This module simulates a producer-consumer example as it could be written using Java threads. In particular, we want to demonstrate the risk of deadlock when producers and consumers wait on the same object.

**EXTENDS** *Naturals, Sequences*

**CONSTANTS**
- *Producers*, the (nonempty) set of producers
- *Consumers*, the (nonempty) set of consumers
- *BufCapacity*, the maximum number of messages in the bounded buffer
- *Data*, the set of values that can be produced and/or consumed

**ASSUME**
- $\forall Producers \neq \{\}$, at least one producer
- $\forall Consumers \neq \{\}$, at least one consumer
- $\forall Producers \cap Consumers = \{\}$, no thread is both consumer and producer
- $\forall BufCapacity > 0$, buffer capacity is at least 1
- $\forall Data \neq \{\}$, the type of data is nonempty

**VARIABLES**
- *buffer*, the buffer, as a sequence of objects
- *waitSet*, the wait set, as a set of threads

**Participants** $\triangleq Producers \cup Consumers$

**RunningThreads** $\triangleq Participants \setminus waitSet$

**TypeInv** $\triangleq \forall buffer \in \text{Seq}(Data)$
- $\land \text{Len}(buffer) \in 0 \ldots \text{BufCapacity}$
- $\land waitSet \subseteq* Participants$

**Notify** $\triangleq $ IF $waitSet \neq \{\}$
- THEN $\exists x \in waitSet : waitSet' = waitSet \setminus \{x\}$
- ELSE UNCHANGED $waitSet$

**NotifyAll** $\triangleq waitSet' = \{\}$

**Wait** $(t)$ $\triangleq$ $waitSet' = waitSet \cup \{t\}$

**Init** $\triangleq buffer = \langle \rangle \land waitSet = \{\}$

**Put** $(t, m)$ $\triangleq$ IF $\text{Len}(buffer) < \text{BufCapacity}$
- THEN $\land buffer' = \text{Append}(buffer, m)$
- $\land Notify$
- ELSE $\land Wait(t)$
- $\land$ UNCHANGED $buffer$

**Get** $(t)$ $\triangleq$ IF $\text{Len}(buffer) > 0$
- THEN $\land buffer' = \text{Tail}(buffer)$
- $\land Notify$
- ELSE $\land Wait(t)$
- $\land$ UNCHANGED $buffer$

**Next** $\triangleq \exists t \in RunningThreads : \lor t \in Producers \land \exists m \in Data : Put(t, m)$
- $\lor t \in Consumers \land Get(t)$

**Prog** $\triangleq Init \land [\text{Next}]_{buffer, waitSet}$

**NoDeadlock** $\triangleq \Box (RunningThreads \neq \{\})$

**THEOREM** $\text{Prog} \Rightarrow \Box \text{TypeInv} \land \text{NoDeadlock}$