```
──────────────── MODULE facebookcacheinvalidation2 ────────────────
  invalidationQueue

EXTENDS Naturals, Sequences


CONSTANTS
    KEYS


VARIABLES
    database,
      Represents metadata version stored in the cache
    cache,
      Represents the version stored in the cache, This is what is used for comparsions.
      to allow our model to decouple ACTUAL metadata version with STORED version
    cacheVersions,
    cacheFillStates,
    invalidationQueue,
    counter  Used to prevent repeated states for liveness conditions

  We can still test with the same cache requirements we've been using this whole time
INSTANCE cacherequirements

vars ≜ ⟨database, cache, cacheFillStates,
    invalidationQueue, counter, cacheVersions⟩

InvalidationMessage ≜ [key : KEYS, version : DataVersion]

CacheFillState ≜ [
    state : {
        "inactive",
        "startfillmetadata",
        "respondedtometadata",  Next: CacheFillMetadata
        "startfillversion",
        "responedtoversion"  Next: CacheFillVersion
    },
    version : DataVersion
]

CacheValue ≜ CacheMiss ∪ CacheHit

TypeOk ≜
    ∧  database ∈ [KEYS → DataVersion]
    ∧  cache ∈ [KEYS → CacheValue]
      Cache versions are typed identically to cache
    ∧ cacheVersions ∈ [KEYS → CacheValue]
    ∧ cacheFillStates ∈ [KEYS → CacheFillState]
    ∧ invalidationQueue ∈ Seq(InvalidationMessage)
    ∧ counter ∈ Nat

Init ≜
    ∧ database = [k ∈ KEYS ↦ 0]
     cache (metadata) and cacheVersions start empty together
    ∧ cache = [k ∈ KEYS ↦ [type ↦ "miss"]]
    ∧ cacheVersions = [k ∈ KEYS ↦ [type ↦ "miss"]]

    ∧ cacheFillStates = [k ∈ KEYS ↦ [
                              state ↦ "inactive",
                              version ↦ 0
                              ]
                          ]
    ∧ invalidationQueue = ⟨⟩
    ∧ counter = 0

DatabaseUpdate(k) ≜
    LET updatedVersion ≜ database[k] + 1IN
    ∧ database′ = [database EXCEPT
                     ![k] = updatedVersion]
    ∧ invalidationQueue′ = Append(invalidationQueue,
                                   [key ↦ k, version ↦ updatedVersion])
    ∧ UNCHANGED ⟨cache, cacheVersions, cacheFillStates, counter⟩

CacheStartFillMetadata(k) ≜
    ∧ cache[k] ∈ CacheMiss
    ∧ cacheFillStates[k].state = "inactive"
    ∧ cacheFillStates′ = [cacheFillStates EXCEPT ![k].state = "startfillmetadata"]
    ∧ UNCHANGED ⟨database, cache, cacheVersions, invalidationQueue, counter⟩

DatabaseRespondWithMetadata(k) ≜
    ∧ cacheFillStates[k].state = "startfillmetadata"
    ∧ cacheFillStates′ = [cacheFillStates EXCEPT
                              ![k].state = "respondedtometadata",
                              ![k].version = database[k]]
    ∧ UNCHANGED ⟨database, cache, cacheVersions, invalidationQueue, counter⟩

CacheFillMetadata(k) ≜
     facebookmetdataversion
    ∧ cacheFillStates[k].state = "respondedtometadata"
    ∧ cache′ = [cache EXCEPT
```

```
                                  ![k] = [
                                      type ↦ "hit",
                                      version ↦ cacheFillStates[k].version
                                  ]
                           ]
       ∧ cacheFillStates′ = [cacheFillStates EXCEPT
                                  ![k].state = "inactive",
                                  ![k].version = 0]
       ∧ UNCHANGED ⟨database, cacheVersions, invalidationQueue, counter⟩

CacheStartFillVersion(k) ≜
       ∧ cacheVersions[k] ∈ CacheMiss
       ∧ cacheFillStates[k].state = "inactive"
       ∧ cacheFillStates′ = [cacheFillStates EXCEPT ![k].state = "startfillversion"]
       ∧ UNCHANGED ⟨database, cache, cacheVersions, invalidationQueue, counter⟩

DatabaseRespondWithVersion(k) ≜
       ∧ cacheFillStates[k].state = "startfillversion"
       ∧ cacheFillStates′ = [cacheFillStates EXCEPT
                                  ![k].state = "responedtoversion",
                                  ![k].version = database[k]]
       ∧ UNCHANGED ⟨database, cache, cacheVersions, invalidationQueue, counter⟩

CacheFillVersion(k) ≜
       facebookversion
       ∧ cacheFillStates[k].state = "responedtoversion"
       ∧ ∨ cacheVersions[k] ∈ CacheMiss
         ∨ ∧ cacheVersions[k] ∉ CacheMiss
           ∧ cacheVersions[k].version < cacheFillStates[k].version
       ∧ cacheVersions′ = [cacheVersions EXCEPT
                                  ![k] = [
                                      type ↦ "hit",
                                      version ↦ cacheFillStates[k].version
                                  ]
                           ]
       ∧ cacheFillStates′ = [cacheFillStates EXCEPT
                                  ![k].state = "inactive",
                                  ![k].version = 0]
       ∧ UNCHANGED ⟨database, cache, invalidationQueue, counter⟩

CacheIgnoreFillVersion(k) ≜
       ∧ cacheFillStates[k].state = "responedtoversion"
       ∧ ∧ cacheVersions[k] ∈ CacheHit
```

```
        $\land\ cacheVersions[k].version \geq cacheFillStates[k].version$
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT
                                $![k].state =$ "inactive",
                                $![k].version = 0]$

    $\land\ counter' = counter + 1$
    $\land$ UNCHANGED $\langle database, cache, cacheVersions, invalidationQueue \rangle$

$CacheFailFill(k) \triangleq$
    $\land\ cacheFillStates[k].state \in$ {"respondedtometadata", "responedtoversion"}
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT
                                $![k].state =$ "inactive",
                                $![k].version = 0]$

    $\land\ counter' = counter + 1$
    $\land$ UNCHANGED $\langle database, cache, cacheVersions, invalidationQueue \rangle$


$CacheEvict(k) \triangleq$
    $\land\ cache[k] \in CacheHit$
    $\land\ cacheFillStates[k].state =$ "inactive"
    $\land\ cache' = [cache$ EXCEPT $![k] = [type \mapsto$ "miss"]]
    $\land\ cacheVersions' = [cache$ EXCEPT  $![k] = [type \mapsto$ "miss"]]

    $\land\ counter' = counter + 1$
    $\land$ UNCHANGED $\langle database, cacheFillStates, invalidationQueue \rangle$

$UpdateFromInvalidationMessage \triangleq$
    $\land\ invalidationQueue \neq \langle\rangle$
    $\land$ LET $message \triangleq Head(invalidationQueue)$IN
           metadata
       $\land$   $\lor$ $\land\ cache[message.key] \in CacheHit$
                $\land\ cacheVersions[message.key] \in CacheMiss$

             $\lor$ $\land\ cacheVersions[message.key] \in CacheHit$
                $\land\ cacheVersions[message.key].version \leq message.version$


       $\land\ cacheFillStates[message.key].state =$ "inactive"

       metadataversion
       $\land\ cache' = [cache$ EXCEPT
                         $![message.key] = [$
                                 $type \mapsto$ "hit",
                              version ↦ message.version
                          ]
                      ]
          ∧ cacheVersions′ = [cacheVersions EXCEPT
                                  ![message.key] = [
                                      type ↦ "hit",
                                      version ↦ message.version
                                  ]
                              ]
          ∧ invalidationQueue′ = Tail(invalidationQueue)

          ∧ UNCHANGED ⟨database, cacheFillStates, counter⟩

IgnoreInvalidationMessage ≜
      ∧ invalidationQueue ≠ ⟨⟩
      ∧ LET message ≜ Head(invalidationQueue)IN
              key
          ∧   ∨ ∧ cache[message.key] ∈ CacheMiss
                  ∧ cacheFillStates[message.key].state = "inactive"

              ∨ ∧ cacheVersions[message.key] ∈ CacheHit
                  ∧ cacheVersions[message.key].version > message.version
          ∧   invalidationQueue′ = Tail(invalidationQueue)

          ∧ counter′ = counter + 1
      ∧ UNCHANGED ⟨database, cache, cacheVersions, cacheFillStates⟩


FailUpdateInvalidationMessageIgnore ≜
    ∧ invalidationQueue ≠ ⟨⟩
    ∧ LET message ≜ Head(invalidationQueue)IN
        \* version  version
        ∧ cacheVersions[message.key] ∈ CacheHit
        ∧ cacheVersions[message.key].version ≥ message.version
        ∧ invalidationQueue′ = Tail(invalidationQueue)
        \*
        ∧ counter′ = counter + 1
    ∧ UNCHANGED ⟨database, cache, cacheVersions, cacheFillStates⟩


FailUpdateInvalidationMessageIgnore ≜
      ∧ invalidationQueue ≠ ⟨⟩
      ∧ LET message ≜ Head(invalidationQueue)IN
            version  version
          ∧ cacheVersions[message.key] ∈ CacheHit
          ∧ cacheVersions[message.key].version > message.version
          ∧ invalidationQueue′ = Tail(invalidationQueue)

          ∧ counter′ = counter + 1
          ∧ UNCHANGED ⟨database, cache, cacheVersions, cacheFillStates⟩

FailUpdateInvalidationMessageEvictkey ≜
      ∧ invalidationQueue ≠ ⟨⟩
      ∧ LET message ≜ Head(invalidationQueue)IN
              metadata
          ∧   ∨ ∧ cache[message.key] ∈ CacheHit
                  ∧ cacheVersions[message.key] ∈ CacheMiss

              ∨ ∧ cacheVersions[message.key] ∈ CacheHit
                 update
                  ∧ cacheVersions[message.key].version < message.version


          ∧ cacheFillStates[message.key].state = "inactive"

          ∧ invalidationQueue′ = Tail(invalidationQueue)
          ∧ cache′ = [cache EXCEPT ![message.key] = [type ↦ "miss"]]
          ∧ cacheVersions′ = [cacheVersions EXCEPT ![message.key] = [type ↦ "miss"]]
      ∧ UNCHANGED ⟨database, cacheFillStates, counter⟩


CacheFairness ≜
      ∨ ∃ k ∈ KEYS :
          ∨ CacheStartFillMetadata(k)
          ∨ DatabaseRespondWithMetadata(k)
          ∨ CacheFillMetadata(k)
          ∨ CacheStartFillVersion(k)
          ∨ DatabaseRespondWithVersion(k)
          ∨ CacheFillVersion(k)
          ∨ CacheIgnoreFillVersion(k)
      ∨ UpdateFromInvalidationMessage
      ∨ IgnoreInvalidationMessage
      ∨ FailUpdateInvalidationMessageIgnore
      ∨ FailUpdateInvalidationMessageEvictkey

Next ≜
      ∨ ∃ k ∈ KEYS :
           Database state
          ∨ DatabaseUpdate(k)
           Cache state
          ∨ CacheStartFillMetadata(k)
```

$$
\begin{array}{l}
\qquad\quad \lor \mathit{DatabaseRespondWithMetadata}(k) \\
\qquad\quad \lor \mathit{CacheFillMetadata}(k) \\
\qquad\quad \lor \mathit{CacheStartFillVersion}(k) \\
\qquad\quad \lor \mathit{DatabaseRespondWithVersion}(k) \\
\qquad\quad \lor \mathit{CacheFillVersion}(k) \\
\qquad\quad \lor \mathit{CacheIgnoreFillVersion}(k) \\
\qquad\quad \lor \mathit{CacheFailFill}(k) \\
\qquad\quad \lor \mathit{CacheEvict}(k) \\
\quad\;\; \lor \mathit{UpdateFromInvalidationMessage} \\
\quad\;\; \lor \mathit{IgnoreInvalidationMessage} \\
\quad\;\; \lor \mathit{FailUpdateInvalidationMessageIgnore} \\
\quad\;\; \lor \mathit{FailUpdateInvalidationMessageEvictkey}
\end{array}
$$

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \land \mathrm{WF}_{\mathit{vars}}(\mathit{CacheFairness})$$

$$\mathit{CounterBound} \triangleq \mathit{counter} \leq 2$$