A tutorial on transactions and locks

The text "An introduction to O-O systems development with Jade" only mentions in passing the issues of contention for objects and Jade's locks of various types.

But these are serious matters for the well-being of non-trivial systems. Therefore, with the permission of Jade Software Corporation, we reproduce here their tutorial on the topic which is supported by the TransactionsAndLocks schema, which is on the text's cd-rom as \Demos\Dm31_TransactionsAndLocks.

For further information refer to the Jade Help system. Look at "Jade Locking Overview", chapter 9 of the Developer's Reference.

Background

The need for locks

Although the database—the repository—may be seen as a persistent disk file or files it is part of the design of Jade that objects in use be copied into 'caches' in local RAM where they can be accessed without either the delays of disk access or the latency of data transport across a network.

However, once and while multiple copies of an object exist it is essential that there be mechanisms to ensure that no object is released to a requesting process until it is free to be released and that any requestor gets the latest state of an object.

Every object has an .edition attribute. Any process seeking to access an object wants the latest edition, which is incremented when the update is committed.

A system of 'locks' manages this process. In any system this system must work reliably, the more so in systems with multiuser accesses to objects and with objects of high volatility. These locks will be imposed automatically by the system and can also be imposed 'manually' by an application. Locks are recorded and managed in tables in the application, the node (client and server) and by the system as a whole.

When accessing objects without locking, a process can access an object even when another process is updating that object. The process does not see the updates, but instead views the previous (the most recently committed) edition of the objects.

There are three types of lock.

To manually lock objects you can use the exclusiveLock, sharedLock, and reserveLock methods of the Object class.

Exclusive Locks

Before an object can be updated, it must have an exclusive lock placed on it. When you lock an object using an exclusive lock, no other process can lock the same object. JADE automatically applies an exclusive lock when an object is updated. By default, updated objects are locked for the duration of the transaction.

Shared Locks

If you lock an object using a shared lock, other processes attempting to update the object or explicitly acquire an exclusive lock wait until the lock is released but can acquire a shared lock or a reserve lock.

Reserve Locks

A reserve lock is available for situations where you intend to update an object but you need to minimize the length of time the object is locked with an exclusive lock. When you place a reserve lock on an object, other processes attempting to acquire an exclusive lock or a reserve lock on that same object wait until the reserve lock is relinquished, but those processes attempting to acquire a shared lock succeed.

Combinations

The valid concurrent lock combinations are shown in the following table.

Rank	Type	Exclusive	Share	Reserve
Highest	**Exclusive**	***No***	***No***	***No***
Middle	**Reserve**	***No***	***Yes***	***No***
Lowest	**Shared**	***No***	***Yes***	***Yes***

From this we see that for an object, at any time:

- Only one exclusive lock is allowed. (Since it is an exclusive lock no other lock is allowed.)
- More than one shared lock can coexist.
- A reserve lock is incompatible with another reserve lock.

Volatility

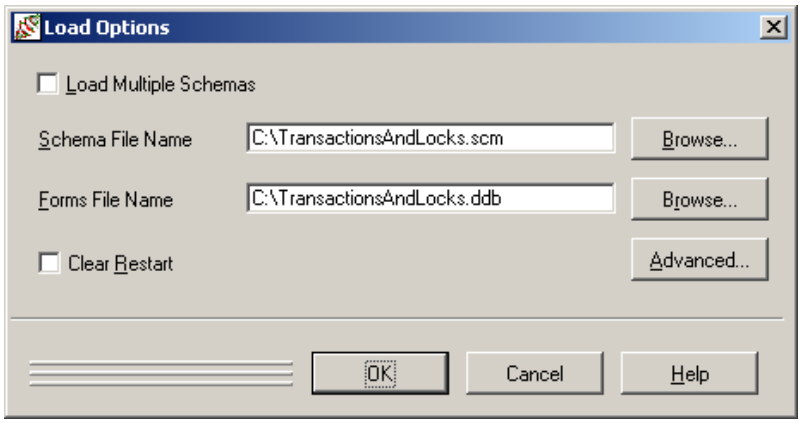
The efficiency of the system can be aided by advice. When a class is added in the Class Browser the Add Class dialog offers a Volatility tab bearing three 'Instance volatility' values: Volatile, Stable and Frozen.[1] Naturally, the default is Volatile.

Automatic locking activity by the system

The locking of a collection when the foreach command is being used is the classic example.

A tutorial on transactions and locks

Load the **Dm31_TransactionsAndLocks** schema into your JADE environment using the Schema | Load . . . menu option.

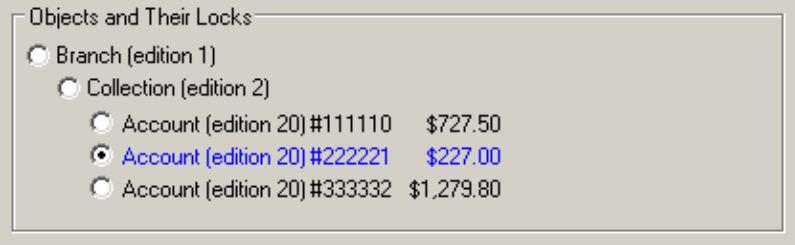


You are now ready to follow the steps in the tutorial.

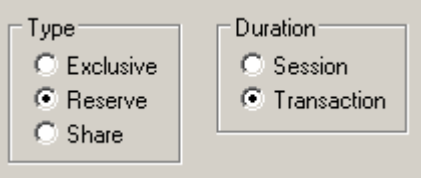
How to lock and unlock an object

1. Run the default application in the **Dm31_TransactionsAndLocks** schema.
2. First select the object to lock.

[1] Frozen objects cannot be updated (until their volatility status is altered.)



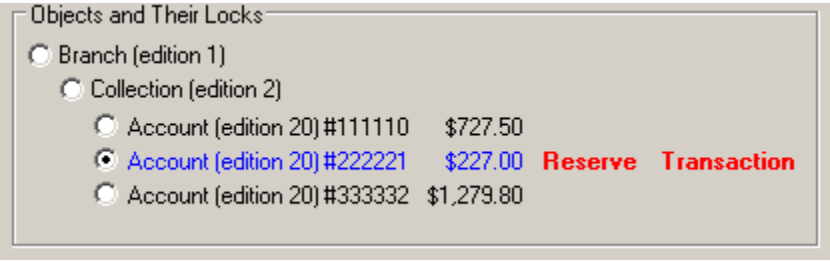
3. Select the lock Type and Duration.



4. Click the Lock button to apply the lock.

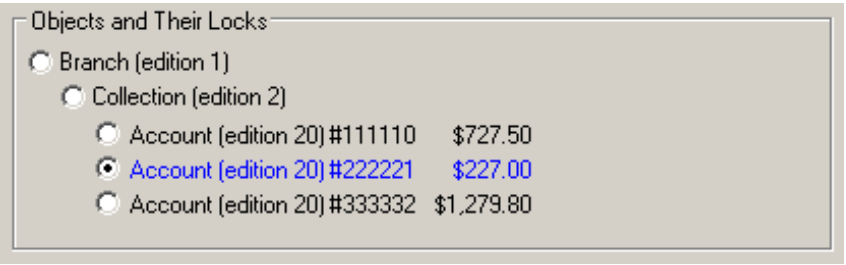


The lock status is displayed.



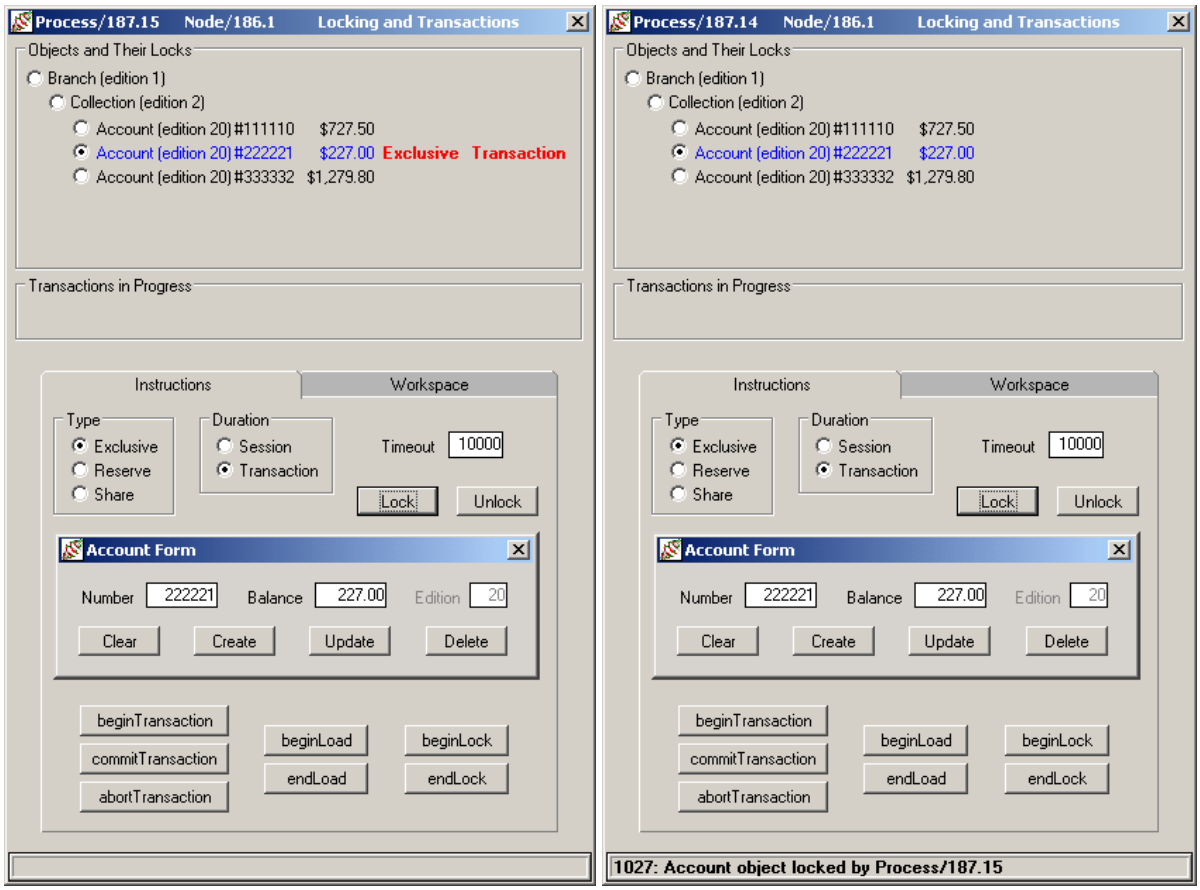
5. Click the Unlock button to release the lock on the object.





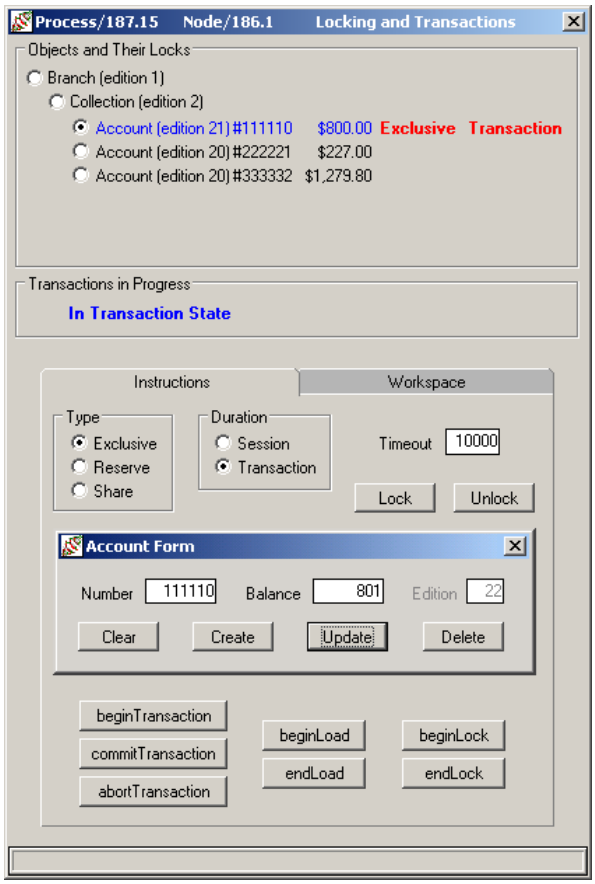
How to create a lock exception

Place an exclusive, transaction duration lock on the second account object on the process on the left. Now try to place a similar lock on the same object in right process. A lock exception is raised and is visible on the status line after the timeout period has expired.



How to carry out an update transaction

6. Select an account to update.
7. Click the beginTransaction button.
8. Change the *balance* by adding an extra dollar.
9. Click the Update button and notice the object is locked.



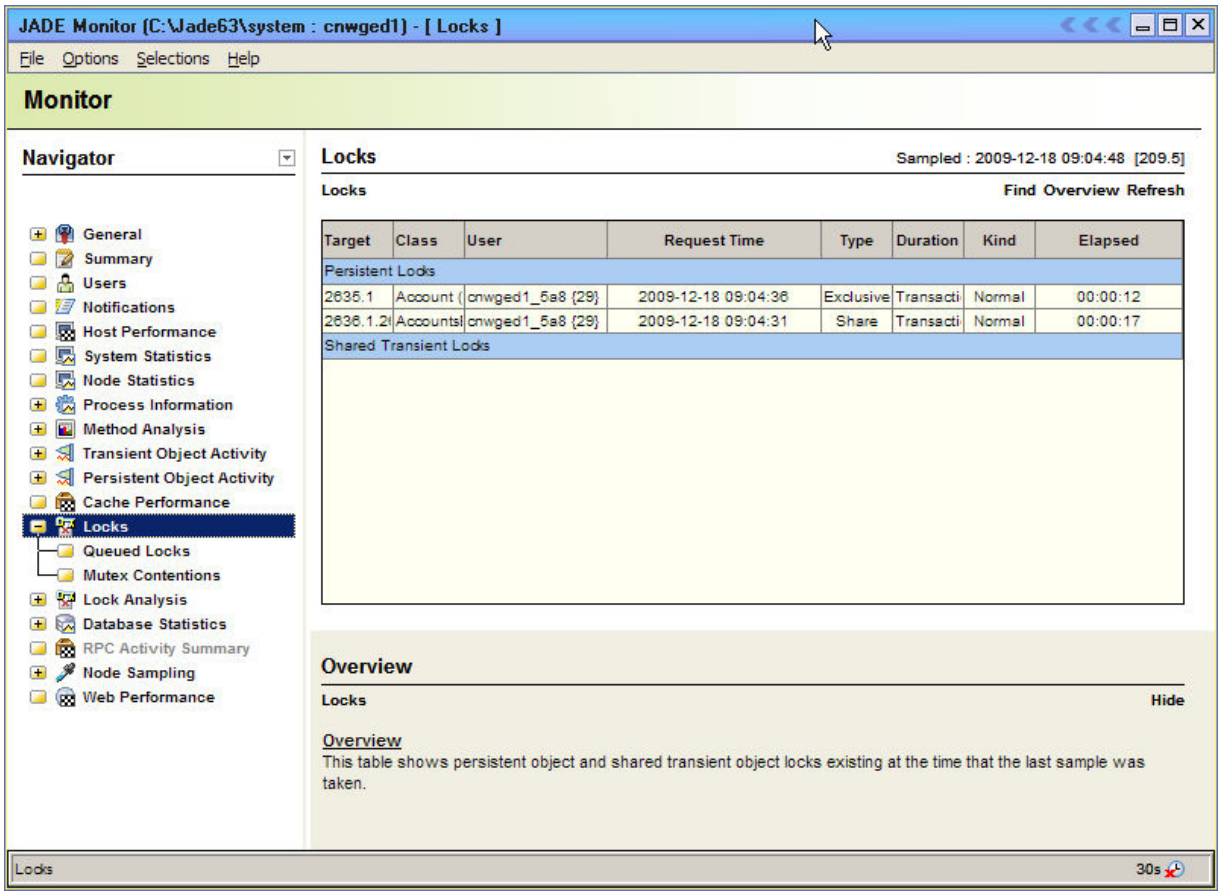
10. To complete the transaction, click the commitTransaction button. Notice the object edition is incremented.

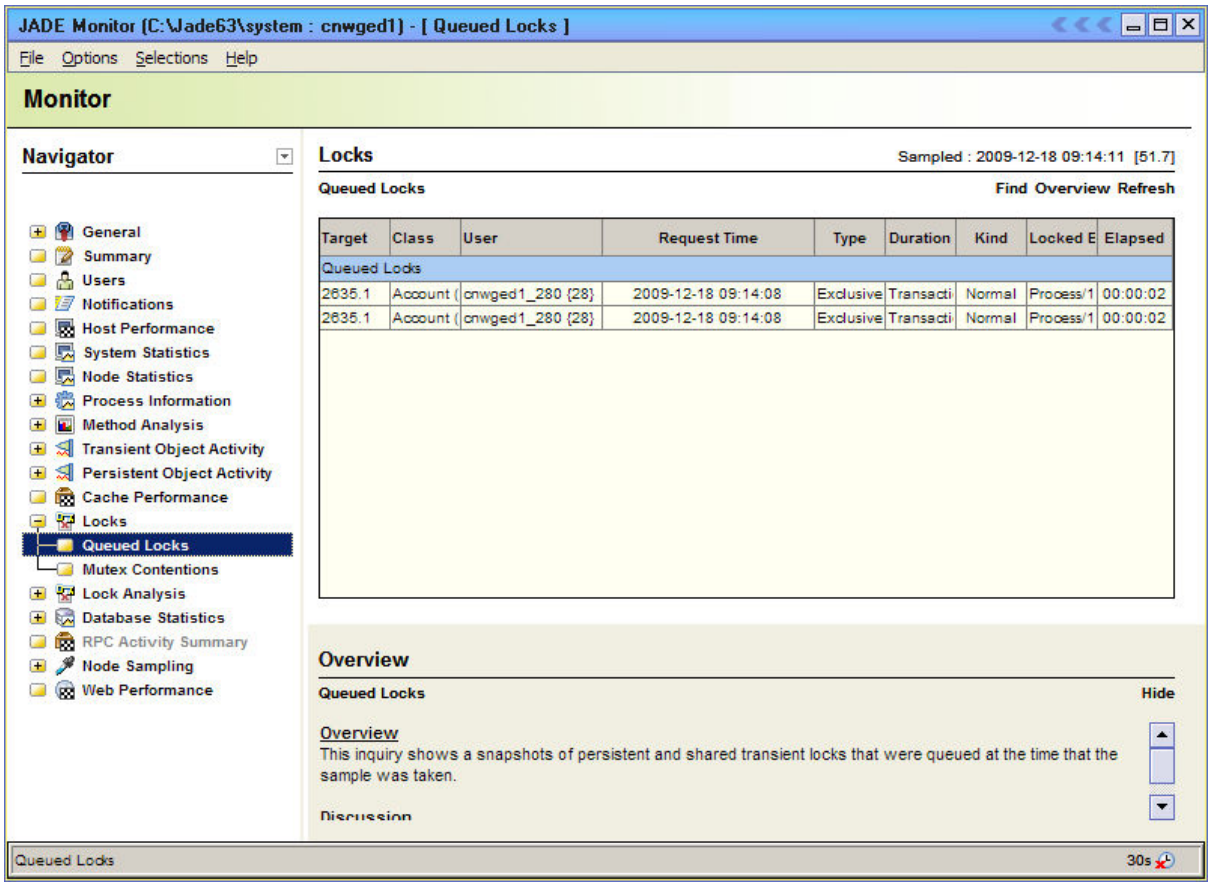
11. Repeat the above process but first use the process on the right to place a share lock on the object.

How to monitor locks and queued locks

12. From the JADE development environment run the monitor by clicking the  icon.

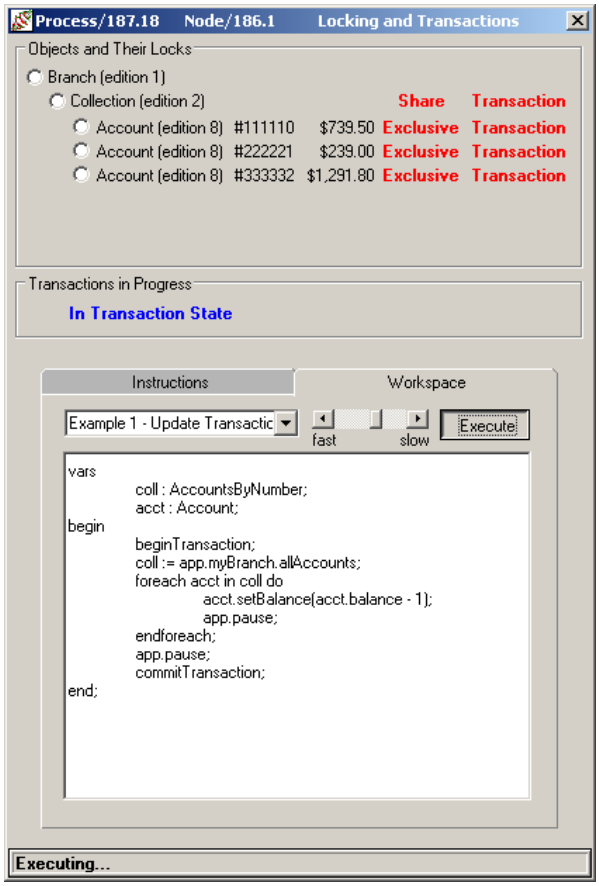
13. Select the Locks and Queued Locks windows.


JADE Monitor (C:\Jade63\system : cnwged1) - [Locks]
File Options Selections Help
Monitor
Navigator
General
Summary
Users
Notifications
Host Performance
System Statistics
Node Statistics
Process Information
Method Analysis
Transient Object Activity
Persistent Object Activity
Cache Performance
Locks
Queued Locks
Mutex Contentions
Lock Analysis
Database Statistics
RPC Activity Summary
Node Sampling
Web Performance
Locks
Sampled : 2009-12-18 09:04:48 [209.5]
Locks
Find Overview Refresh
Target Class User Request Time Type Duration Kind Elapsed
Persistent Locks
2635.1 Account (cnwged1_5a8 {29} 2009-12-18 09:04:36 Exclusive Transacti Normal 00:00:12
2636.1.2 Accounts cnwged1_5a8 {29} 2009-12-18 09:04:31 Share Transacti Normal 00:00:17
Shared Transient Locks
Overview
Locks
Hide
Overview
This table shows persistent object and shared transient object locks existing at the time that the last sample was taken.
Locks
30s


JADE Monitor (C:\Jade63\system : cnwged1) - [Queued Locks]
File Options Selections Help
Monitor
Navigator
General
Summary
Users
Notifications
Host Performance
System Statistics
Node Statistics
Process Information
Method Analysis
Transient Object Activity
Persistent Object Activity
Cache Performance
Locks
Queued Locks
Mutex Contentions
Lock Analysis
Database Statistics
RPC Activity Summary
Node Sampling
Web Performance
Locks
Sampled : 2009-12-18 09:14:11 [51.7]
Queued Locks
Find Overview Refresh
Target Class User Request Time Type Duration Kind Locked E Elapsed
Queued Locks
2635.1 Account (cnwged1_280 {28} 2009-12-18 09:14:08 Exclusive Transacti Normal Process/1 00:00:02
2635.1 Account (cnwged1_280 {28} 2009-12-18 09:14:08 Exclusive Transacti Normal Process/1 00:00:02
Overview
Queued Locks
Hide
Overview
This inquiry shows a snapshots of persistent and shared transient locks that were queued at the time that the sample was taken.
Discussion
Queued Locks
30s

How to execute code and observe locking

14. Click the *Workspace* tab and select **Example 1 – Update Transactions (1)** from the combo box.



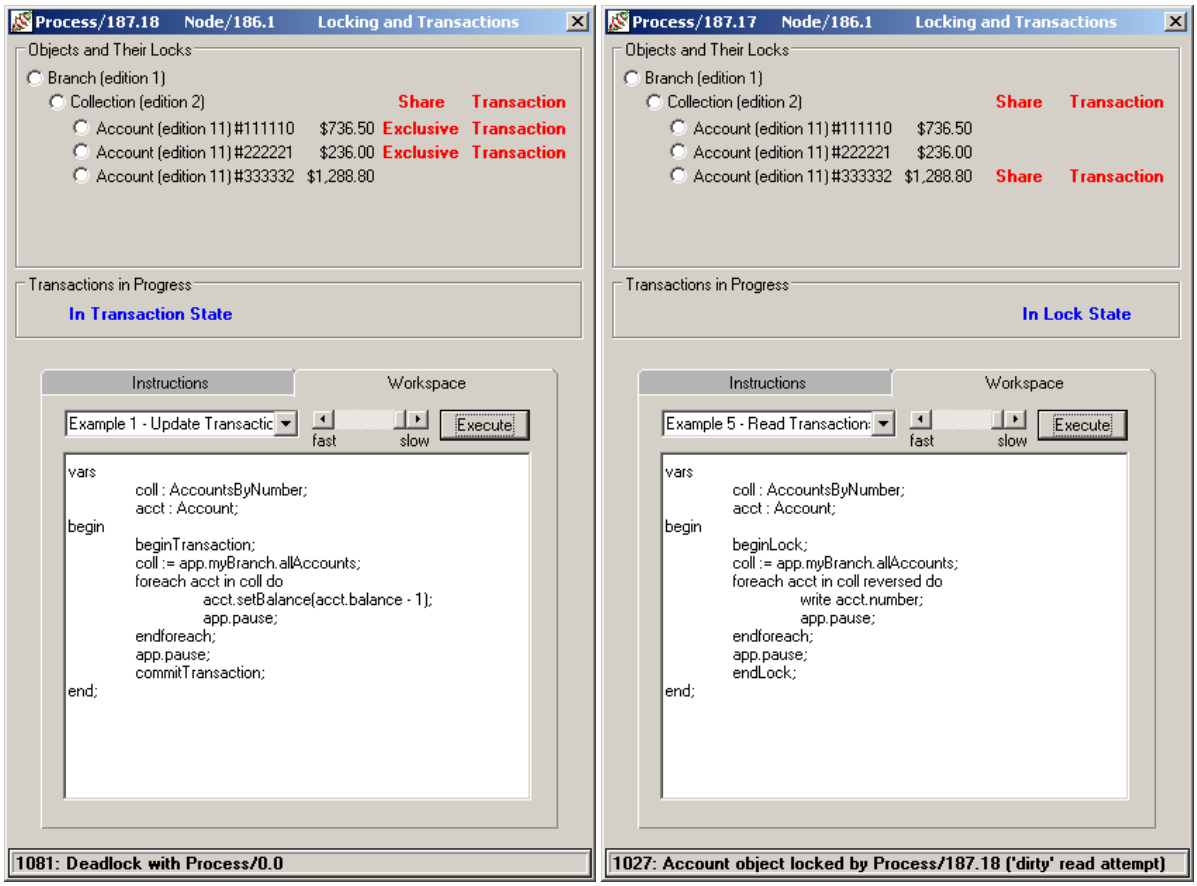
15. Click the **Execute** button to run the code.

16. Try several of the methods from the list.

How to cause a deadlock exception

17. Select **Example 5- Read Transactions (3) from the combo box** on the right process. This method traverses the accounts collection in reverse order.

18. Set the speed scroll bar to very slow. Execute both the methods and watch the result.


Process/187.18 Node/186.1 Locking and Transactions
Objects and Their Locks
Branch (edition 1)
Collection (edition 2) Share Transaction
Account (edition 11) #111110 $736.50 Exclusive Transaction
Account (edition 11) #222221 $236.00 Exclusive Transaction
Account (edition 11) #333332 $1,288.80
Transactions in Progress
In Transaction State
Instructions
Workspace
Example 1 - Update Transactic
fast
slow
Execute
vars
coll : AccountsByNumber;
acct : Account;
begin
beginTransaction;
coll := app.myBranch.allAccounts;
foreach acct in coll do
acct.setBalance(acct.balance - 1);
app.pause;
endforeach;
app.pause;
commitTransaction;
end;
1081: Deadlock with Process/0.0
Process/187.17 Node/186.1 Locking and Transactions
Objects and Their Locks
Branch (edition 1)
Collection (edition 2) Share Transaction
Account (edition 11) #111110 $736.50
Account (edition 11) #222221 $236.00
Account (edition 11) #333332 $1,288.80 Share Transaction
Transactions in Progress
In Lock State
Instructions
Workspace
Example 5 - Read Transaction
fast
slow
Execute
vars
coll : AccountsByNumber;
acct : Account;
begin
beginLock;
coll := app.myBranch.allAccounts;
foreach acct in coll reversed do
write acct.number;
app.pause;
endforeach;
app.pause;
endLock;
end;
1027: Account object locked by Process/187.18 ('dirty' read attempt)