```
MODULE Commons
EXTENDS Integers

CONSTANTS CLIENT_NUM,   the number of clients
          MAX_KILL      maximum allowed kill events

VARIABLES exec_state,   the execution state of the program: running, success, or fatal
          clients,      clients sending value update requests to master and backup
          master,       array of master instances, only one is active
          backup,       array of backup instances, only one is active
          msgs,         in-flight messages
          killed        number of invoked kill actions to master or backup

 Identifiers related to master and backup instance ids
FIRST_ID ≜ 1
MAX_INSTANCE_ID ≜ MAX_KILL + 1
INSTANCE_ID ≜ FIRST_ID .. MAX_INSTANCE_ID
UNKNOWN_ID ≜ 0
NOT_INSTANCE_ID ≜ −1

 Identifiers related to master and backup instance statuses
INST_STATUS_NULL ≜ "null"      null, not used yet
INST_STATUS_ACTIVE ≜ "active"  active and handling client requests
INST_STATUS_LOST ≜ "lost"      lost
NOT_STATUS ≜ "invalid"         invalid status
INSTANCE_STATUS ≜ {INST_STATUS_NULL,
                   INST_STATUS_ACTIVE,
                   INST_STATUS_LOST}

 Master instance record structure
Master ≜ [id : INSTANCE_ID, backupId : INSTANCE_ID ∪ {UNKNOWN_ID},
          status : INSTANCE_STATUS, value : Nat, version : Nat]

 Invalid master instance
NOT_MASTER ≜ [id ↦ NOT_INSTANCE_ID, backupId ↦ NOT_INSTANCE_ID,
              status ↦ NOT_STATUS, value ↦ −1, version ↦ −1]

 Backup instance record structure
Backup ≜ [id : INSTANCE_ID, masterId : INSTANCE_ID ∪ {UNKNOWN_ID},
          status : INSTANCE_STATUS, value : Nat, version : Nat]

 Invalid backup instance
NOT_BACKUP ≜ [id ↦ NOT_INSTANCE_ID, masterId ↦ NOT_INSTANCE_ID,
              status ↦ NOT_STATUS, value ↦ −1, version ↦ −1]

LastLostMaster ≜
  Return the lost master, or NOT_MASTER if master is alive
```

```
51    LET mset ≜ {m ∈ INSTANCE_ID : master[m].status = INST_STATUS_LOST}
52    IN  IF mset = {} THEN NOT_MASTER
53        ELSE master[(CHOOSE n ∈ mset : ∀ m ∈ mset : n ≥ m)]

55  FindMaster(mStatus) ≜
      Return the master with given status or NOT_MASTER otherwise
59    LET mset ≜ {m ∈ INSTANCE_ID : master[m].status = mStatus}
60    IN  IF mset = {} THEN NOT_MASTER
61        ELSE master[(CHOOSE x ∈ mset : TRUE)]

63  LastKnownMaster ≜
      Return the last known master, whether active or lost
67    LET mset ≜ {m ∈ INSTANCE_ID : master[m].status ≠ INST_STATUS_NULL}
68    IN  master[(CHOOSE n ∈ mset : ∀ m ∈ mset : n ≥ m)]

70  FindBackup(bStatus) ≜
      Return the backup with given status or NOT_BACKUP otherwise
74    LET bset ≜ {b ∈ INSTANCE_ID : backup[b].status = bStatus}
75    IN  IF bset = {} THEN NOT_BACKUP
76        ELSE backup[(CHOOSE x ∈ bset : TRUE)]

78  LastLostBackup ≜
      Return the lost backup, or NOT_BACKUP if backup is alive
82    LET bset ≜ {b ∈ INSTANCE_ID : backup[b].status = INST_STATUS_LOST}
83    IN  IF bset = {} THEN NOT_BACKUP
84        ELSE backup[(CHOOSE n ∈ bset : ∀ m ∈ bset : n ≥ m)]

86  LastKnownBackup ≜
      Return the last known backup, whether active or lost
90    LET bset ≜ {b ∈ INSTANCE_ID : backup[b].status ≠ INST_STATUS_NULL}
91    IN  backup[(CHOOSE n ∈ bset : ∀ m ∈ bset : n ≥ m)]

93  ----------------------------------------------------------------------

94    Identifiers related to client ids and phases
95  CLIENT_ID ≜ 1 .. CLIENT_NUM
96  NOT_CLIENT_ID ≜ −1

98    client phases
99  CLIENT_PHASE ≜ 1 .. 4
100 PH1_PENDING ≜ 1
101 PH2_WORKING ≜ 2
102 PH2_COMPLETED ≜ 3
103 PH2_COMPLETED_FATAL ≜ 4
104 NOT_CLIENT_PHASE ≜ −1

106   Client record structure
```

```
Client ≜ [id : CLIENT_ID, phase : CLIENT_PHASE, value : Nat,
          masterId : INSTANCE_ID,  the master instance last communicated with
          backupId : INSTANCE_ID ∪ {UNKNOWN_ID}  the backup instance last communicated with, initially unl
         ]

 Invalid client instance
NOT_CLIENT ≜ [id ↦ NOT_CLIENT_ID, phase ↦ NOT_CLIENT_PHASE, value ↦ 0]

FindClient(phase) ≜
  Return a client matching the given phase, or NOT_CLIENT otherwise
  LET cset ≜ {c ∈ CLIENT_ID : clients[c].phase = phase}
  IN  IF cset = {} THEN NOT_CLIENT
      ELSE clients[(CHOOSE x ∈ cset : TRUE)]

──────────────────────────────────────────────────────────────────────

 Message record structure
Messages ≜ [from : {"c", "m", "b", "sys"}, to : {"c", "m", "b"},
            clientId  : CLIENT_ID,
            masterId : INSTANCE_ID ∪ {UNKNOWN_ID},
            backupId : INSTANCE_ID ∪ {UNKNOWN_ID},
            value : Nat,
            tag : {"masterDo", "masterDone",
                   "backupDo", "backupDone",
                   "masterGetNewBackup", "newBackupId",
                   "backupGetNewMaster", "newMasterId"
                  }]

 Invalid message instance
NOT_MESSAGE ≜ [from ↦ "na", to ↦ "na"]

SendMsg(m) ≜
  Add message to the msgs set
  msgs' = msgs ∪ {m}

RecvMsg(m) ≜
  Delete message from the msgs set
  msgs' = msgs \ {m}

ReplaceMsg(toRemove, toAdd) ≜
  Remove an existing message and add another one
  msgs' = (msgs \ {toRemove}) ∪ {toAdd}

FindMessageToWithTag(to, status, optionalTag) ≜
  Return a message matching the given criteria, or NOT_MESSAGE otherwise
  LET mset ≜ {m ∈ msgs : ∧ m.to = to
                         ∧ IF to = "m"
```

```
163                                   THEN master[m.masterId].status = status
164                                   ELSE IF to = "b"
165                                   THEN backup[m.backupId].status = status
166                                   ELSE FALSE
167                                 ∧ IF optionalTag = "NA"
168                                   THEN TRUE
169                                   ELSE m.tag = optionalTag}
170    IN   IF mset = {} THEN NOT_MESSAGE
171         ELSE (CHOOSE x ∈ mset : TRUE)

173 FindMessageTo(to, status) ≜ FindMessageToWithTag(to, status, "NA")

175 FindMessageToClient(from, tag) ≜
      Return a message sent to client matching given criteria, or NOT_MESSAGE otherwise
180    LET mset ≜ {m ∈ msgs : ∧ m.from = from
181                           ∧ m.to = "c"
182                           ∧ m.tag = tag}
183    IN   IF mset = {} THEN NOT_MESSAGE
184         ELSE (CHOOSE x ∈ mset : TRUE)

186 ================================================================================
```