```
1 /*
2 This program creates nine Add/Delete processes. Only
  thirty data entries can be held by the buffer at once. The
   buffer
3 will hold task until able to store data. The buffer
  releases task once room is created. The buffer holds two
  arrays one
4 holding branches and the other holding values.
5  */
6 import java.io.File;
7 import java.io.PrintWriter;
8 import java.util.concurrent.ExecutorService;
9 import java.util.concurrent.Executors;
10 import java.util.concurrent.locks.Condition;
11 import java.util.concurrent.locks.Lock;
12 import java.util.concurrent.locks.ReentrantLock;
13
14 public class AddDelete {
15
16     public static void main(String args [])throws
  Exception{
17         // Create a PrintWriter for the tasks.
18         PrintWriter outf1;
19         outf1=new PrintWriter(new File("ThreadsCooperating
  .txt"));
20         //Create executor
21         ExecutorService executor = Executors.
  newFixedThreadPool(9);
22         // Create a Buffer
23         Buffer Buf1=new Buffer(outf1);
24         Buffer Buf2=new Buffer(outf1);
25         //Create Processes
26         Add one = new Add(Buf1,Buf2,"PB1",5,-3);
27         Add two = new Add(Buf1,Buf2,"FB2",6,78);
28         Add three = new Add(Buf1,Buf2,"PB1",8,13);
29         Add four = new Add(Buf1,Buf2,"MB3",10,22);
30         Add five = new Add(Buf1,Buf2,"FB4",6,75);
31         Delete a = new Delete(Buf2,Buf1,"FB2", 2, 78);
32         Delete b = new Delete(Buf2,Buf1,"PB1", 2, -3);
33         Delete c = new Delete(Buf2,Buf1,"MB3", 4, 22);
34         Delete d = new Delete(Buf2,Buf1,"PB1", 3, 13);
35
36         //Execute processes
37         executor.execute(one);
38         executor.execute(two);
```

```
39          executor.execute(three);
40          executor.execute(four);
41          executor.execute(five);
42          executor.execute(a);
43          executor.execute(b);
44          executor.execute(c);
45          executor.execute(d);
46
47          //Make sure you flush
48          System.out.flush();
49          outf1.flush();
50          executor.shutdown();
51          Buf1.printdata();
52          Buf2.printdata();
53          System.out.flush();
54          outf1.flush();
55
56
57
58      }
59
60      public static  class Buffer
61      { private Object[] storage;
62          private int fill;
63          PrintWriter out1;
64          private int bsize;
65          private static Lock lockstrclr=new ReentrantLock()
    ;//create a lock for objects in this class
66          private static Condition infostorage=lockstrclr.
    newCondition();//create a condition for the lock
67          public Buffer(PrintWriter x)
68          {//This is the constructor
69              out1=x;
70              fill=0;
71              //set the buffer storage to 30 integers
72              storage=new Object[30];
73              // now initialize all integers to 0
74              for(int i=0;i<=29;i++)storage[i]=0;
75          }//end of constructor
76          public int getfill(){return fill;}
77          public void printdata()
78          {//list the storage array
79              for(int i=0;i<=29;i++)System.out.println("x["+
    i+"]="+storage[i]);
80              System.out.flush();
```

```
 81         }
 82         public void storedata (int amt, Object value)
 83         {//first acquire the lock
 84             lockstrclr.lock();
 85             //now try to store the data
 86             try{
 87                 System.out.println("in storedata amt "+
    amt+" value "+value+" fill "+fill);
 88                 while((fill+amt-1)>9)infostorage.await();
    //wait till there is room for the data
 89                 System.out.println("now there is room amt
     "+amt+" value "+value+" fill "+fill);
 90
 91                 System.out.println("in storedata amt "+
    amt+" value "+value+" fill "+ fill);
 92                 out1.println("in storedata amt "+amt+"
    value "+value+" fill "+fill);
 93                 System.out.flush();
 94                 out1.flush();
 95                 // now store the data
 96                 for(int i=fill;i<=fill+amt-1;i++)storage[
    i]=value;
 97                 fill=fill+amt;//fill always points to the
     next available spot
 98                 System.out.println("now the array looks
    like");
 99                 for(int i=0; i<=fill-1;i++)System.out.
    println("x["+i+"]="+storage[i]);
100                 System.out.println("new fill is"+fill);
101                 System.out.flush();
102                 //now that the buffer has a value tell
    conditional to all
103             }//end of  try
104             catch (InterruptedException ex)
105             {System.out.println("trouble in catch in
    storedata");
106                 ex.printStackTrace();
107             }//end of catch
108             finally
109             {//release the lock signal all and release
    the lock
110                 infostorage.signalAll();
111                 lockstrclr.unlock();
112             }//end of finally
113
```

```
114         }//end of storedata
115         public void cleardata(int amt, Object br)
116         {
117             // first acquire the lock
118             lockstrclr.lock();
119             try
120             {
121                 System.out.println("in cleardata amt "+
    amt+" fill "+fill);
122                 System.out.flush();
123                 // trying to clear data.  There must be
    at least one to clear
124                 while(fill<1)infostorage.await();//must
    be at least one value to clear
125 
126 
127                 // now clear this amt of data or less.
128                 if(fill-amt-1<0)
129                 {// the buffer is completely cleared.
130                     int ifmt=fill-1;
131                     System.out.println("M1 Clearing from
     "+ifmt+" to 0");
132                     System.out.flush();
133                     for(int i=fill-1;i>=0;i--)
134                     {//clearing data from the fill-1 down
135                         System.out.println("c"+i);
136                         System.out.flush();
137                         System.out.println("clearing x["+
    "] "+storage[i]);
138                         storage[i]=0;
139                         System.out.flush();
140                     }//end of for
141                     fill=0;
142 
143                 }//end of true
144                 else
145                 {//we can clear the amt and still have
    data
146                     int ifmt=fill-amt;
147                     int ifmt2=fill-1;
148                     System.out.println("M2 Clearing from
     "+ifmt2+" to"+ifmt);
149                     System.out.flush();
150 
151                     for(int i=fill-1;i>=fill-amt;i++)
```

```
152                     {//clearing data from the fill-1 down
153                         if(storage[i]== br) {
154                             System.out.println(i);
155                             System.out.flush();
156                             System.out.println("clearing
    x[" + "] " + storage[i]);
157                             storage[i] = 0;
158                             System.out.flush();
159                             i--;
160                         }//end of if
161                         i--;
162                     }// end of for
163                     fill=fill-amt;
164                     System.out.println("Now the Fill is"+
    fill);
165                     System.out.flush();
166 
167                 }//end of false
168                 System.out.println("leaving cleardata
    fill is "+fill);
169                 System.out.flush();
170             }//end of try
171             catch (InterruptedException ex)
172             {System.out.println("trouble in catch in
    cleardata");
173                 ex.printStackTrace();
174             }//end of catch
175             finally
176             {  //now that the buffer is cleared tell
    conditional to all
177                 infostorage.signalAll();
178 
179 
180 
181                 //release the lock
182                 lockstrclr.unlock();
183             }//end of finally
184         }//end of cleardata
185     }//end of Buffer Class
186 
187 
188     static class Add implements Runnable{//Start of Add
    class
189         private int amount;//Amount on storage entries
190         private int data;//data for storage
```

```
191         private String branch;//Location of data
192         Buffer Bufx;//data buffer
193         Buffer Bufy;//branch buffer
194         public Add(Buffer x, Buffer y, String a, int b,
    int c){//Add constructor
195             branch = a;
196             amount = b;
197             data = c;
198             Bufx=x;
199             Bufy=y;
200         }//end of add constructor
201         public void run() {//Start of task
202             Bufx.storedata(amount, data);//add to Data
    Store
203             Bufy.storedata(amount, branch);//add to
    Credential Store
204 
205         }//end of task
206     }//end of add class
207     static class Delete implements Runnable{
208         Buffer Bufx;//branch buffer
209         Buffer Bufy;//value buffer
210         private String branch;
211         private int amount;
212         private int value;
213         public Delete(Buffer x, Buffer y, String a, int b
    , int c){//Start of Delete constructor
214             Bufx=x;
215             branch = a;
216             amount= b;
217             value = c;
218             Bufy = y;
219         }//end of delete constructor
220         public void run() {//Start of delete task
221             Bufx.cleardata(amount,branch);
222             Bufy.cleardata(amount, value);
223         }//en of delete task
224     }//end of delete class
225 }//End of AddDelete
226 
```