[1] **Monitory: Producent-konsument w Javie (I)**

```
public class ProducerConsumer {
  static final int N = 100; static producer p = new producer();
  static consumer c = new consumer();
  static our_monitor mon = new our_monitor();
  public static void main(String args[]) {
    p.start();
    c.start(); }

  static class producer extends Thread {
    public void run() {  // contains thread code
      int item;
      while (true) {
          item = produce_item();
          mon.insert(item); } }
    private int produce_item(){ } }
  static class consumer extends Thread {
    public void run(){
      int item;
      while(true){
          item = mon.remove();
          consume_item(); } }
    private void consume_item(){ } }
```

[2] **Monitory: Producent-konsument w Javie (II)**

```
static class our_monitor{
  private int buffer[]] = new int [N];
  private int count = 0, lo = 0, hi = 0;

  public synchronized void insert(int val){
        if (count == N) go_to_sleep();
        buffer(hi) = val;
        hi = (hi +1) % N;                 // ring buffer
        count = count + 1;                // one more item in buffer
        if (count == 1) notify();  }      // notify() in in Java
  public synchonized int remove(){
        int val;
        if (count == 0) go_to_sleep();  // buffer empty
        val = buffer[lo];
        lo = (lo+1) % N;
        count = count - 1;
        if ( count == N-1) notify();
        return val; }
  private go_to_sleep(){
        try {wait();}
        catch(InterruptedException exp){}; }
}
```