# Producer-consumer and BlockingCollection Demo

Simon Robinson

@TechieSimon | TechieSimon.com

## Module 5 Overview

- *Demo* `ConcurrentQueue<T>` *in producer-consumer situation*
- `ConcurrentQueue<T>` not adequate *because of issue of polling for tasks*
- *`BlockingCollection<T>`* provides additional *services* (and *solves* the polling issue)

CODE DEMO

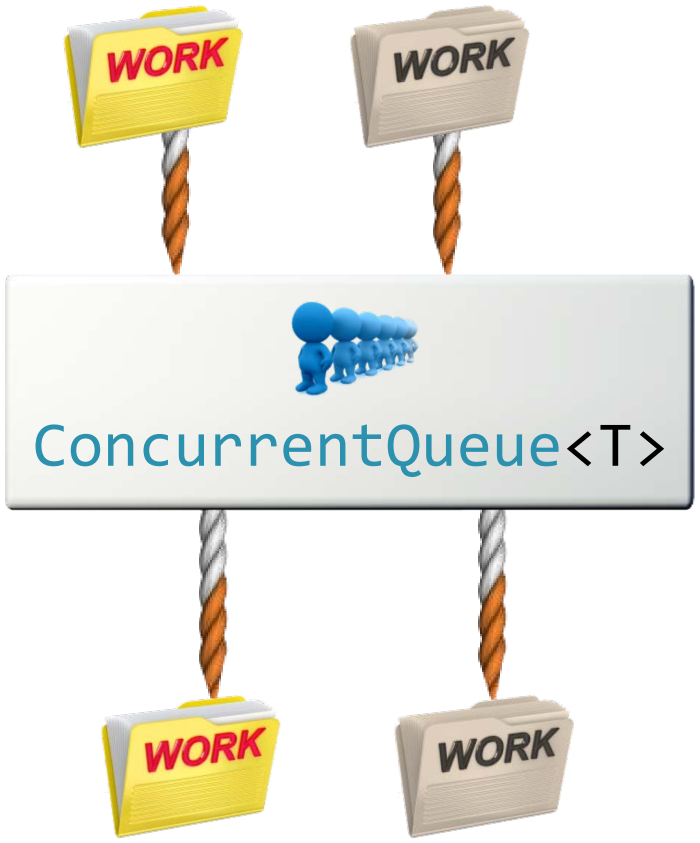
This slide must not appear in the recorded course

CODE DEMO

Grey area must not appear in the recorded course

## Producer-Consumer Demo

Some threads add tasks

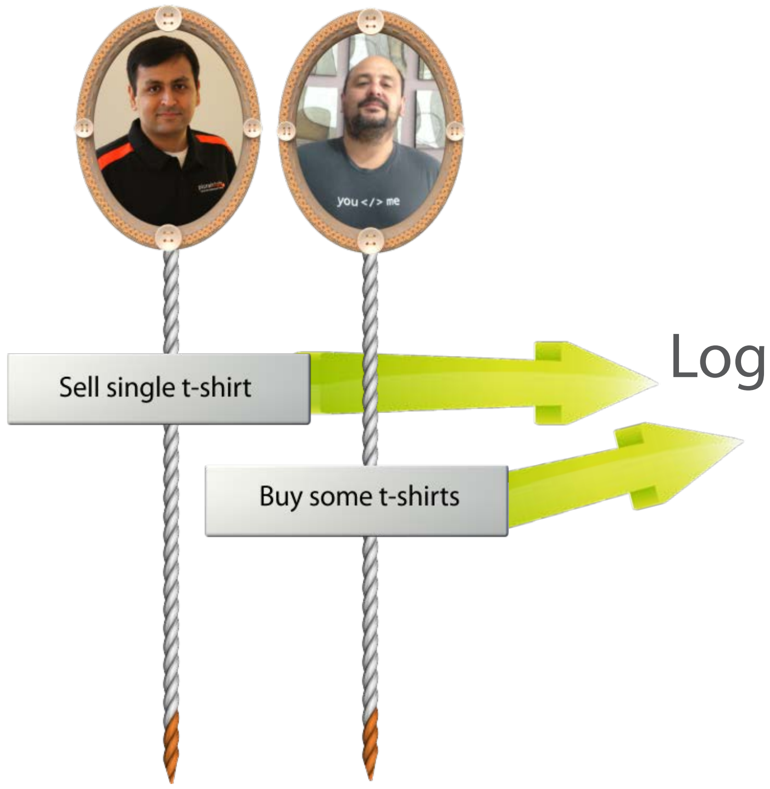


Other threads remove tasks

CODE DEMO

This slide must not appear in the recorded course

# New in SalesBonuses App…



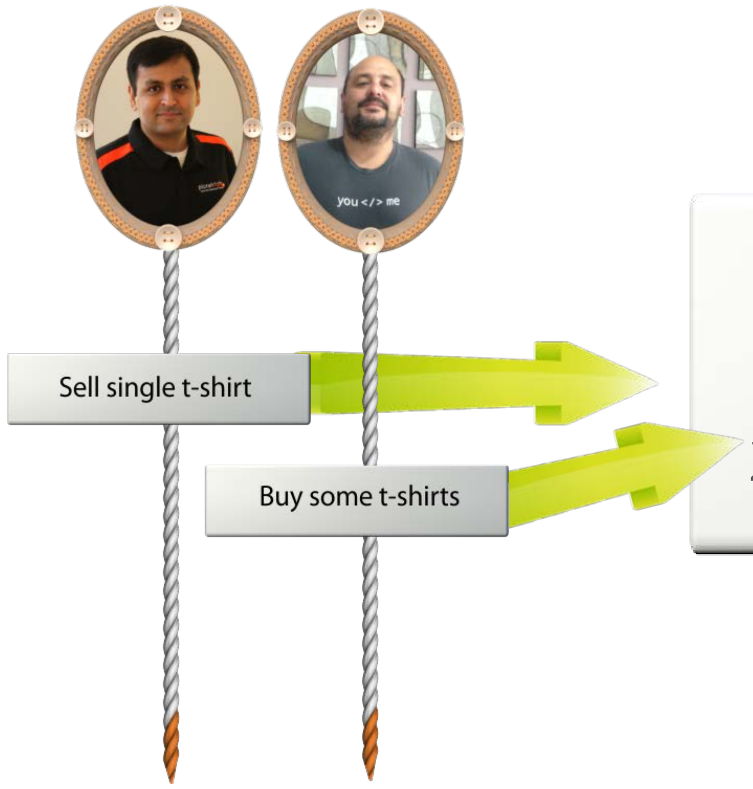
Log trade for calculating staff bonus

Calculating bonus is…

…not urgent

…time-consuming

# When a Trade Is Made…



Immediately…

1. Update stock levels
2. Log the trade

Use a `ConcurrentQueue<T>` for this!

CODE DEMO

This slide must not appear in the recorded course

Famous TechieSimon Perfect

Geek Clothing Company

## Short Intermission

CODE DEMO

This slide must not appear in the recorded course

# What Can BlockingCollection<T> Wrap?

`IProducerConsumerCollection<T>`

✓

- `ConcurrentQueue<T>`
- `ConcurrentStack<T>`
- `ConcurrentBag<T>`

✗

- `ConcurrentDictionary<TKey, TValue>`

CODE DEMO

This slide must not appear in the recorded course

# BlockingCollection<T> Extra Features

`Take()`
is cancellable

`TryTake()`
allows a timeout

Maximum
collection size

`Add()`
is cancellable

`TryAdd()`
allows a timeout

## Module 5 Summary

- *Demo'd* `ConcurrentQueue<T>` to *store* tasks awaiting processing
- `ConcurrentQueue<T>` lacks ability *to wait* until an item is on the queue
  - `ConcurrentStack<T>` and `ConcurrentBag<T>` have same problem
- *`BlockingCollection<T>`* fixes this