```
 1: // $Id: waitnotifybuffer.java,v 1.16 2013-08-12 19:35:19-07 - - $
 2:
 3: //
 4: // Producer-consumer example using wait and notify on a
 5: // bounded buffer.  Producer blocks when queue is full, and
 6: // consumer blocks when queue is empty.  All classes are
 7: // static inner classes to make the example fit in one file.
 8: // Usually, each class would be in a separate file.
 9: //
10:
11: import java.io.*;
12: import java.util.*;
13: import static java.lang.String.*;
14: import static java.lang.System.*;
15:
16: class waitnotifybuffer {
17:
18: static class arrayqueue<item_t> {
19:    private static final int EMPTY = -1;
20:    private int front = EMPTY;
21:    private int rear = EMPTY;
22:    private item_t[] items;
23:
24:    @SuppressWarnings ("unchecked")
25:    public arrayqueue (int size) {
26:       items = (item_t[]) new Object[size];
27:    }
28:
29:    public boolean is_empty() {
30:       return front == EMPTY;
31:    }
32:
33:    public boolean is_full() {
34:       return (rear + 1) % items.length == front;
35:    }
36:
37:    public void push_rear (item_t item) {
38:       if (is_full()) throw new
39:             IllegalStateException ("arrayqueue.push_rear");
40:       if (is_empty()) front = rear = 0;
41:                 else rear = (rear + 1) % items.length;
42:       items[rear] = item;
43:    }
44:
45:    item_t pop_front() {
46:       if (is_empty()) throw new
47:             IllegalStateException ("arrayqueue.pop_front");
48:       item_t result = items[front];
49:       if (front == rear) front = rear = EMPTY;
50:                   else front = (front + 1) % items.length;
51:       return result;
52:    }
53: }
54:
```

```
 55:
 56: interface buffer<item_t> {
 57:     void put (item_t item);
 58:     item_t get();
 59:     void set_eof();
 60: }
 61:
 62: static class arraybuffer<item_t> implements buffer<item_t> {
 63:     private arrayqueue<item_t> queue;
 64:     private boolean eof = false;
 65:
 66:     public arraybuffer (int size) {
 67:         queue = new arrayqueue<item_t> (size);
 68:     }
 69:
 70:     synchronized public void set_eof() {
 71:         eof = true;
 72:     }
 73:
 74:     synchronized public void put (item_t item) {
 75:         if (eof) throw new IllegalStateException ("put");
 76:         while (queue.is_full()) {
 77:             try {
 78:                 wait();
 79:             }catch (InterruptedException exn) {
 80:                 display ("arraybuffer.put: %s", exn.getMessage());
 81:             }
 82:         }
 83:         queue.push_rear (item);
 84:         notifyAll();
 85:     }
 86:
 87:     synchronized public item_t get() {
 88:         while (queue.is_empty()) {
 89:             if (eof) return null;
 90:             try {
 91:                 wait();
 92:             }catch (InterruptedException exn) {
 93:                 display ("arraybuffer.get: %s", exn.getMessage());
 94:             }
 95:         }
 96:         item_t result = queue.pop_front();
 97:         notifyAll();
 98:         return result;
 99:     }
100: }
101:
```

```
102:
103: static class producer<item_t> implements Runnable {
104:     private buffer<item_t> buf;
105:     private String name;
106:     private long delay_msec;
107:     private item_t[] data;
108:
109:     producer (buffer<item_t> _buf, String _name, long _delay,
110:              item_t[] _data) {
111:        buf = _buf;
112:        name = _name;
113:        delay_msec = _delay;
114:        data = _data;
115:     }
116:
117:     public void run() {
118:        Thread self = Thread.currentThread();
119:        self.setName ("producer " + name);
120:        display ("starting");
121:        int count = 0;
122:        for (item_t datum: data) {
123:           try {
124:               Thread.sleep (delay_msec);
125:           }catch (InterruptedException exn) {
126:               display ("%s", exn.getMessage());
127:           }
128:           display ("put (\"%s\")", datum);
129:           buf.put (datum);
130:        }
131:        display ("finished");
132:     }
133: }
134:
```

```
135:
136: static class consumer<item_t> implements Runnable {
137:     private buffer<item_t> buf;
138:     private String name;
139:     long delay_msec;
140:
141:     consumer (buffer<item_t> _buf, String _name, long _delay) {
142:         buf = _buf;
143:         name = _name;
144:         delay_msec = _delay;
145:     }
146:
147:     public void run() {
148:         Thread self = Thread.currentThread();
149:         self.setName ("consumer " + name);
150:         display ("starting");
151:         for (;;) {
152:             item_t item = buf.get();
153:             if (item == null) break;
154:             try {
155:                 Thread.sleep (delay_msec);
156:             }catch (InterruptedException exn) {
157:                 display ("%s", exn.getMessage());
158:             }
159:             display ("get() = \"%s\"", item);
160:         }
161:         display ("finished");
162:     }
163: }
164:
```

```
165:
166:     static long nanostart = nanoTime();
167:
168:     synchronized static void display (String fmt, Object... args) {
169:         Thread self = Thread.currentThread();
170:         double millisec = (nanoTime() - nanostart) / 1e6;
171:         out.printf ("%10.3f: %s(%d): ",
172:                     millisec, self.getName(), self.getId());
173:         out.printf (fmt + "%n", args);
174:         out.flush();
175:     }
176:
177:     public static void main (String[] args) {
178:         display ("starting");
179:         String[] latin = {
180:             "prîmus", "secundus,", "tertius", "quârtus", "quîntus",
181:             "sextus", "septimus", "octâvus", "nônus", "decimus"
182:         };
183:         String[] greek = { // Transliterated, of course.
184:             "prôtos", "deuteros", "tritos", "tetartos", "pentos",
185:             "hektos", "hebdomos", "ogdoos", "enatos", "dekatos"
186:         };
187:         buffer<String> buf = new arraybuffer<String> (5);
188:         Thread[] producers = {
189:             new Thread (new producer<String> (buf, "Romans", 100, latin)),
190:             new Thread (new producer<String> (buf, "Greeks", 300, greek))
191:         };
192:         Thread[] consumers = {
193:             new Thread (new consumer<String> (buf, "Gauls", 200)),
194:             new Thread (new consumer<String> (buf, "Picts", 500))
195:         };
196:         for (Thread thread: producers) thread.start();
197:         for (Thread thread: consumers) thread.start();
198:         for (int itor = 0; itor < producers.length; ++itor) {
199:             try {
200:                 producers[itor].join();
201:             }catch (InterruptedException exn) {
202:                 display ("join: %s", exn.getMessage());
203:             }
204:         }
205:         buf.set_eof();
206:         display ("finished");
207:     }
208: }
209:
210: //TEST// ./waitnotifybuffer >waitnotifybuffer.lis 2>&1
211: //TEST// mkpspdf waitnotifybuffer.ps \
212: //TEST//         waitnotifybuffer.java* waitnotifybuffer.lis
213:
```

```
 1: * @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting waitnotifybuffer.java
 2: * waitnotifybuffer.java: $Id: waitnotifybuffer.java,v 1.16 2013-08-12
 3: 19:35:19-07 - - $
 4: * javac -Xlint waitnotifybuffer.java
 5: * jar cfm waitnotifybuffer Manifest waitnotifybuffer$arraybuffer.class
 6: waitnotifybuffer$arrayqueue.class waitnotifybuffer$buffer.class
 7: waitnotifybuffer$consumer.class waitnotifybuffer$producer.class
 8: waitnotifybuffer.class
 9: * chmod +x waitnotifybuffer
10: * rm -f waitnotifybuffer$arraybuffer.class
11: waitnotifybuffer$arrayqueue.class waitnotifybuffer$buffer.class
12: waitnotifybuffer$consumer.class waitnotifybuffer$producer.class
13: waitnotifybuffer.class
14: * @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished waitnotifybuffer.java
```

```
 1:       0.046: main(1): starting
 2:       8.413: producer Romans(8): starting
 3:       9.293: consumer Gauls(10): starting
 4:      10.050: producer Greeks(9): starting
 5:      10.611: consumer Picts(11): starting
 6:     109.373: producer Romans(8): put ("prîmus")
 7:     210.164: producer Romans(8): put ("secundus,")
 8:     310.675: producer Greeks(9): put ("prôtos")
 9:     311.385: producer Romans(8): put ("tertius")
10:     410.904: consumer Gauls(10): get() = "secundus,"
11:     412.123: producer Romans(8): put ("quârtus")
12:     512.856: producer Romans(8): put ("quîntus")
13:     610.226: consumer Picts(11): get() = "prîmus"
14:     611.450: producer Greeks(9): put ("deuteros")
15:     612.128: consumer Gauls(10): get() = "prôtos"
16:     613.543: producer Romans(8): put ("sextus")
17:     714.288: producer Romans(8): put ("septimus")
18:     812.809: consumer Gauls(10): get() = "quârtus"
19:     814.978: producer Romans(8): put ("octâvus")
20:     912.198: producer Greeks(9): put ("tritos")
21:     915.736: producer Romans(8): put ("nônus")
22:    1013.545: consumer Gauls(10): get() = "quîntus"
23:    1110.989: consumer Picts(11): get() = "tertius"
24:    1114.312: producer Romans(8): put ("decimus")
25:    1114.988: producer Romans(8): finished
26:    1212.946: producer Greeks(9): put ("tetartos")
27:    1214.300: consumer Gauls(10): get() = "deuteros"
28:    1414.989: consumer Gauls(10): get() = "septimus"
29:    1515.003: producer Greeks(9): put ("pentos")
30:    1611.736: consumer Picts(11): get() = "sextus"
31:    1615.679: consumer Gauls(10): get() = "octâvus"
32:    1815.689: producer Greeks(9): put ("hektos")
33:    1816.413: consumer Gauls(10): get() = "nônus"
34:    2017.098: consumer Gauls(10): get() = "decimus"
35:    2112.482: consumer Picts(11): get() = "tritos"
36:    2116.366: producer Greeks(9): put ("hebdomos")
37:    2217.796: consumer Gauls(10): get() = "tetartos"
38:    2417.121: producer Greeks(9): put ("ogdoos")
39:    2418.473: consumer Gauls(10): get() = "hektos"
40:    2613.251: consumer Picts(11): get() = "pentos"
41:    2619.160: consumer Gauls(10): get() = "hebdomos"
42:    2717.802: producer Greeks(9): put ("enatos")
43:    2918.490: consumer Gauls(10): get() = "enatos"
44:    3018.480: producer Greeks(9): put ("dekatos")
45:    3019.090: producer Greeks(9): finished
46:    3019.635: main(1): finished
47:    3114.020: consumer Picts(11): get() = "ogdoos"
48:    3114.699: consumer Picts(11): finished
49:    3219.178: consumer Gauls(10): get() = "dekatos"
50:    3219.902: consumer Gauls(10): finished
```