Chapter 1

Process Management

1.1. Process Scheduling

1.2. Process Synchronization

1.2.1. Producer-Consumer Problem

```
// Shared bounded buffer
class Buffer {
    int[] items;
    int size;
    int count;
    int in;
    int out;

    // Constructor
    void Buffer(int capacity) {
        size = capacity;
        items = new int[size];
        count = 0;
        in = 0;
        out = 0;
    }

    // Add an item to the buffer (producer)
    void put(int value) {
        while (count == size) {
            wait();    // buffer full
        }

        items[in] = value;
        in = (in + 1) % size;
        count = count + 1;

        notify();       // wake up consumer
    }

    // Remove an item from the buffer (consumer)
    int get() {
        while (count == 0) {
            wait();     // buffer empty
        }

        int value = items[out];
        out = (out + 1) % size;
        count = count - 1;

        notify();       // wake up producer
        return value;
    }
}

// Producer thread
class Producer {
    Buffer buffer;

    void Producer(Buffer b) {
        buffer = b;
    }

    void run() {
        int x = 0;

        while (true) {
            buffer.put(x);
            println("Produced: " + x);
            x = x + 1;
        }
    }
}

// Consumer thread
class Consumer {
    Buffer buffer;

```

```
    void Consumer(Buffer b) {
        buffer = b;
    }

    void run() {
        while (true) {
            int v = buffer.get();
            println("Consumed: " + v);
        }
    }
}

// Program entry point
void main() {
    Buffer buf = new Buffer(5);

    Producer p = new Producer(buf);
    Consumer c = new Consumer(buf);

    startThread(p);
    startThread(c);
}
```