Main Classes:

**TransactionManager:**

Purpose: Handles all Transaction operations. Takes requests for reads and writes on variables and sends them to the appropriate site. It will use the SiteManager for shared functions to access Sites and the DeadlockManager to detect and kill deadlocks.

Dependencies:

* SiteManager
* DeadlockManager

Functions:

* Execute(operations): takes a list of operations that are running in that tick and executes them. At the beginning of every tick, all waiting transactions are rerun
* BeginTransaction(operation): creates a new transaction from the operation
* ReadVariable(operation): TM will find the sites the variable is located. It will try each site until it can get a lock and then read the value
* WriteVariable(operation): TM will find the sites the variable is located. It will try to get a lock on all sites. Once all sites are locked it will write the variable.
* EndTransaction(operation): TM will decide whether the transaction can commit. It then appropriately aborts or commits the transaction.

**SiteManager:**

Purpose: Stores a list of the sites and handles the site-specific operations (fail, recover, etc). Will initiate all the sites and their variables.

Functions:

* Execute(operations): takes a list of operations for sites and runs them in the same tick.
* Fail(site): Gets the site and tells it to fail
* Recover(site): Gets the site and tells it to recover
* Dump(site, variable): Dumps the variables at the sites. Can specify the specific site or the specific variable
* GetSites(): Helper function for other classes to get the sites
* GetSitesWithVariable(variable): Helper function for other classes to get the sites where the variable is located

**DeadlockManager:**

Purpose: Detects and kills deadlocks. Used by the TransactionManager to allow continuation of transaction operations.

Dependencies:

* SiteManager

Functions:

* DetectDeadlocks(transaction): constructs a waits-for graph of all activate transactions. It will then detect cycles and kill the youngest transaction

**Site:**

Purpose: Stores the data and allows access to transactions sent to it. Uses the LockManager to determine if transaction can get locks or not

Dependencies:

* LockManager

Functions:

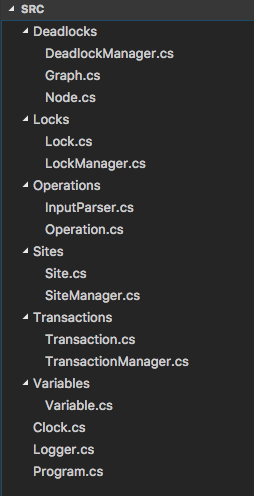
* ReadData(transaction, variable): returns the value of the variable. Depending on the transaction, will send data from a certain time, currently committed data, or written but not yet committed data
* WriteData(transaction, variable, value): writes to a variable as a transaction. Doesn’t commit the data, but writes to a temporary location
* CommitData(transaction): commits all variable values that this transaction has written to

**LockManager:**

Purpose: A separate instance manages the locks at each site. It will distribute or deny locks as requested by the site.

Functions:

* GetReadLock(transaction, variable): attempts to get a read lock on the variable for the running transaction. Will create the lock and return true or false if it was successful.
* GetWriteLock(transaction, variable): attempts to get a write lock on the variable for the running transaction. Will create the lock and return true or false if it was successful.
* ClearLocks(transaction): Clears the locks at the site for the current transaction



LockManager

Site

SiteManager

DeadLockManager

TransactionManager