

**Bankers Algorithm Reference Manual**

**By Cody Precord**

## Contents

# 1 Bankers Algorithm Using Threads

## 1.1 Introduction and Overview

Bankers Algorithm This program uses multiple threads of execution to simulate the Bankers Algorithm. The Bankers Algorithm is a common problem used to illustrate thread management through a simulation of having many customers who need resources from a bank, and a bank that has a set number of resources to provide those customers. If a customers request is granted the customer then holds the resources until all of his/her needs have been met, then the customer returns all resources to the bank.

This presents a problem of the possibility of Thread Lock if resources are granted in such a manner that prevents any customer from fulfilling its needs, because other customers are holding resources that are needed. This problem leads to the need for managing Thread Safety during the execution of the program to closely manage the allocation of resources as to avoid this dilemma.

## 1.2 Program Usage

Compile each of the three source files ([Bank.java](#), [Customer.java](#), and [RunBank.java](#))

```
dev@tux> javac Bank.java
          javac Customer.java
          javac RunBank.java
```

Then simply execute [RunBank](#)

```
dev@tux> java RunBank
```

# 2 Bankers Algorithm Class Index

## 2.1 Bankers Algorithm Class List

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

<a href="#">Bank</a> (Class to Simulate a bank and its resources )	??
<a href="#">Customer</a> (Describes a customer and their actions )	??
<a href="#">RunBank</a> (Simulates the interactions between the Customers and the <a href="#">Bank</a> )	??

# 3 Bankers Algorithm File Index

## 3.1 Bankers Algorithm File List

Here is a list of all files with brief descriptions:

<a href="#">Bankers_Algorithm/Bank.java</a>	??
---	----

[Bankers\\_Algorithm/Customer.java](#) ??

[Bankers\\_Algorithm/RunBank.java](#) ??

## 4 Bankers Algorithm Class Documentation

### 4.1 Bank Class Reference

Class to Simulate a bank and its resources.

#### Public Member Functions

- [Bank](#) (int [resources](#), int [customers](#))  
*Constructor for a Bank Class object.*
- void [printResourceSet](#) ()  
*Prints the Banks available resources matrix.*
- void [printAllocationSet](#) ()  
*Prints the Bank's Allocation Matrix.*
- boolean [checkDone](#) (int ID)  
*Checks if all requests for a certain customer have been satisfied. /\*! [checkDone\(int ID\)](#) recieves a customers ID number and uses that ID number to look up the customers requestSet[][] to see if all the customers requests have been granted or not. If they have the function returns TRUE else it will return FALSE to indicate that the customer still needs more resources.*
- synchronized int [request](#) (int ID, int requestsMade)  
*Handles much of the dirty work of issuing resources and checking thread safety.*
- synchronized void [returnResources](#) (int ID)  
*Returns all of a customer's resources to the bank.*

#### Private Attributes

- int [resources](#)  
*Number of different Resources the Bank has.*
- int [customers](#)  
*Number of Customers at the Bank.*
- int [maxSet](#) [ ][ ]
- int [requestSet](#) [ ][ ]  
*Matrix of requests each [Customer](#) needs filled.*
- int [allocationSet](#) [ ][ ]  
*Matrix of resources allocated to each customer.*

- int `initialAvailSet` []  
*Initial set of available Resources at the Bank.*
- int `currentAvailSet` []  
*Set of Currently available Resources at the Bank.*
- int `requestingSet` []  
*Set of requests the *Customer* is currently making.*

#### 4.1.1 Detailed Description

Class to Simulate a bank and its resources.

The Bank Class tracks it available resources sets, and recieves requests from customers. As the requests come in the bank does an analysis of what it has availble compared to what the needs are of its customers and allocates its resources in such a way to avoid leaving any of its customers unsatisfied

Definition at line 26 of file Bank.java.

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 Bank.Bank (int resources, int customers)

Constructor for a Bank Class object.

Constructor for the Bank Class recieves a number of resources and a number of customers. From these numbers the constructor creates a set of random numbers for the Set of available resources and the set of maximum resources each customer will request. It also initializes the allocationSet to zero's

Definition at line 46 of file Bank.java.

References allocationSet, currentAvailSet, initialAvailSet, maxSet, and requestSet.

#### 4.1.3 Member Function Documentation

##### 4.1.3.1 boolean Bank.checkDone (int ID)

Checks if all requests for a certain customer have been satisfied. /\*! `checkDone(int ID)` recieves a customers ID number and uses that ID number to look up the customers requestSet[][] to see if all the customers requests have been granted or not. If they have the function returns TRUE else it will return FALSE to indicate that the customer still needs more resources.

##### Parameters:

*ID* A Customers ID number

##### Returns:

True If all requests in the customers request set have been satisfied  
False If not all requests have been satisfied

Definition at line 145 of file Bank.java.

Referenced by Customer.run().

#### 4.1.3.2 void Bank.printAllocationSet ()

Prints the Bank's Allocation Matrix.

Prints the Banks Current Allocation Matrix to the consol.

Definition at line 120 of file Bank.java.

References allocationSet, customers, and resources.

Referenced by RunBank.main().

#### 4.1.3.3 void Bank.printResourceSet ()

Prints the Banks available resources matrix.

Prints the Banks currently available resource matrix set to the consol. By printing each type in a differnt row.

Definition at line 106 of file Bank.java.

References initialAvailSet, and resources.

Referenced by RunBank.main().

#### 4.1.3.4 synchronized int Bank.request (int ID, int requestsMade)

Handles much of the dirty work of issuing resources and checking thread safety.

[request\(int ID, int requestsMade\)](#) recieves two parameters a customer ID and the number of requests that customer has made. This function handles the customers requests by first initializing a random resource request based on the customers max requestSet[][].

The function then checks the currentAvailSet[] to see if the request is possible to grant if not the [request\(\)](#) function breaks and updates the customers number of requests made and outputs that the request was not granted. If the [request\(\)](#) is possible to grant the function first checks for a safe sequence and if one cannot be found the function does as above. Else it will add the requested values to the customers allocationSet[[[]], update the currentAvailSet[] and the customers number of requests made.

##### Parameters:

*ID* A Customers ID number

*requestsMade* Number of requests the customer has made

##### Returns:

int The number of requestsMade + 1

Definition at line 173 of file Bank.java.

Referenced by Customer.run().

#### 4.1.3.5 synchronized void Bank.returnResources (int ID)

Returns all of a customer's resources to the bank.

[returnResources\(int ID\)](#) receives a customer ID number as a parameter and returns all that customers held resources to the currentAvailSet[]. It also updates the allocationSet to show that that customer is not holding any more resources.

##### Parameters:

*ID* A customers ID number

**Returns:**

void

Definition at line 303 of file Bank.java.

Referenced by Customer.run().

**4.1.4 Member Data Documentation****4.1.4.1 int Bank.allocationSet[][]** [private]

Matrix of resources allocated to each customer.

Definition at line 33 of file Bank.java.

Referenced by Bank(), and printAllocationSet().

**4.1.4.2 int Bank.currentAvailSet[]** [private]

Set of Currently available Resources at the Bank.

Definition at line 36 of file Bank.java.

Referenced by Bank().

**4.1.4.3 int Bank.customers** [private]

Number of Customers at the Bank.

Definition at line 29 of file Bank.java.

Referenced by printAllocationSet().

**4.1.4.4 int Bank.initialAvailSet[]** [private]

Initial set of available Resources at the Bank.

Definition at line 35 of file Bank.java.

Referenced by Bank(), and printResourceSet().

**4.1.4.5 int Bank.maxSet[][]** [private]

Definition at line 31 of file Bank.java.

Referenced by Bank().

**4.1.4.6 int Bank.requestingSet[]** [private]

Set of requests the Customer is currently making.

Definition at line 37 of file Bank.java.

**4.1.4.7 int Bank.requestSet[][]** [private]

Matrix of requests each Customer needs filled.

Definition at line 32 of file Bank.java.

Referenced by Bank().

**4.1.4.8** `int Bank.resources` [`private`]

Number of different Resources the Bank has.

Definition at line 28 of file `Bank.java`.

Referenced by `printAllocationSet()`, and `printResourceSet()`.

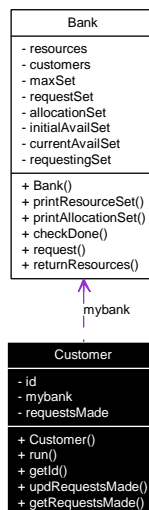
The documentation for this class was generated from the following file:

- `Bankers_Algorithm/Bank.java`

**4.2 Customer Class Reference**

Describes a customer and their actions.

Collaboration diagram for Customer:

**Public Member Functions**

- `Customer` (`int i`, `Bank b`)  
*Class Constructor for Customer.*
- `void run ()`  
*Run implimentation for the Bankers Algorithm Project.*
- `int getId ()`  
*Returns a Customer ID.*
- `void updRequestsMade (int newReq)`  
*Updates number of request made by the customer to the bank.*
- `int getRequestsMade ()`  
*Returns requests made.*

### Private Attributes

- `int id`  
*Holds the customers ID number.*
- `Bank mybank`  
*Is the [Bank](#) that the customer is using.*
- `int requestsMade`  
*Holds the number of requests the customer has made.*

### 4.2.1 Detailed Description

Describes a customer and their actions.

Definition at line 21 of file Customer.java.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 `Customer.Customer (int i, Bank b)`

Class Constructor for Customer.

Definition at line 35 of file Customer.java.

References `mybank`, and `requestsMade`.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 `int Customer.getId ()`

Returns a Customer ID.

The function `getId()` returns the ID number of the customer that invoked the `getId()` function.

Definition at line 81 of file Customer.java.

References `id`.

#### 4.2.3.2 `int Customer.getRequestsMade ()`

Returns requests made.

Returns the number of requests that the customer has made to the [Bank](#)

Definition at line 101 of file Customer.java.

References `requestsMade`.

#### 4.2.3.3 `void Customer.run ()`

Run implimentation for the Bankers Algorithm Project.

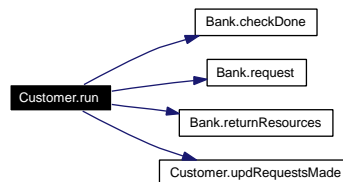
The `run()` function is invoked when a customer thread is started. The `run()` function takes care of making the customers requests and holding resources for a random amount of time between one and five seconds. It

also prior to making request checks if the customer's requests have been satisfied using the `checkDone(int id)` function. Once the customer has finished making requests the `run()` for that customer returns.

Definition at line 49 of file `Customer.java`.

References `Bank.checkDone()`, `mybank`, `Bank.request()`, `requestsMade`, `Bank.returnResources()`, and `updRequestsMade()`.

Here is the call graph for this function:



#### 4.2.3.4 void Customer.updRequestsMade (int newReq)

Updates number of request made by the customer to the bank.

The function `updRequestsMade(int newReq)` takes an integer value as an argument and updates the number of requests made of the customer that invoked the function to the value specified in the `updRequestsMade`'s `newReq` argument.

##### Parameters:

*newReq* Value to change number of requests made

##### Returns:

void

Definition at line 94 of file `Customer.java`.

References `requestsMade`.

Referenced by `run()`.

## 4.2.4 Member Data Documentation

### 4.2.4.1 int Customer.id [private]

Holds the customer's ID number.

Definition at line 23 of file `Customer.java`.

Referenced by `getId()`.

### 4.2.4.2 Bank Customer.mybank [private]

Is the `Bank` that the customer is using.

Definition at line 24 of file `Customer.java`.

Referenced by `Customer()`, and `run()`.

#### 4.2.4.3 int `Customer.requestsMade` [private]

Holds the number of requests the customer has made.

Definition at line 25 of file Customer.java.

Referenced by `Customer()`, `getRequestsMade()`, `run()`, and `updRequestsMade()`.

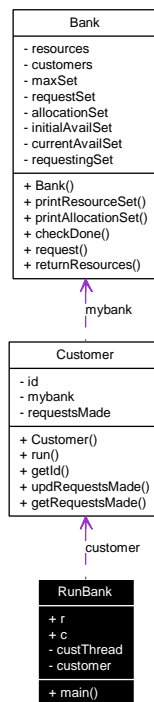
The documentation for this class was generated from the following file:

- Bankers\_Algorithm/[Customer.java](#)

### 4.3 RunBank Class Reference

The RunBank class simulates the interactions between the Customers and the [Bank](#).

Collaboration diagram for RunBank:



#### Static Public Member Functions

- static void `main` (String args[ ]) throws IOException  
*Main Method for the Bankers Algorithm Project.*

#### Static Public Attributes

- static int `r`  
*The number of Resources the [Bank](#) has, taken from System.in.*

- static int `c`

*The number of Customers the `Bank` has, taken from `System.in`.*

### Static Private Attributes

- static Thread[] `custThread`

*An array for holding `Customer` Threads.*

- static `Customer[]` `customer`

*An array for holding `Customers`.*

### 4.3.1 Detailed Description

The `RunBank` class simulates the interactions between the `Customers` and the `Bank`.

The `Runbank` Class simulates the ineractions that take place between the `Customer` and the `Bank` Classes. It contains the main method that initiates the parameters of execution and then starts the appropriate number of threads.

Definition at line 51 of file `RunBank.java`.

### 4.3.2 Member Function Documentation

#### 4.3.2.1 static void `RunBank.main (String args[])` throws `IOException` [static]

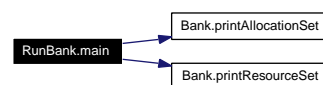
Main Method for the Bankers Algorithm Project.

Recieves input from the user and starts the appropriate number of `Customer` threads. Once all threads have finished execution the method prints the final Resource and Allocation sets.

Definition at line 63 of file `RunBank.java`.

References `c`, `customer`, `custThread`, `Bank.printAllocationSet()`, `Bank.printResourceSet()`, and `r`.

Here is the call graph for this function:



### 4.3.3 Member Data Documentation

#### 4.3.3.1 int `RunBank.c` [static]

The number of Customers the `Bank` has, taken from `System.in`.

Definition at line 56 of file `RunBank.java`.

Referenced by `main()`.

#### 4.3.3.2 `Customer [] RunBank.customer` [static, private]

An array for holding Customers.

Definition at line 54 of file RunBank.java.

Referenced by main().

#### 4.3.3.3 `Thread [] RunBank.custThread` [static, private]

An array for holding `Customer` Threads.

Definition at line 53 of file RunBank.java.

Referenced by main().

#### 4.3.3.4 `int RunBank.r` [static]

The number of Resources the `Bank` has, taken from System.in.

Definition at line 55 of file RunBank.java.

Referenced by main().

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

- Bankers\_Algorithm/[RunBank.java](#)

## 5 Bankers Algorithm File Documentation

### 5.1 Bankers\_Algorithm/Bank.java File Reference

#### Classes

- class [Bank](#)  
*Class to Simulate a bank and its resources.*

#### 5.1.1 Detailed Description

Contains the implementation of the `Bank` Class which is the holder of resources in the Bankers Algorithm. It generates the initial random resources and manages the thread safety of the transactions that take place.

Definition in file [Bank.java](#).

### 5.2 Bankers\_Algorithm/Customer.java File Reference

#### Classes

- class [Customer](#)  
*Describes a customer and their actions.*

### 5.2.1 Detailed Description

Simulates a customers action while at a [Bank](#). It also contains imlmentation of run() for this project.

Definition in file [Customer.java](#).

## 5.3 Bankers\_Algorithm/RunBank.java File Reference

### Classes

- class [RunBank](#)

*The RunBank class simulates the interactions between the Customers and the [Bank](#).*

### 5.3.1 Detailed Description

Holds the definition of the [RunBank](#) Class which through the use of multiple threads of execution simulates the transactions of customers as the come and go from the bank.

Definition in file [RunBank.java](#).