```
MODULE IdGeneratorByteCode

EXTENDS Integers, Sequences, TLC

CONSTANT NumberOfProcesses, Locking

--fair algorithm IdGenerator{

  variable
  this = [lastId ↦ 42, lock ↦ 0],
  stacks = [i ∈ 1 .. NumberOfProcesses ↦ ⟨⟩],
  returnValue = [i ∈ 1 .. NumberOfProcesses ↦ −1]
  ;

  define {
    Last(S) ≜ S[Len(S)]
    Pop(S) ≜ SubSeq(S, 1, Len(S) − 1)
    AllIsDone ≜ (∀ i ∈ 1 .. NumberOfProcesses : pc[i] = "Done")
    AllStacksAreEmpty ≜ (∀ i ∈ 1 .. NumberOfProcesses : stacks[i] = ⟨⟩)
    IdsAreAllDifferent ≜ (∀ i, j ∈ 1 .. NumberOfProcesses : i ≠ j ⇒ returnValue[i] ≠ returnValue[j])
    IdGeneratorInvariant ≜ AllIsDone ⇒ AllStacksAreEmpty ∧ IdsAreAllDifferent
   }

   update and read are now one atomic step
  process ( id ∈ 1 .. NumberOfProcesses ) {
     if Locking constant is TRUE, then we can't start the process of executing ++ lastId unless the lock is unlocked
    checkLocking: if ( Locking ) {
                          waitForLock:
                              await this.lock = 0;
                              this.lock := self;
                        } ;
     Load _this_ onto the operand stacks
    aload0: stacks[self] := Append(stacks[self], this);
     copy the top of the stacks
    dup: stacks[self] := Append(stacks[self], Last(stacks[self]));
     retrieve the value of field lastId from _this_ and store it back on the top of the stacks
    getfield_lastId:
          with ( lastId = Last(stacks[self]).lastId ) {
            stacks[self] := Append(Pop(stacks[self]), lastId);
           } ;
     push the integer constant 1 on the stacks
    iconst_1: stacks[self] := Append(stacks[self], 1);
      integer add the top two values on the top of the stacks
    iadd:
        with ( a = Last(stacks[self]), b = Last(Pop(stacks[self])) ) {
            stacks[self] := Append(Pop(Pop(stacks[self])), a + b);
         } ;
    duplicate the value on top of the stack and put it before _this_
    $dup\_x1$:
        $stacks[self] := \langle Last(stacks[self]) \rangle \circ stacks[self]$ ;
    Store the top value on the operand stack into the field value of the
    current object, represented by the net-to-top value on the operand stack, _this_
    $putfield$:
        $this.lastId := Last(stacks[self])$ ;
        $stacks[self] := Pop(Pop(stacks[self]))$ ;
    return the top(and only) value on the stack
    $ireturn$:
        $returnValue[self] := Last(stacks[self])$ ;
        $stacks[self] := Pop(stacks[self])$ ;
     if ( Locking ) {
        $unlock$: $this.lock := 0$ ;
      } ;
   }
 }
```

* Modification History
* *Last* modified Sun *Dec* 13 15:22:05 *EST* 2015 by *balopat*
* Created *Thu Dec* 10 19:27:24 *EST* 2015 by *balopat*