Appendix B: Source Code of Sample Transactions

In Subsection 7.1.1, we described the implementation of a sample enterprise application. The enterprise application has been built on top of our *Free Data Objects* framework and executes six different types of transactions on business data objects. Below, the Java source code for each transaction type is shown. All six code sections use a variable *om*, which references the local object manager component. Also, a variable `rand` is used, which references a random number generator.

Transaction type 1: Insertion of a new landlord instance.

```
01: try {
02:    om.beginTx();
03:    Landlord ll = (Landlord) om.createNewDataObject( Landlord.class );
04:    ll.setName( "Lan D. Lord-" + rand.nextInt(10000) );
05:    ll.setAddress( "Takustrasse " + rand.nextInt(100)
06:       + ", Berlin, Germany" );
07:    ll.setEmail( "meyer@abc.fu-berlin.de" );
08:    ll.setPhone( "++49-30-838-98765" + rand.nextInt(1000) );
09:    om.commitTx();
10: } catch( FDOException ex ) { om.rollbackTx(); }
```

Transaction type 2: Query all landlord instances and then add a new accommodation instance for a random landlord.

```
01: try {
02:    om.beginTx();
03:
04:    // query all landlord instances
05:    List results = om.query( "SELECT FROM Landlord" );
06:    if( results.size() > 0 ) {
07:       int pos = rand.nextInt( results.size() );
08:       Landlord ll = (Landlord) results.get( pos );
09:
10:       // insert a new accommodation instance
11:       Accommodation a = (Accommodation) om.createNewDataObject(
12:          Accommodation.class );
13:       a.setCountryCode( new Short( (short) rand.nextInt( 100) ) );
14:       a.setAccommodationType( new Short( (short) 8) );
15:       a.setDescription( "Nice B&B in the center of the city" );
16:       a.setCoordX( new Double( rand.nextDouble() ) );
17:       a.setCoordY( new Double( rand.nextDouble() ) );
18:       a.setLandlord( ll );
19:
20:       // insert an empty reservation for our new accommodation
21:       Reservation r = (Reservation) om.createNewDataObject(
22:          Reservation.class );
23:       r.setAccommodation( a );
24:       r.setReservedBy( null );            // = not reserved yet
25:       r.setFirstDay( new Integer(0) );
26:       r.setLastDay( new Integer( 999999 ) );
27:
28:    } else {
29:       System.out.println( "no landlord found, not inserted" );
30:    }
31:    om.commitTx();
32: } catch( FDOException ex ) { om.rollbackTx(); }
```

Transaction type 3: Insertion of a new customer instance.

```
01: try {
02:    om.beginTx();
03:    Customer c = (Customer) om.createNewDataObject( Customer.class );
04:    c.setName( "T. U. Rist-" + rand.nextInt(10000) );
05:    c.setAddress( "Unter den Linden " + rand.nextInt(100)
06:       + ", Berlin, Germany" );
07:    c.setContact( "only after 5pm, phone:" + rand.nextInt(10000000) );
08:    c.setIsPremiumCustomer( new Boolean(false) );
09:    c.setCustomerNo( new Integer( rand.nextInt(100000) ) );
10:    om.commitTx();
11: } catch( FDOException ex ) { om.rollbackTx(); }
```

Transaction type 4: Query all accommodation instances and all customer instances. Then reserve a random accommodation for a random customer for a random time period.

```
01: try {
02:    om.beginTx();
03:
04:    // get random accommodation instance
05:    List results = om.query( "SELECT FROM Accommodation" );
06:    if( results.size() > 0 ) {
07:       int pos = rand.nextInt( results.size() );
08:       Accommodation a = (Accommodation) results.get( pos );
09:
10:       // calculate random values for firstDay and lastDay
11:       int firstDay = rand.nextInt(100);
12:       int lastDay = firstDay + rand.nextInt(14);
13:
14:       // try to find a reservation instance for our accommodation that
15:       // spans (firstDay, lastDay) and has no associated customer
16:       results = om.query( "SELECT FROM Reservation WHERE accommodation="
17:          + a.getRef().oid + " AND firstDay<=" + firstDay
18:          + " AND lastDay>=" + lastDay + " AND reservedBy IS NULL" );
19:       if( results.size() == 1 ) {
20:          Reservation r = (Reservation) results.get( 0 );
21:
22:          // split reservation instance, if necessary, on its left
23:          // and/or right side
24:          int oldFirstDay = r.getFirstDay().intValue();
25:          int oldLastDay = r.getLastDay().intValue();
26:          if( oldFirstDay < firstDay ) {
27:             Reservation rLeft = (Reservation) om.createNewDataObject(
28:                Reservation.class );
29:             rLeft.setAccommodation( a );
30:             rLeft.setReservedBy( null );
31:             rLeft.setFirstDay( new Integer( oldFirstDay ) );
32:             rLeft.setLastDay( new Integer( firstDay-1 ) );
33:          }
34:          if( oldLastDay > lastDay ) {
35:             Reservation rRight = (Reservation) om.createNewDataObject(
36:                Reservation.class );
37:             rRight.setAccommodation( a );
38:             rRight.setReservedBy( null );
39:             rRight.setFirstDay( new Integer( lastDay+1 ) );
40:             rRight.setLastDay( new Integer( oldLastDay ) );
41:          }
42:
43:          // get random customer instance
44:          results = om.query( "SELECT FROM Customer" );
45:          if( results.size() > 0 ) {
46:             pos = rand.nextInt( results.size() );
47:             Customer c = (Customer) results.get( pos );
48:
49:             // write our reservation data
50:             r.setReservedBy( c );
51:             r.setFirstDay( new Integer( firstDay ) );
52:             r.setLastDay( new Integer( lastDay ) );
53:
54:          } else {
```

```
55:             System.out.println( "no customer found, not reserved" );
56:          }
57:
58:       } else {
59:          System.out.println( "already reserved for that time" );
60:       }
61:
62:    } else {
63:       System.out.println( "no accommodation found, not reserved" );
64:    }
65:    om.commitTx();
66: } catch( FDOException ex ) { om.rollbackTx(); }
```

Transaction type 5: Query all reservations and cancel a random reservation.

```
01: try {
02:    om.beginTx();
03:
04:    // get random reservation
05:    List results = om.query( "SELECT FROM Reservation”
06:       + “ WHERE reservedBy IS NOT NULL");
07:    if( results.size() > 0 ) {
08:       int pos = rand.nextInt( results.size() );
09:       Reservation r = (Reservation) results.get( pos );
10:       Accommodation a = r.getAccommodation();
11:       int firstDay = r.getFirstDay().intValue();
12:       int lastDay = r.getLastDay().intValue();
13:
14:       // merge reservation with left neighbor, if necessary
15:       results = om.query( "SELECT FROM Reservation”
16:          + “ WHERE reservedBy IS NULL AND accommodation="
17:          + a.getRef().oid + " AND lastDay=" + (firstDay-1) );
18:       if( results.size() == 1 ) {
19:          Reservation rLeft = (Reservation) results.get( 0 );
20:          r.setFirstDay( rLeft.getFirstDay() );
21:          rLeft.delete();
22:       }
23:
24:       // merge reservation with right neighbor, if necessary
25:       results = om.query( "SELECT FROM Reservation”
26:          + “ WHERE reservedBy IS NULL AND accommodation="
27:          + a.getRef().oid + " AND firstDay=" + (lastDay+1) );
28:       if( results.size() == 1 ) {
29:          Reservation rRight = (Reservation) results.get( 0 );
30:          r.setLastDay( rRight.getLastDay() );
31:          rRight.delete();
32:       }
33:
34:       // set reservation instance to "not reserved"
35:       r.setReservedBy( null );
36:
37:    } else {
38:       System.out.println( "no reservation found, not deleted" );
39:    }
40:    om.commitTx();
41: } catch( FDOException ex ) { om.rollbackTx(); }
```

Transaction type 6: Query all accommodation instances and change the description of a random accommodation.

```
01: try {
02:    om.beginTx();
03:    List results = om.query( "SELECT FROM Accommodation" );
04:    if( results.size() > 0 ) {
05:       int pos = rand.nextInt( results.size() );
06:       Accommodation a = (Accommodation) results.get( pos );
07:       a.setDescription( "Nice B&B-for more info call ++49-3083855"
08:          + rand.nextInt(1000) );
09:    } else {
10:       System.out.println( "no accommodation found, not changed" );
11:    }
12:    om.commitTx();
13: } catch( FDOException ex ) { om.rollbackTx(); }
```