
- int **eventTime**
- int **eventLength**

The documentation for this class was generated from the following file:

- PA05/Event.h

3.6 LinkQueue Class Reference

Public Member Functions

- LinkQueue ()

  The default constructor of a LinkQueue object.
- ∼LinkQueue ()

  The destructor of a LinkQueue object.
- bool IsEmpty ()

  Checks if the LinkQueue is empty.
- bool Enqueue (int entry)

  Adds an item to the queue.
- bool Dequeue ()

  Removes the item at the front of the queue.
- int Peek ()

  Gets the item at the front of the queue.
- void Print ()

  Prints the contents of the queue.

Private Attributes

- Node $*$ **front**
- Node $*$ **back**

3.6.1 Constructor & Destructor Documentation

3.6.1.1 LinkQueue::LinkQueue ( )

The default constructor of a LinkQueue object.

This constructor initializes values of a LinkQueue object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.6.1.2 LinkQueue::∼LinkQueue ( )

The destructor of a LinkQueue object.

This safely removes a LinkQueue object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.6.2 Member Function Documentation

3.6.2.1 bool LinkQueue::Dequeue ( )

Removes the item at the front of the queue.

Removes the item at the front of the queue, the value is not returned.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns true if an item could be removed, false if it was already empty.

Note

This function can be modified to return the dequeued value.

3.6.2.2 bool LinkQueue::Enqueue (int *entry*)

Adds an item to the queue.

Adds an item to the back of the queue, this implementation uses nodes.

Algorithm None.

Parameters

in	*entry*	The integer value to insert into the queue
out	*None.*	

Returns

Returns true if there is space for the new value, false if the queue is full. Node based means that it will never be full.

Note

None.

3.6.2.3 bool LinkQueue::IsEmpty ( )

Checks if the LinkQueue is empty.

Checks if the front and back of the queue are equal to NULL

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns true if the queue is empty, false otherwise.

Note

None.

3.6.2.4 int LinkQueue::Peek ( )

Gets the item at the front of the queue.

Checks the front of the queue and returns the value if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.6.2.5 void LinkQueue::Print ( )

Prints the contents of the queue.

Runs through the queue and prints the contents.

Algorithm Prints each node in the queue.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/LinkQueue.h
- PA05/LinkQueue.cpp

3.7 LinkQueueBank Class Reference

Public Member Functions

- LinkQueueBank ()

  The default constructor of a LinkQueueBank object.
- ∼LinkQueueBank ()

  The destructor of a LinkQueueBank object.
- bool IsEmpty ()

  Checks if the queue is empty.
- bool Enqueue (int sentArr, int sentTrans)

  Adds an item to the queue.
- bool Dequeue ()

  Removes the item at the front of the queue.
- int PeekArrival ()

  Gets the arrival time of the client at the front of the queue.
- int PeekTransaction ()

  Gets the transaction time of the client at the front of the queue.
- void Print ()

  Prints the contents of the queue.

Private Attributes

- NodeBank ∗ **front**
- NodeBank ∗ **back**

3.7.1 Constructor & Destructor Documentation

3.7.1.1 LinkQueueBank::LinkQueueBank ( )

The default constructor of a LinkQueueBank object.

This constructor initializes values of a LinkQueueBank object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.7.1.2 LinkQueueBank::∼LinkQueueBank ( )

The destructor of a LinkQueueBank object.

This safely removes a LinkQueueBank object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.7.2 Member Function Documentation

3.7.2.1 bool LinkQueueBank::Dequeue ( )

Removes the item at the front of the queue.

Removes the item at the front of the queue, the value is not returned.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns true if an item could be removed, false if it was already empty.

Note

This function can be modified to return the dequeued value.

3.7.2.2 bool LinkQueueBank::Enqueue (int *sentArr,* int *sentTrans*)

Adds an item to the queue.

Adds an item to the back of the queue, this implementation uses nodes.

Algorithm None.

Parameters

`in`	*entry*	The integer value to insert into the queue
`out`	*None.*	

Returns

Returns true if there is space for the new value, false if the queue is full. Node based means that it will never be full.

Note

None.

3.7.2.3 bool LinkQueueBank::IsEmpty ( )

Checks if the queue is empty.

Checks if the front and back of the queue are equal to NULL

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns true if the queue is empty, false otherwise.

Note

None.

3.7.2.4 int LinkQueueBank::PeekArrival ( )

Gets the arrival time of the client at the front of the queue.

Checks the front of the queue and returns the arrival time of the client if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.7.2.5 int LinkQueueBank::PeekTransaction ( )

Gets the transaction time of the client at the front of the queue.

Checks the front of the queue and returns the transaction time of the client if there is one.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value if there is one, -1 if there isn't (in this project the data used is only positive)

Note

None.

3.7.2.6 void LinkQueueBank::Print ( )

Prints the contents of the queue.

Runs through the queue and prints the contents.

Algorithm Prints each node in the queue.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/LinkQueueBank.h
- PA05/LinkQueueBank.cpp

3.8 Node Class Reference

Public Member Functions

- Node ()

  The default constructor of a Node object.
- Node (int sentVal, Node ∗nextNode)

  The default constructor of a Node object.
- ∼Node ()

  The destructor of a Node object.
- int GetValue ()

  Gets the value of the node.
- Node ∗ GetNext ()

  Gets the address of the next node.
- void SetValue (int sentVal)

  Sets the value of the node.
- void SetNext (Node ∗nextPtr)

  Sets the address of the next node.
- void Print ()

  Prints the value of the node.

Private Attributes

- int **value**
- Node ∗ **next**

3.8.1 Constructor & Destructor Documentation

3.8.1.1 Node::Node ( )

The default constructor of a Node object.

This constructor initializes values of a Node object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.8.1.2 Node::Node (int *sentVal,* Node ∗ *nextNode*)

The default constructor of a Node object.

This constructor initializes values of a Node object to sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.8.1.3 Node::∼Node ( )

The destructor of a Node object.

Safely removes this Node object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.8.2 Member Function Documentation

3.8.2.1 Node ∗ Node::GetNext ( )

Gets the address of the next node.

Returns the address of the next node.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the address of the next node.

Note

None.

3.8.2.2 int Node::GetValue ( )

Gets the value of the node.

Returns the value of the node.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the value of the node.

Note

None.

3.8.2.3 void Node::Print ( )

Prints the value of the node.

Prints the value of the node formatted as "[VALUE]"

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.8.2.4 void Node::SetNext (Node ∗ *nextPtr*)

Sets the address of the next node.

Sets the address of the next node.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.8.2.5 void Node::SetValue (int *sentVal*)

Sets the value of the node.

Sets the value of the node.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/Node.h
- PA05/Node.cpp

3.9 NodeBank Class Reference

Public Member Functions

- NodeBank ()

  The default constructor of a NodeBank object.
- NodeBank (int sentArr, int sentTran, NodeBank ∗nextNodeBank)

  The default constructor of a NodeBank object.
- ∼NodeBank ()

  The destructor of a NodeBank object.
- int GetArrival ()

  Gets the arrival time of this Node.
- int GetTransaction ()

  Gets the transaction time of this Node.
- NodeBank ∗ GetNext ()

  Gets the address of the next node.
- void SetArrival (int sentVal)

  Sets the arrival time of this Node.
- void SetTransaction (int sentVal)

  Sets the transaction time of this Node.
- void SetNext (NodeBank ∗nextPtr)

  Sets the address of the next node.
- void Print ()

  Prints the value of the node.

Private Attributes

- int **arrival**
- int **transaction**
- NodeBank ∗ **next**

3.9.1 Constructor & Destructor Documentation

3.9.1.1 NodeBank::NodeBank ( )

The default constructor of a NodeBank object.

This constructor initializes values of a NodeBank object to default values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.9.1.2 NodeBank::NodeBank (int *sentArr,* int *sentTran,* NodeBank ∗ *nextNode*)

The default constructor of a NodeBank object.

This constructor initializes values of a NodeBank object to sent values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.9.1.3 NodeBank::∼NodeBank ( )

The destructor of a NodeBank object.

Safely removes this NodeBank object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.9.2 Member Function Documentation

3.9.2.1 int NodeBank::GetArrival ( )

Gets the arrival time of this Node.

Returns the arrival value of this Node object

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of the arrival time of this Node.

Note

None.

3.9.2.2 NodeBank ∗ NodeBank::GetNext ( )

Gets the address of the next node.

Returns the address of the next node.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the address of the next node.

Note

None.

3.9.2.3 int NodeBank::GetTransaction ( )

Gets the transaction time of this Node.

Returns the transaction value of this Node object

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of the transaction time of this Node.

Note

None.

3.9.2.4 void NodeBank::Print ( )

Prints the value of the node.

Prints the value of the node formatted as "[ARRIVAL - TRANSACTION]"

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.9.2.5 void NodeBank::SetArrival (int *sentVal*)

Sets the arrival time of this Node.

Modifies the arrival value of this Node object

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.9.2.6 void NodeBank::SetNext (NodeBank * *nextPtr*)

Sets the address of the next node.

Sets the address of the next node.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.9.2.7 void NodeBank::SetTransaction (int *sentVal*)

Sets the transaction time of this Node.

Modifies the transaction value of this Node object

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

The documentation for this class was generated from the following files:

- PA05/NodeBank.h
- PA05/NodeBank.cpp

3.10 Simulation1A Class Reference

Public Member Functions

- Simulation1A ()

  The default constructor of a Simulation1A object.
- Simulation1A (Client ∗sentClients, int count)

  The default constructor of a Simulation1A object.
- ∼Simulation1A ()

  The destructor of a Simulation1A object.
- void SendClients (Client ∗sentClients, int count)

  Sets the clients of this simulation.
- void ResetSimulation ()

  Resets the simulation.
- void Simulate ()

  Runs the simulation.
- void ProcessArrival (Client ∗sentClient, int time)

  Processes the arrival of a client.
- void ProcessDeparture (Client ∗sentClient, int time)

  Processes the departure of a client.

Private Attributes

- Client ∗ **clients**
- ArrayQueueBank **clientQueue**
- ArrayQueueBank **bankQueue**
- ArrayQueue **arrivalEvents**
- ArrayQueue **departureEvents**
- int **clientCount**
- bool **tellerAvailable**

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Simulation1A::Simulation1A ( )

The default constructor of a Simulation1A object.

This constructor initializes values of a Simulation1A object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.10.1.2 Simulation1A::Simulation1A (Client $*$ *sentClients*, int *count*)

The default constructor of a Simulation1A object.

This constructor initializes values of a Simulation1A object to sent values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.10.1.3 Simulation1A::∼Simulation1A ( )

The destructor of a Simulation1A object.

Safely removes this object from memeory.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.10.2 Member Function Documentation

3.10.2.1 void Simulation1A::ProcessArrival (Client * *sentClient*, int *time*)

Processes the arrival of a client.

Takes the client that arrived and adds it to the bankQueue.

Algorithm None.

Parameters

`in`	*sentClient*	Pointer to the client that arrived
`in`	*time*	The time that the client arrived.
`out`	*None.*	

Returns

None.

Note

None.

3.10.2.2 void Simulation1A::ProcessDeparture (Client * *sentClient*, int *time*)

Processes the departure of a client.

Takes the client that departed and adds removes it from the bank queue.

Algorithm None.

Parameters

`in`	*sentClient*	Pointer to the client that arrived
`in`	*time*	The time that the client left.
`out`	*None.*	

Returns

None.

Note

None.

3.10.2.3 void Simulation1A::ResetSimulation ( )

Resets the simulation.

Completely dequeues both queues of the simulation.

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.10.2.4 void Simulation1A::SendClients (Client ∗ *sentClients,* int *count*)

Sets the clients of this simulation.

Allows the user to change the clients used for the simulation

Algorithm None.

Parameters

`in`	*sentClients*	The address of the Clients array to be used
`in`	*count*	The amount of Clients in the array.
`out`	*None.*	

Returns

None.

Note

None.

3.10.2.5 void Simulation1A::Simulate ( )

Runs the simulation.

Runs the simulation using the current clients and count.

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

This function couldn't be fully implemented due to time constraints.

The documentation for this class was generated from the following files:

- PA05/Simulation1A.h
- PA05/Simulation1A.cpp

4 File Documentation

4.1 PA05/ArrayQueue.cpp File Reference

This is the implementation of the ArrayQueue class.

```
#include "ArrayQueue.h"
```

4.1.1 Detailed Description

This is the implementation of the ArrayQueue class.

Author

Bryce Monaco

This file contains the implementation of the ArrayQueue class

Version

1.0

Note

This is a version only meant for integer values.

4.2 PA05/ArrayQueue.h File Reference

This is the header of the ArrayQueue class.

```
#include <iostream>
```

Classes

- class ArrayQueue

4.2.1 Detailed Description

This is the header of the ArrayQueue class.

Author

Bryce Monaco

This file contains the header of the ArrayQueue class

Version

1.0

Note

This is a version only meant for integer values.

4.3 PA05/ArrayQueueBank.cpp File Reference

This is the implementation of the ArrayQueueBank class.

```
#include "ArrayQueueBank.h"
```

4.3.1 Detailed Description

This is the implementation of the ArrayQueueBank class.

Author

Bryce Monaco

This file contains the implementation of the ArrayQueueBank class

Version

1.0

Note

This is a version only meant for Client objects.

4.4 PA05/ArrayQueueBank.h File Reference

This is the header of the ArrayQueueBank class.

```
#include <iostream>
#include "Client.h"
```

Classes

- class ArrayQueueBank

4.4.1 Detailed Description

This is the header of the ArrayQueueBank class.

Author

Bryce Monaco

This file contains the header of the ArrayQueueBank class

Version

1.0

Note

This is a version only meant for Client objects.

4.5 PA05/Client.cpp File Reference

This is the implementation of the Client class.

```
#include "Client.h"
```

4.5.1 Detailed Description

This is the implementation of the Client class.

Author

Bryce Monaco

This file contains the implementation of the Client class

Version

1.0

Note

None.

4.6 PA05/Client.h File Reference

This is the header of the Client class.

```
#include <iostream>
```

Classes

- class Client

4.6.1 Detailed Description

This is the header of the Client class.

Author

Bryce Monaco

This file contains the header of the Client class

Version

1.0

Note

None.

4.7 PA05/CountingSort.cpp File Reference

This is the implementation of the CountingSort class.

```
#include "CountingSort.h"
```

4.7.1 Detailed Description

This is the implementation of the CountingSort class.

Author

Bryce Monaco

This file contains the implementation of the CountingSort class

Version

1.0

Note

This is modified from PA04's counting sort to work with Clients. Sorts by arrival times.

4.8 PA05/CountingSort.h File Reference

This is the header of the CountingSort class.

```
#include <iostream>
#include <ctime>
#include "Client.h"
```

Classes

- class CountingSort

4.8.1 Detailed Description

This is the header of the CountingSort class.

Author

Bryce Monaco

This file contains the header of the CountingSort class

Version

1.0

Note

Modified version of PA04's CountingSort

4.9 PA05/Event.h File Reference

This is the header of the Event class.

Classes

- class Event

4.9.1 Detailed Description

This is the header of the Event class.

Author

Bryce Monaco

This file contains the header of the Event class

Version

1.0

Note

This header does not have an implementation file.

4.10 PA05/LinkQueue.cpp File Reference

This is the implementation of the LinkQueue class.

```
#include "LinkQueue.h"
```

4.10.1 Detailed Description

This is the implementation of the LinkQueue class.

Author

Bryce Monaco

This file contains the implementation of the LinkQueue class

Version

1.0

Note

This is a version only meant for integer values.

4.11 PA05/LinkQueue.h File Reference

This is the header of the LinkQueue class.

```
#include <iostream>
#include "Node.h"
#include <memory>
```

Classes

- class LinkQueue

4.11.1 Detailed Description

This is the header of the LinkQueue class.

Author

Bryce Monaco

This file contains the header of the LinkQueue class

Version

1.0

Note

This is a version only meant for integer values.

4.12 PA05/LinkQueueBank.cpp File Reference

This is the implementation of the LinkQueueBank class.

```
#include "LinkQueueBank.h"
```

4.12.1 Detailed Description

This is the implementation of the LinkQueueBank class.

Author

Bryce Monaco

This file contains the implementation of the LinkQueueBank class

Version

1.0

Note

This is a version only meant for Client values.

4.13 PA05/LinkQueueBank.h File Reference

This is the header of the LinkQueueBank class.

```
#include <iostream>
#include "NodeBank.h"
```

Classes

- class LinkQueueBank

4.13.1 Detailed Description

This is the header of the LinkQueueBank class.

Author

Bryce Monaco

This file contains the header of the LinkQueueBank class

Version

1.0

Note

This is a version only meant for Client values.

4.14 PA05/Node.cpp File Reference

This is the implementation of the Node class.

```
#include "Node.h"
```

4.14.1 Detailed Description

This is the implementation of the Node class.

Author

Bryce Monaco

This file contains the implementation of the Node class

Version

1.0

Note

This is a version only meant for integer values.

4.15 PA05/Node.h File Reference

This is the header of the Node class.

```
#include <iostream>
```

Classes

- class Node

4.15.1 Detailed Description

This is the header of the Node class.

Author

Bryce Monaco

This file contains the header of the Node class

Version

1.0

Note

This is a version only meant for integer values.

4.16 PA05/NodeBank.cpp File Reference

This is the implementation of the NodeBank class.

```
#include "NodeBank.h"
```

4.16.1 Detailed Description

This is the implementation of the NodeBank class.

Author

Bryce Monaco

This file contains the implementation of the NodeBank class

Version

1.0

Note

This is essentially an early prototype of a Client

4.17 PA05/NodeBank.h File Reference

This is the header of the NodeBank class.

```
#include <iostream>
```

Classes

- class NodeBank

4.17.1 Detailed Description

This is the header of the NodeBank class.

Author

Bryce Monaco

This file contains the header of the NodeBank class

Version

1.0

Note

This is essentially an early prototype of a Client

4.18 PA05/PA05.cpp File Reference

This is the main driver of PA05.

```
#include <iostream>
#include <fstream>
#include "LinkQueueBank.h"
#include "ArrayQueueBank.h"
#include "CountingSort.h"
#include "Client.h"
#include <cstdlib>
#include <time.h>
```

Functions

- void ReadInLine (Client ∗sentClients, int amount)

  Reads in values from a file and stores them as clients.
- void OutputLine (Client ∗sentClients, int amount)

  Prints the values of a client.
- void GenerateValues (int amount)

  Generates a certain amount of random values and stores them in a file.
- int **main** ()

4.18.1 Detailed Description

This is the main driver of PA05.

Author

Bryce Monaco

This file is the main driver of PA05.

Version

1.0

Note

Because the simulation is incomplete this file simply generates random clients, sorts them by arrival, and outputs them for debugging.

4.18.2 Function Documentation

4.18.2.1 void GenerateValues (int *amount*)

Generates a certain amount of random values and stores them in a file.

This function generates a certain amount of random values and them dumps them into a file for easy reference later

Algorithm Generates random values into an array, then traverses the array and outputs the values to a file.

Parameters

`in`	*amount*	The amount of values to generate.
`out`	*Creates*	ten files each populated with a certain amount of random values.

Returns

None.

Note

This function modified from PA04, this version generates two values separated by a space, the arrival time (0-100000) and the transaction time (0-100)

4.18.2.2 void OutputLine (Client ∗ *sentClients,* int *amount*)

Prints the values of a client.

Runs through the array and prints the values of each clients.

Algorithm None.

Parameters

`in`	*sentClients*	The address of the client array
`in`	*amount*	The amount of clients to output in.
`out`	*None.*	

Returns

None.

Note

None.

4.18.2.3 void ReadInLine (Client ∗ *sentClients,* int *amount*)

Reads in values from a file and stores them as clients.

Reads in the randomly generated values from a file and stores them as Client objects in the array.

Algorithm None.

Parameters

`in`	*sentClients*	The address of the client array
`in`	*amount*	The amount of clients to read in.
`out`	*The*	Client array is now populated.

Returns

None.

Note

None.

4.19 PA05/Simulation1A.cpp File Reference

This is the implementation of the Simulation1A class.

```
#include "Simulation1A.h"
```

4.19.1 Detailed Description

This is the implementation of the Simulation1A class.

Author

Bryce Monaco

This file contains the implementation of the Simulation1A class

Version

1.0

Note

The implementation of this class is incomplete.

4.20 PA05/Simulation1A.h File Reference

This is the header of the Simulation1A class.

```
#include <iostream>
#include "ArrayQueueBank.h"
#include "ArrayQueue.h"
#include "Client.h"
```

Classes

- class Simulation1A

4.20.1 Detailed Description

This is the header of the Simulation1A class.

Author

Bryce Monaco

This file contains the header of the Simulation1A class

Version

1.0

Note

The implementation of this class is incomplete.

Index