Concurrency – one more step ahead

What is concurrent object? They key thing that several concurrent object may perform their operations concurrently. But can each of such objects perform several its own operations concurrently? Let’s see.

**concurrent unit** Consumer[Data]

**pure** consume (data: Data) **do**

**if** canConsume **do**

/\* Process the data \*/

**else**

**concurrent raise** FailedToConsumeError

**end**

**end**

**end**

**concurrent unit** Producer[Data]

**pure** produce: Data => **new** Data

**end**

**unit** Connect[Data]

consumer: Consumer[Data]

producer: Producer[Data]

queue: Queue[Data]

Connect (**as** consumer; **as** producer)

**pure** process **do**

**while** true **do**

consumer.consumer (producer.produce)

**when** FailedToConsumeError **do**

// Handle the case when consumer failed to process the data

**end**

**end**

**end**

consumer **is** **new** Consumer[SomeUnitType] // single consumer

**while** producer **in**

(**new** Producer[SomeUnitType], **new** Producer[SomeUnitType],**new** Producer[SomeUnitType])

**do**

**new** Connect[SomeUnitType] (consumer, producer).process

**end**

Concurrent execution rules:

1. Only one “dirty” routine can run on the object at a time
   1. If the “routine” has pure or safe blocks in its body then during execution of these blocks unlimited number of other pure or safe routines can be executed
2. Unlimited number of pure or safe routines can be run on the object
3. Routine which is may generate exception in the caller using concurrent raise

**concurrent unit** Queue[G]

getElement: G

**require** canGet

**do**

**return** := elements(1)

elements.delete (1)

**end**

**pure** canGet: Boolean => elements.count > 0

{}elements: Array [G]

putElement(e: G) **do**

elements.append (e)

**end**

**end**

**concurrent unit** Queue<G>

G getElement

**require** canGet

{

**return** := elements(1)

elements.delete (1)

}

**pure** Boolean canGet => elements.count > 0

[] Array <G> elements

putElement(G e) {

elements.append (e)

}

}

-----------------

Use stack to parse!

A < B >