

# ISW2 - Software Testing

Pierpaolo Spaziani - 0316331

Repository BookKeeper: <https://github.com/pierpaolospaziani/bookkeeper>

Repository OpenJPA: <https://github.com/pierpaolospaziani/openjpa>

## 0. Preambolo

### 1. BookKeeper - ReadCache

void put(long ledgerId, long entryId, ByteBuf entry) .....	1
ByteBuf get(long ledgerId, long entryId) .....	3

### 2. BookKeeper - Journal

List<Long> listJournalIds(File journalDir, JournalIdFilter filter) .....	4
long scanJournal(long journalId, long journalPos, JournalScanner scanner) ...	6

### 3. OpenJPA - BrokerImpl

void handleCallbackExceptions(Exception[] excep, int mode) .....	8
void throwNestedExceptions(List<Exception> excep, boolean datastore)	10

### 4. OpenJPA - ProxyManagerImpl

Proxy newCustomProxy(Object orig, boolean autoOff) .....	11
Object copyArray(Object orig) .....	13

### 5. OpenJPA - Integration Test

LifecycleEventManager & BrokerImpl .....	14
--	----

### 6. Allegati

## 0. Preambolo

In entrambi i progetti è stato utilizzato *GitHub Actions* come framework di CI. In ciascuno è possibile trovare 3 workflow:

- **SonarCloud + Jacoco**
  - [https://sonarcloud.io/project/overview?id=pierpaolospaziani\\_bookkeeper](https://sonarcloud.io/project/overview?id=pierpaolospaziani_bookkeeper)
  - [https://sonarcloud.io/project/overview?id=pierpaolospaziani\\_openjpa](https://sonarcloud.io/project/overview?id=pierpaolospaziani_openjpa)
- **Mutation testing**
- **Badua**

Nei primi due è possibile scaricare gli artifact che contendono i report di *Jacoco* e *Pitest*.

Tuttavia non è stato possibile configurare correttamente il workflow *Badua* poiché, nonostante siano stati inclusi nel progetto i jar necessari, installati tramite pom e tramite il workflow, la macchina tenta di reperire *ba-dua-cli* da maven central che ne è sprovvisto.

Inoltre Badua necessita di Java 8, per farlo funzionare su OpenJPA è stato ‘sacrificato’ SonarCloud poiché il plugin fornito dal sito stesso non risulta compatibile con questa versione. È stato tentato anche di fare un downgrade del plugin utilizzando la versione 3.7.0.1746 che dovrebbe essere quella compatibile con Java 8, ma niente. Le build precedentemente andate a buon fine utilizzavano Java 11 con cui OpenJPA funzionava grazie ad una modifica effettuata nel pom (modifica rimossa per far funzionare Badua).

Come già anticipato tramite mail, il report di Badua, in OpenJPA, non riporta coverage sui metodi testati nonostante la build vada a buon fine, non vengono segnalati errori ed i test vengono eseguiti con successo. Per questo motivo, nell’analisi dei metodi di OpenJPA, il tool non è stato inserito.

## 1. BookKeeper - ReadCache

Descrizione della classe ReadCache:

*“Uses the specified amount of memory and pairs it with a hashmap. The memory is splitted in multiple segments that are used in a ring-buffer fashion. When the read cache is full, the oldest segment is cleared and rotated to make space for new entries to be added to the read cache.”*

I metodi analizzati sono:

- void **put**(long ledgerId, long entryId, ByteBuf entry)
- ByteBuf **get**(long ledgerId, long entryId)

**void put(long ledgerId, long entryId, ByteBuf entry)**

Il metodo **put** viene utilizzato per inserire nella cache il contenuto del **ByteBuf entry**, identificato da **entryId** nel ledger specificato da **ledgerId**.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
long ledgerId	{≤0}, {>0}
long entryId	{≤0}, {>0}
ByteBuf entry	{null}, {valid_instance}, {invalid_instance} *invalid: entry con dimensione superiore a quella della cache istanziata

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
long ledgerId	-1, 0, 1
long entryId	-1, 0, 1
ByteBuf entry	Null, valid, invalid

Essendo presenti solo 3 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

ledgerId	entryId	entry	Output
-1	-1	null	NullPointerException
-1	-1	valid	IllegalArgumentException
-1	-1	invalid	Valid_Input
-1	0	null	NullPointerException
-1	0	valid	IllegalArgumentException
-1	0	invalid	Valid_Input
-1	1	null	NullPointerException
-1	1	valid	IllegalArgumentException
-1	1	invalid	Valid_Input
0	-1	null	NullPointerException
0	-1	valid	Valid_Input
0	-1	invalid	Valid_Input
0	0	null	NullPointerException
0	0	valid	Valid_Input

0	0	invalid	Valid_Input
0	1	null	NullPointerException
0	1	valid	Valid_Input
0	1	invalid	Valid_Input
1	-1	null	NullPointerException
1	-1	valid	Valid_Input
1	-1	invalid	Valid_Input
1	0	null	NullPointerException
1	0	valid	Valid_Input
1	0	invalid	Valid_Input
1	-1	null	NullPointerException
1	-1	valid	Valid_Input
1	-1	invalid	Valid_Input

NOTA: essendo il valore di ritorno del metodo `void`, per `Valid_Input` si intende che non viene sollevata alcun'eccezione.

Essendo il metodo `put` privo di documentazione, l'output atteso è stato recuperato utilizzando un approccio white-box. In particolare:

- `ledgerId` viene verificato se è  $\geq 0$  con il metodo `checkBiggerEqualZero` nella classe `ConcurrentLongLongPairHashMap`, può alzare `IllegalArgumentException`.
- `entryId` non ha alcun vincolo sul valore;
- `entry` non ha controlli esplicativi, ma con il metodo `readableBytes` può alzare l'eccezione `NullPointerException`.

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

Premessa: nei test è stata implementata un'istanza di `ReadCache` con capacità massima della cache pari a 4096 byte

Di conseguenza per `entry`, sono state considerate:

- `valid_instance` → buffer da 1024 byte
- `invalid_instance` → buffer da 5120 byte (maggiore della capacità massima della cache)

Inoltre i test sottintendono la validità del metodo `size` di `ReadCache`, utilizzato per convalidare lo stato della cache prima e dopo l'immissione dell'`entry`.

I test inizialmente sviluppati mirano a verificare che:

- In caso di input valido, la cache sia vuota prima di effettuare la `put` e che successivamente abbia dimensione pari a 1
- In caso ci si aspetti un'eccezione, il metodo restituisca effettivamente l'eccezione attesa

Effettivamente tutti i test si comportano come ci si aspettava e vengono eseguiti con successo.

Tuttavia la coverage risultante non è ottimale ([Figura 1](#)). In particolare, non si passa per un branch che verrebbe preso se non si riesce a scrivere l'`entry` nel primo segmento → è stata migliorata l'implementazione del test: nei casi con `entry` valida, dopo la prima, viene effettuata un ulteriore `put` e viene controllato se effettivamente la `size` della cache sia pari a 2.

Essendo che il doppio inserimento in cache con stessi parametri porta alla sovrascrittura dell'`entry` precedente con quella nuova, la seconda `put` viene effettuata con `ledgerId` incrementato di 1. Inoltre anche la dimensione del buffer è stata incrementata, altrimenti non avrebbe soddisfatto la condizione di branch.

La modifica porta ad una *statement coverage* del 100% e *branch coverage* del 83% ([Figura 2](#)).

Non risulta tuttavia possibile ottenere una *branch coverage* del 100% poiché è impossibile negare la condizione di branch a riga 123, ovvero (`offset + entrySize > segmentSize`), in quanto `offset` viene calcolato con

`currentSegmentOffset.getAndAdd(entrySize)`, ma dopo aver già effettuato a riga 103, `currentSegmentOffset.getAndAdd(alignedSize)`.

Per migliorare l'adeguatezza dei casi di test, è stata analizzata la *mutation coverage* ([Figura 3](#)). Non essendo ottimale sono stati aggiunti dei casi di test mirati ad uccidere i mutanti.

Tuttavia, nonostante i numerosi tentativi, è stato possibile eliminarne solo la mutazione relativa a riga 99. In particolare, rimane ignoto il motivo per cui la mutazione a riga 104 'changed conditional boundary' sopravviva nonostante siano presenti test che coprano la condizione in tutte e 3 le modalità:

- `(offset + entrySize > segmentSize)`
- `(offset + entrySize == segmentSize)`
- `(offset + entrySize < segmentSize)`

È stata analizzata la *data flow coverage* ([Figura 4](#)). Avendo raggiunto 70 su 78 *def-use coverage* e considerando le altre analisi appena discusse, si ritiene la test suite sufficiente. Il metodo `put` si comporta come da atteso.

#### **ByteBuf get(long ledgerId, long entryId)**

Il metodo `get` viene utilizzato per ottenere un oggetto `ByteBuf` con entry identificata da `entryId` nel ledger identificato con `ledgerId`.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>long ledgerId</code>	{≤0}, {>0}
<code>long entryId</code>	{≤0}, {>0}

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>long ledgerId</code>	-1, 0, 1
<code>long entryId</code>	-1, 0, 1

Essendo presenti solo 2 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

ledgerId	entryId	Output
-1	-1	IllegalArgumentException
-1	0	IllegalArgumentException
-1	1	IllegalArgumentException
0	-1	Valid_Input
0	0	Valid_Input
0	1	Valid_Input
1	-1	Valid_Input
1	0	Valid_Input
1	1	Valid_Input

NOTA: Non sapendo a priori se la entry sia presente in cache e quindi prevedere se il metodo ritorni un `ByteBuf` valido o null, per `Valid_Input` si intende che non viene sollevata alcun'eccezione.

Essendo il metodo `get` privo di documentazione, l'output atteso è stato recuperato utilizzando un approccio white-box. In particolare:

- `ledgerId` viene verificato se è  $\geq 0$  con il metodo `checkBiggerEqualZero` nella classe `ConcurrentLongLongPairHashMap`, può alzare `IllegalArgumentException`.
- `entryId` non ha alcun vincolo sul valore.

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

Premessa: i test sottintendono la validità del metodo `put` di `ReadCache`, utilizzato per inserire in cache una entry da leggere e verificarne la correttezza.

I test inizialmente sviluppati mirano a verificare che:

- In caso di input valido, il buffer restituito dal metodo sia uguale a quello immesso precedentemente (se presente) oppure null
- In caso ci si aspetti un'eccezione, il metodo restituiscia effettivamente l'eccezione attesa

Effettivamente tutti i test si comportano come ci si aspettava e vengono eseguiti con successo.

La test suite permette di ottenere il 100% sia di *statement* che di *branch coverage* ([Figura 5](#)).

È stata analizzata anche la *mutation coverage* ([Figura 6](#)), ma non tutti i mutanti vengono uccisi.

Tuttavia, la mutazione sopravvissuta di riga 161 relativa a `lock.readLock().unlock()`, è un comportamento atteso, in quanto anche senza togliere il lock di lettura, non vengono sollevate eccezioni e quindi il comportamento rimane inalterato.

Il mutante riferito a riga 147, è indipendente dai parametri e la variabile `cacheSegments` è privata. Quindi sulla chiamata `cacheSegments.size()` non è stato possibile utilizzare *mock* e risulta quindi impossibile assegnare alla variabile `i` un valore fuori range e quindi uccidere il mutante. Per le stesse considerazioni, neanche quello riferito a riga 148 può essere ucciso.

È stata analizzata la *data flow coverage* ([Figura 7](#)) e si raggiunge 31 su 32 *def-use coverage*.

Lasciando quindi la *mutation coverage* non ottimale, si ritiene sufficiente la test suite avendo raggiunto il 100% sia di *statement* che di *branch coverage* e quasi la totalità di *def-use coverage*. Il metodo `get` si comporta come da attese.

## 2. BookKeeper - Journal

I file *Journal* sono uno dei 3 tipi di file con cui i *Bookies* gestiscono i dati in maniera *log-structured*.

Un file *Journal* contiene i log delle transazioni di *BookKeeper*. Prima che avvenga qualsiasi aggiornamento di un ledger, il bookie si assicura che una transazione che descrive l'aggiornamento sia scritta su un archivio non volatile. Un nuovo file journal viene creato una volta che il bookie inizia o il file di journal più vecchio raggiunge la soglia delle dimensioni del file di journal.

I metodi analizzati sono:

- `List<Long> listJournalIds(File journalDir, JournalIdFilter filter)`
- `long scanJournal(long journalId, long journalPos, JournalScanner scanner)`

**`List<Long> listJournalIds(File journalDir, JournalIdFilter filter)`**

Il metodo `listJournalIds` elenca tutti gli ID dei journal presenti nella directory specificata (`journalDir`) che soddisfano il filtro fornito (`filter`). Parametri:

- `journalDir`: un oggetto `File` che rappresenta la directory dei journal.
- `filter`: un oggetto `JournalIdFilter` che rappresenta il filtro degli ID del journal.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>File journalDir</code>	{null}, {valid_instance}, {invalid_instance} *invalid: il percorso specificato non esiste o non è una directory
<code>JournalIdFilter filter</code>	{null}, {valid_instance}, {invalid_instance}

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>File journalDir</code>	Null, valid, invalid
<code>JournalIdFilter filter</code>	Null, valid, invalid

Essendo presenti solo 2 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

journalDir	filter	Output
Null	Null	NullPointerException
Null	valid	NullPointerException
Null	invalid	NullPointerException
valid	Null	Lista_valida
valid	valid	Lista_valida
valid	invalid	Lista_valida
invalid	Null	Lista_vuota
invalid	valid	Lista_vuota
invalid	invalid	Lista_vuota

NOTA: Non sapendo a priori se sono presenti degli ID dei journal nella directory specificata e quali siano se presenti, per `Lista_valida` si intende una lista con gli ID trovati o una lista vuota.

Essendo la documentazione del metodo `listJournalIds` povera, l'output atteso è stato recuperato utilizzando un approccio white-box. In particolare:

- su `journalDir` viene utilizzato il metodo `listFiles` che ritorna null in caso il file sia null o invalido. In questi casi viene restituita una lista vuota;
- `filter` non ha alcun vincolo sul valore, se null o invalido non viene applicato alcun filtro.

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

I test inizialmente sviluppati mirano a verificare che:

- In caso di `journalDir` non nullo, la lista restituita non sia null
- In caso ci si aspetti un'eccezione, il metodo restituisca effettivamente l'eccezione attesa

Per verificare al meglio se la lista (quando restituita) sia corretta, è stata creata una directory temporanea con dei file `txn` ed un `txt`. La `journalDir` valida viene istanziata proprio a partire da questa. Quello che ci si aspetta è il metodo ritorni la lista dei file `txn`. L'istanza di `journalDir` invalida è stata creata invece a partire da un path non valido ("... " nello specifico).

Come `filter` valido è stato considerato l'accettazione solo di id pari. Per la creazione di un `filter` invalido è stato utilizzato in `mock` che ritorna false sulla chiamata al metodo `accept`.

Tutti i test si comportano come ci si aspettava e vengono eseguiti con successo.

La test suite permette di ottenere una *statement coverage* del 100% e *branch coverage* del 91% ([Figura 8](#)), questo perché la `journalDir` valida è stata sempre creata a partire da una directory non vuota. Per raggiungere il 100% anche in *branch coverage* basta aggiungere la possibilità di avere una `journalDir` valida viene istanziata a partire da una directory vuota.

Per completezza quindi, sono state introdotte le seguenti combinazioni alla test suite:

journalDir	filter	Output
valid_empty	Null	Lista_vuota
valid_empty	valid	Lista_vuota
valid_empty	invalid	Lista_vuota

NOTA: queste non vanno aggiunte nella fase di *domain partitioning* poiché `valid_empty` rientra nel caso `valid_instance`.

La modifica porta anche la *branch coverage* al 100% ([Figura 9](#)).

Sono state analizzate anche la *mutation coverage* ([Figura 10](#)) e la *data flow coverage* ([Figura 11](#)) ed essendo ottimali si ritiene la test suite sufficiente. Il metodo `listJournalIds` si comporta come da attese.

**long scanJournal(long journalId, long journalPos, JournalScanner scanner)**

Il metodo `scanJournal` esegue la scansione del journal specificato da `journalId` partendo dall'offset `journalPos` e ritorna un long che rappresenta i byte letti.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>long journalId</code>	{≤0}, {>0}
<code>long journalPos</code>	{≤0}, {>0}
<code>JournalScanner scanner</code>	{null}, {valid_instance}, {invalid_instance}

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>long journalId</code>	-1, 0, 1, valid_id
<code>long journalPos</code>	-1, 0, 1
<code>JournalScanner scanner</code>	Null, valid, invalid

NOTA: Essendo `journalId` un id, per avere dei test significativi, è stato deciso di considerare in aggiunta anche `valid_id`. In questo modo viene testato sia come un long che come un oggetto.

Essendo presenti solo 3 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

journalId	journalPos	scanner	Output
-1	-1	null	Long
-1	-1	valid	Long
-1	-1	invalid	Long
-1	0	null	Long
-1	0	valid	Long
-1	0	invalid	Long
-1	1	null	Long
-1	1	valid	Long
-1	1	invalid	Long
0	-1	null	Long
0	-1	valid	Long
0	-1	invalid	Long
0	0	null	Long
0	0	valid	Long
0	0	invalid	Long
0	1	null	Long
0	1	valid	Long
0	1	invalid	Long
1	-1	null	Long

1	-1	valid	Long
1	-1	invalid	Long
1	0	null	Long
1	0	valid	Long
1	0	invalid	Long
1	1	null	Long
1	1	valid	Long
1	1	invalid	Long
valid	-1	null	NullPointerException
valid	-1	valid	Long
valid	-1	invalid	Exception
valid	0	null	NullPointerException
valid	0	valid	Long
valid	0	invalid	Exception
valid	1	null	Long
valid	1	valid	Long
valid	1	invalid	Long

Essendo la documentazione del metodo `scanJournal` povera, l'output atteso è stato recuperato utilizzando un approccio white-box. In particolare:

- i parametri `journalId` e `journalPos` non hanno alcun vincolo sul valore;
- su scanner viene chiamato il metodo `process` e può alzare le eccezioni:
  - `NullPointerException`;
  - `Exception` (in base a quella specificata nell'implementazione dell'interfaccia).

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

I test inizialmente sviluppati mirano a verificare che:

- In caso non ci si aspetti un'eccezione, il long restituito sia maggiore di quello restituito dallo stesso metodo utilizzato con un file dummy appositamente generato
- In caso ci si aspetti un'eccezione, il metodo restituisca effettivamente l'eccezione attesa

La test suite permette di ottenere una *statement coverage* del 70% e *branch coverage* del 50% ([Figura 12](#)).

Non risulta tuttavia possibile aumentare questi valori poiché:

- La condizione `lenBuff.remaining () != 0` a riga 820 non può essere mai verificata poiché il metodo `fullread`, precedentemente invocato, termina quando `bb.remaining () <= 0` (`bb` coincide con `lenBuff`)
- Non è stato possibile generare altre implementazioni in grado di soddisfare la condizione a riga 829 (e di conseguenza quella a riga 859, essendo `isPaddingRecord` impostata a true solo in quel branch). Quindi il branch compreso tra le righe 830 e 845 non è stato coperto.

È stata analizzata anche la *mutation coverage* ([Figura 13](#)) e non tutti i mutanti vengono uccisi.

Tuttavia le mutazioni sopravvissute, per i motivi descritti nell'analisi della coverage, non risultano raggiungibili con variazioni ai parametri o *mock*, quindi non posso essere uccise.

È stata analizzata la *data flow coverage* ([Figura 14](#)). Avendo raggiunto solo 60 su 97 *def-use coverage* ma considerando le discussioni appena fatte per *branch*, *statement* e *mutation coverage*, si ritiene la test suite sufficiente. Il metodo `scanJournal` si comporta come da attese.

### 3. OpenJPA - BrokerImpl

La classe `BrokerImpl` è una parte fondamentale di OpenJPA che gestisce tutte le operazioni di persistenza. Il suo compito principale è fornire un'interfaccia tra l'applicazione e il sistema di persistenza sottostante.

I metodi analizzati sono:

- `void handleCallbackExceptions (Exception[] excep, int mode)`
- `void throwNestedExceptions (List<Exception> excep, boolean datastore)`

#### `void handleCallbackExceptions (Exception[] excep, int mode)`

Il metodo `handleCallbackExceptions` viene utilizzato per gestire le eccezioni che si verificano durante i callback, in base alla modalità specificata.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>Exception[] excep</code>	{null}, {invalid_array}, {empty_array}, {valid_array} *invalid_array: array con un solo elemento nullo
<code>int mode</code>	{≤0}, {>0}

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>Exception[] excep</code>	Null, invalid_array, empty_array, valid_array
<code>int mode</code>	-1, 0, 1

Essendo presenti solo 2 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

excep	mode	Output
Null	-1	NullPointerException
Null	0	NullPointerException
Null	1	NullPointerException
invalid_array	-1	Valid_Input
invalid_array	0	NullPointerException
invalid_array	1	NullPointerException
empty_array	-1	Valid_Input
empty_array	0	Valid_Input
empty_array	1	Valid_Input
<code>valid_array -&gt;</code> {Exception("Dummy exception!")}	-1	Valid_Input
<code>valid_array -&gt;</code> {Exception("Dummy exception!")}	0	Valid_Input
<code>valid_array -&gt;</code> {Exception("Dummy exception!")}	1	Valid_Input

NOTA: essendo il valore di ritorno del metodo `void`, per `Valid_Input` si intende che non viene sollevata alcun'eccezione.

Essendo la documentazione del metodo `handleCallbackExceptions` povera, l'output atteso è stato recuperato utilizzando un approccio white-box. In particolare:

- sul parametro `excep` viene invocato il metodo `length` che può alzare `NullPointerException`;
- Il parametro `mode` non hanno alcun vincolo sul valore.

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

I test inizialmente sviluppati mirano a verificare che le eccezioni vengano alzate solo se attese. In tal caso si verifica inoltre che siano quelle previste.

La test suite permette di ottenere una *statement coverage* del 51% e *branch coverage* del 50% ([Figura 15](#)).

Per migliorare la qualità dei test, la test suite è stata ampliata osservando l'implementazione del metodo. In particolare:

- nei test precedentemente sviluppati, l'istanza valida di `exceps` è stata istanziata di dimensione unitaria → sono stati implementati dei test con `exceps` di dimensione maggiore;
- nei test precedentemente sviluppati, veniva creata un'eccezione `dummy` con `new Exception`, ma nel metodo viene controllato anche se l'eccezione è una `WrappedException` → sono stati implementati dei test con una `WrappedException`;
- è stato osservato che con il parametro `mode` vendono effettuati degli *AND* con dei flag → sono stati inclusi dei test in cui il parametro `mode` assume questi valori.

Inoltre, essendo i parametri `_flags` e `_log` privati:

- per accedere al branch relativo a riga 904, è stata utilizzata la riflessione per assegnare a `_flags` il valore di `FLAG_ACTIVE`;
- per accedere al branch relativo a riga 908, è stata utilizzata la riflessione per iniettare un `mock` sulla chiamata del metodo `isWarnEnabled` del parametro `_log`.

I test in cui vengono implementate queste ultime due configurazioni sono quelli contrassegnati con l'asterisco (\*).

In aggiunta ai precedenti, sono stati quindi introdotti i seguenti test:

<code>exceps</code>	<code>mode</code>	<code>Output</code>
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	-1	Valid_Input
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	0	Valid_Input
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	1	Valid_Input
<code>wrapped_exception_list -&gt;</code> <code>{WrappedException("Wrapped exception!")}</code>	-1	Valid_Input
<code>wrapped_exception_list -&gt;</code> <code>{WrappedException("Wrapped exception!")}</code>	0	WrappedException
<code>wrapped_exception_list -&gt;</code> <code>{WrappedException("Wrapped exception!")}</code>	1	WrappedException
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_IGNORE	Valid_Input
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_ROLLBACK	Valid_Input
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_ROLLBACK	Exception*
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_LOG	NullPointerException
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_LOG	Valid_Input*

<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	CALLBACK_RETHROW	CallbackException
--	------------------	-------------------

NOTA: le definizioni delle 4 “CALLBACK\_” sono specificate in CallbackModes.

Le modifiche portano la *statement coverage* al 95% e la *branch coverage* al 94% ([Figura 16](#)).

È stata analizzata anche la *mutation coverage* ([Figura 17](#)) ed essendo i mutanti sopravvissuti dipendenti dagli attributi privati `_flags` e `_log`, si ritiene la test suite sufficiente. Il metodo `handleCallbackExceptions` si comporta come da attese.

**void throwNestedExceptions(List<Exception> exeps, boolean datastore)**

Il metodo `throwNestedExceptions` viene utilizzato per gestire eccezioni annidate all'interno del broker.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
List<Exception> exeps	{null}, {empty_list}, {invalid_list}, {valid_list} *invalid_list: list con un solo elemento nullo
boolean datastore	False, True

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
List<Exception> exeps	Null, empty_list, invalid_list, single_exception_list, multiple_exception_list
boolean datastore	False, True

Essendo presenti solo 2 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

exeps	datastore	Output
Null	False	Valid_Input
Null	True	Valid_Input
empty_list	False	Valid_Input
empty_list	True	Valid_Input
invalid_list	False	UserException
invalid_list	True	NullPointerException
<code>single_exception_list -&gt;</code> <code>{Exception("Dummy exception!")}</code>	False	UserException
<code>single_exception_list -&gt;</code> <code>{Exception("Dummy exception!")}</code>	True	RuntimeException
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	False	UserException
<code>multiple_exception_list -&gt;</code> <code>{Exception("First exception!"),</code> <code>Exception("Second exception!")}</code>	True	StoreException

NOTA: essendo il valore di ritorno del metodo `void`, per `Valid_Input` si intende che non viene sollevata alcun'eccezione.

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

I test inizialmente sviluppati mirano a verificare che le eccezioni vengano alzate solo se attese. In tal caso si verifica inoltre che siano quelle previste.

La test suite permette di ottenere una *statement coverage* del 91% e *branch coverage* del 81% ([Figura 18](#)).

Per migliorare la qualità dei test, la test suite è stata ampliata osservando l'implementazione del metodo. In particolare, nei casi di `multiple_exception_list`, sono state aggiunte alla lista:

- un'eccezione di tipo `OpenJPAException` con il parametro `fatal` impostato su `false`;
- un'eccezione di tipo `OpenJPAException` con il parametro `fatal` impostato su `true` tramite il metodo `setFatal`.

Le modifiche portano sia la *statement coverage* che la *branch coverage* al 100% ([Figura 19](#)).

È stata analizzata anche la *mutation coverage* ([Figura 20](#)) e c'è un unico mutante sopravvissuto. Tuttavia, nonostante le modifiche apportate ai test, non è stato possibile ucciderlo. In particolare, rimane ignoto il motivo per cui la mutazione '*negated conditional*' sopravviva nonostante siano presenti test che coprano il branch sia con condizione vera che falsa. Inoltre è stato anche controllato che le eccezioni siano correttamente '*fatal*'.

Considerando le analisi appena discusse, si ritiene la test suite sufficiente. Il metodo `throwNestedExceptions` si comporta come da attese.

## 4. OpenJPA - ProxyManagerImpl

La classe `ProxyManagerImpl` è responsabile della gestione dei proxy per le entità persistenti. Attraverso i proxy, consente di evitare il caricamento eccessivo dei dati dal database e fornisce un meccanismo per ottimizzare le operazioni di accesso ai dati.

I metodi analizzati sono:

- `Proxy newCustomProxy(Object orig, boolean autoOff)`
- `Object copyArray(Object orig)`

### `Proxy newCustomProxy(Object orig, boolean autoOff)`

Descrizione della classe `newCustomProxy`:

“Return a proxy for the given object, or null if this manager cannot proxy the object.”

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>Object orig</code>	{null}, {invalid_instance}, {valid_instance}
<code>boolean autoOff</code>	False, True

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>Object orig</code>	Null, invalid, valid
<code>boolean autoOff</code>	False, True

Essendo presenti solo 2 parametri con poche classi di equivalenza, è stato deciso di optare per un approccio multidimensionale.

I valori di output attesi dai test progettati sono:

orig	autoOff	Output
Null	False	null
Null	True	null
invalid -> False	False	null
invalid -> False	True	null
valid -> Proxy_instance	False	Proxy_instance
valid -> Proxy_instance	True	Proxy_instance

## Analisi dei test

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

I test inizialmente sviluppati mirano a verificare che, se previsto, il metodo ritorni un'istanza valida di `Proxy`.

La test suite permette di ottenere una *statement coverage* del 13% e *branch coverage* del 25% ([Figura 21](#)).

Per migliorare la qualità dei test, la test suite è stata ampliata osservando l'implementazione del metodo. In particolare, sono state aggiunte:

orig	autoOff	Output
Null	False	null
PersistenceCapable_instance	False	null
"That's a string!"	False	null
0	False	null
Collection_instance	False	Proxy_instance
Map_instance	False	Proxy_instance
Date_instance	False	Proxy_instance
Calendar_instance	False	Proxy_instance
Dummy_instance	False	null
SortedSet_instance	False	Proxy_instance
SortedMap_instance	False	Proxy_instance
Timestamp_instance	False	Proxy_instance
Null	True	Proxy_instance
PersistenceCapable_instance	True	null
"That's a string!"	True	null
0	True	null
Collection_instance	True	Proxy_instance
Map_instance	True	Proxy_instance
Date_instance	True	Proxy_instance
Calendar_instance	True	Proxy_instance
Dummy_instance	True	null
SortedSet_instance	True	Proxy_instance
SortedMap_instance	True	Proxy_instance
Timestamp_instance	True	Proxy_instance

Le modifiche portano la *statement coverage* al 97% e la *branch coverage* al 95% ([Figura 22](#)).

Non risulta possibile aumentare questi valori essendo il metodo `getFactoryProxyCollection` privato e non potendo quindi utilizzare *mock*.

È stata analizzata la *mutation coverage* ([Figura 23](#)). Essendoci 5 sopravvissuti, sono stati aggiunti degli *assert* mirati ad ucciderli. Tuttavia è stato possibile eliminarne solo 3 ([Figura 24](#)), in particolare:

- Nonostante l'*assert* mirato al controllo sul suo valore ritornato, il mutante riferito a riga 325 sopravvive;
- Quello riferito a riga 336, è collegato all'irraggiungibilità del metodo `getFactoryProxyCollection`.

Considerando le analisi appena discusse, si ritiene la test suite sufficiente. Il metodo `newCustomProxy` si comporta come da attese.

## **Object copyArray (Object orig)**

Descrizione della classe copyArray:

*“Return a new array of the same component type as the given array and containing the same elements.  
Works for both primitive and object array types.”*

Dalla documentazione si evince che, nonostante il parametro `orig` sia un `Object`, il metodo richiede un array, per questo motivo è stato trattato come tale.

Partizione in classi di equivalenza:

Parametro	Classi di equivalenza
<code>Object orig</code>	{null}, {invalid_instance}, {empty_array}, {valid_array} *invalid_instance: Object diverso da un array

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Parametro	Boundaries
<code>Object orig</code>	Null, invalid_instance, empty_array, valid_array

I valori di output attesi dai test progettati sono:

orig	Output
Null	Null
invalid_instance -> String.class	IllegalArgumentException
empty_array	{}
valid_array -> {1, 2, 3}	{1, 2, 3}

## **Analisi dei test**

Sono stati implementati i test con i parametri descritti nel *category partitioning*.

La test suite permette di ottenere il 100% sia di *statement* che di *branch coverage* ([Figura 25](#)).

È stata analizzata anche la *mutation coverage* ([Figura 26](#)) e c'è un mutante sopravvissuto 'replaced return value with null'. Per risolvere il problema è stato modificato lo schema degli assert in seguito alla chiamata del metodo e i cambiamenti apportati riescono ad uccidere anche questo mutante ([Figura 27](#)).

Si ritiene la test suite sufficiente. Il metodo `copyArray` si comporta come da atteso.

## 5. OpenJPA - Integration Test

### LifecycleEventManager & BrokerImpl

È stato effettuato un test di integrazione tra le classi `BrokerImpl` e `LifecycleEventManager` utilizzando l'interazione tra i rispettivi metodi `handleCallbackExceptions` e `fireEvent`.

In particolare, il metodo `fireEvent` “*fire lifecycle event to all registered listeners*”, restituisce una lista di eccezioni che può essere passata come parametro a `handleCallbackExceptions` che si occupa di gestirle.

Le segnature dei metodi sono le seguenti:

- `Exception[] fireEvent(Object source, Object related, ClassMetaData meta, int type)`
- `void handleCallbackExceptions(Exception[] exeps, int mode)`

NOTA: sarebbe consone effettuare *integration testing* su classi di cui è stato precedentemente effettuato *unit testing*, tuttavia per uno studio approfondito del metodo `fireEvent`, sarebbe necessario uno studio più attendo sul funzionamento del sistema, cosa non banale. Si ipotizza quindi che il metodo sia stato testato e si comporti come da attese.

Essendo lo scopo del test d'integrazione la verifica che le classi interagiscano in maniera corretta/desiderata, l'analisi è stata svolta sulla variabile comune, ovvero la lista di eccezioni.

Partizione in classi di equivalenza:

Variabile	Classi di equivalenza
<code>Exception[] exeps</code>	{empty_array}, {valid_array}

Non sono stati presi in considerazione `null` e `invalid_array` poiché, al momento dello studio, il metodo `fireEvent` non può tornare nessuno dei due valori.

Scelta dei valori rappresentativi di ogni partizione tramite Boundary Values Analysis:

Variabile	Boundaries
<code>Exception[] exeps</code>	<code>empty_array, single_exception_array,</code> <code>multiple_exception_array, wrapped_exception_array</code>

Per ognuno dei due metodi, sono state quindi scelte delle configurazioni sui parametri e sono stati utilizzati i *mock* in modo che:

- La lista restituita da `fireEvent` sia conforme ai 4 *boundaries* (almeno un caso di test per ognuno);
- Le eventuali eccezioni alzate da `handleCallbackExceptions` siano in linea con le modalità fornite ([riferimento all'analisi di handleCallbackExceptions](#)).

### Analisi dei test

I test implementati vengo eseguiti con successo e si comportano come da attese, in particolare `fireEvent` restituisce liste che risultano valide per `handleCallbackExceptions`.

È possibile affermare quindi che i metodi delle due classi comunicano nel modo corretto ed il risultato della loro interazione coincide con quello atteso.

## 6. Allegati

*Figura\_1:* Analisi della coverage del metodo put (Jacoco)

[\[torna su\]](#)

```
public void put(long ledgerId, long entryId, ByteBuf entry) {
    int entrySize = entry.readableBytes();
    int alignedSize = align64(entrySize);

    lock.readLock().lock();

    try {
        if (entrySize > segmentSize) {
            log.warn("entrySize {} > segmentSize {}, skip update read cache!", entrySize, segmentSize);
            return;
        }
        int offset = currentSegmentOffset.getAndAdd(alignedSize);
        if (offset + entrySize > segmentSize) {
            // Roll-over the segment (outside the read-lock)
        } else {
            // Copy entry into read cache segment
            cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(),
                entry.readableBytes());
            cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize);
            return;
        }
    } finally {
        lock.readLock().unlock();
    }

    // We could not insert in segment, we to get the write lock and roll-over to
    // next segment
    lock.writeLock().lock();

    try {
        int offset = currentSegmentOffset.getAndAdd(entrySize);
        if (offset + entrySize > segmentSize) {
            // Rollover to next segment
            currentSegmentIdx = (currentSegmentIdx + 1) % cacheSegments.size();
            currentSegmentOffset.set(alignedSize);
            cacheIndexes.get(currentSegmentIdx).clear();
            offset = 0;
        }

        // Copy entry into read cache segment
        cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(), entry.readableBytes());
        cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize);
    } finally {
        lock.writeLock().unlock();
    }
}
```

**Figura\_2:** Percentuale e analisi della coverage del metodo put (Jacoco) post miglioramento

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">put(long, long, ByteBuf)</a>	100%	83%		
<pre> 92.     public void put(long ledgerId, long entryId, ByteBuf entry) { 93.         int entrySize = entry.readableBytes(); 94.         int alignedSize = align64(entrySize); 95. 96.         lock.readLock().lock(); 97. 98.         try { 99.             if (entrySize &gt; segmentSize) { 100.                 log.warn("entrySize {} &gt; segmentSize {}, skip update read cache!", entrySize, segmentSize); 101.                 return; 102.             } 103.             int offset = currentSegmentOffset.getAndAdd(alignedSize); 104.             if (offset + entrySize &gt; segmentSize) { 105.                 // Roll-over the segment (outside the read-lock) 106.             } else { 107.                 // Copy entry into read cache segment 108.                 cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(), 109.                                 entry.readableBytes()); 110.                 cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize); 111.                 return; 112.             } 113.         } finally { 114.             lock.readLock().unlock(); 115.         } 116. 117.         // We could not insert in segment, we to get the write lock and roll-over to 118.         // next segment 119.         lock.writeLock().lock(); 120. 121.         try { 122.             int offset = currentSegmentOffset.getAndAdd(entrySize); 123.             if (offset + entrySize &gt; segmentSize) { 124.                 // Rollover to next segment 125.                 currentSegmentIdx = (currentSegmentIdx + 1) % cacheSegments.size(); 126.                 currentSegmentOffset.set(alignedSize); 127.                 cacheIndexes.get(currentSegmentIdx).clear(); 128.                 offset = 0; 129.             } 130. 131.             // Copy entry into read cache segment 132.             cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(), entry.readableBytes()); 133.             cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize); 134.         } finally { 135.             lock.writeLock().unlock(); 136.         } 137.     } </pre>				

*Figura\_3*: Analisi della mutation coverage del metodo put (Pit)

[[torna su](#)]

```
92     public void put(long ledgerId, long entryId, ByteBuf entry) {
93         int entrySize = entry.readableBytes();
94         int alignedSize = align64(entrySize);
95
96     1       lock.readLock().lock();
97
98         try {
99     2       if (entrySize > segmentSize) {
100             log.warn("entrySize {} > segmentSize {}, skip update read cache!", entrySize, segmentSize);
101             return;
102         }
103         int offset = currentSegmentOffset.getAndAdd(alignedSize);
104     3       if (offset + entrySize > segmentSize) {
105             // Roll-over the segment (outside the read-lock)
106         } else {
107             // Copy entry into read cache segment
108             cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(),
109                     entry.readableBytes());
110             cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize);
111             return;
112         }
113     } finally {
114     1       lock.readLock().unlock();
115     }
116
117         // We could not insert in segment, we to get the write lock and roll-over to
118         // next segment
119     1       lock.writeLock().lock();
120
121         try {
122             int offset = currentSegmentOffset.getAndAdd(entrySize);
123     3       if (offset + entrySize > segmentSize) {
124             // Rollover to next segment
125     2       currentSegmentIdx = (currentSegmentIdx + 1) % cacheSegments.size();
126     1       currentSegmentOffset.set(alignedSize);
127     1       cacheIndexes.get(currentSegmentIdx).clear();
128             offset = 0;
129         }
130
131             // Copy entry into read cache segment
132             cacheSegments.get(currentSegmentIdx).setBytes(offset, entry, entry.readerIndex(), entry.readableBytes());
133             cacheIndexes.get(currentSegmentIdx).put(ledgerId, entryId, offset, entrySize);
134         } finally {
135     1       lock.writeLock().unlock();
136     }
137     }
```

**Figura\_4:** Analisi data flow coverage del metodo put (Badua)

[[torna su](#)]

```

▼<method name="put" desc="(JJLio/netty/buffer/ByteBuf;)V">
<du var="this" def="93" use="99" target="100" covered="1"/>
<du var="this" def="93" use="99" target="103" covered="1"/>
<du var="this" def="93" use="103" covered="1"/>
<du var="this" def="93" use="104" target="104" covered="1"/>
<du var="this" def="93" use="104" target="108" covered="1"/>
<du var="this" def="93" use="108" covered="1"/>
<du var="this" def="93" use="110" covered="1"/>
<du var="this" def="93" use="114" covered="1"/>
<du var="this" def="93" use="114" covered="1"/>
<du var="this" def="93" use="119" covered="1"/>
<du var="this" def="93" use="122" covered="1"/>
<du var="this" def="93" use="123" target="125" covered="1"/>
<du var="this" def="93" use="123" target="132" covered="0"/>
<du var="this" def="93" use="132" covered="1"/>
<du var="this" def="93" use="133" covered="1"/>
<du var="this" def="93" use="135" covered="1"/>
<du var="this" def="93" use="125" covered="1"/>
<du var="this" def="93" use="126" covered="1"/>
<du var="this" def="93" use="127" covered="1"/>
<du var="this" def="93" use="100" covered="1"/>
<du var="this" def="93" use="114" covered="1"/>
<du var="ledgerId" def="93" use="110" covered="1"/>
<du var="ledgerId" def="93" use="133" covered="1"/>
<du var="entryId" def="93" use="110" covered="1"/>
<du var="entryId" def="93" use="133" covered="1"/>
<du var="entry" def="93" use="108" covered="1"/>
<du var="entry" def="93" use="132" covered="1"/>
<du var="this.lock" def="93" use="114" covered="1"/>
<du var="this.lock" def="93" use="114" covered="1"/>
<du var="this.lock" def="93" use="119" covered="1"/>
<du var="this.lock" def="93" use="135" covered="1"/>
<du var="this.lock" def="93" use="114" covered="1"/>
<du var="this.segmentSize" def="93" use="99" target="100" covered="1"/>
<du var="this.segmentSize" def="93" use="99" target="103" covered="1"/>
<du var="this.segmentSize" def="93" use="104" target="104" covered="1"/>
<du var="this.segmentSize" def="93" use="104" target="108" covered="1"/>
<du var="this.segmentSize" def="93" use="123" target="125" covered="1"/>
<du var="this.segmentSize" def="93" use="123" target="132" covered="0"/>
<du var="this.segmentSize" def="93" use="100" covered="1"/>
<du var="log" def="93" use="100" covered="1"/>
<du var="this.currentSegmentOffset" def="93" use="103" covered="1"/>
<du var="this.currentSegmentOffset" def="93" use="122" covered="1"/>
<du var="this.currentSegmentOffset" def="93" use="126" covered="1"/>
<du var="this.cacheSegments" def="93" use="108" covered="1"/>
<du var="this.cacheSegments" def="93" use="132" covered="1"/>
<du var="this.cacheSegments" def="93" use="125" covered="1"/>
<du var="this.currentSegmentIdx" def="93" use="108" covered="1"/>
<du var="this.currentSegmentIdx" def="93" use="110" covered="1"/>
<du var="this.currentSegmentIdx" def="93" use="132" covered="0"/>
<du var="this.currentSegmentIdx" def="93" use="133" covered="0"/>
<du var="this.currentSegmentIdx" def="93" use="125" covered="1"/>
<du var="this.cacheIndexes" def="93" use="110" covered="1"/>
<du var="this.cacheIndexes" def="93" use="133" covered="1"/>
<du var="this.cacheIndexes" def="93" use="127" covered="1"/>
<du var="entrySize" def="93" use="99" target="100" covered="1"/>
<du var="entrySize" def="93" use="99" target="103" covered="1"/>
<du var="entrySize" def="93" use="104" target="104" covered="1"/>
<du var="entrySize" def="93" use="104" target="108" covered="1"/>
<du var="entrySize" def="93" use="110" covered="1"/>
<du var="entrySize" def="93" use="122" covered="1"/>
<du var="entrySize" def="93" use="123" target="125" covered="1"/>
<du var="entrySize" def="93" use="123" target="132" covered="0"/>
<du var="entrySize" def="93" use="133" covered="1"/>
<du var="entrySize" def="93" use="100" covered="1"/>
<du var="alignedSize" def="94" use="103" covered="1"/>
<du var="alignedSize" def="94" use="126" covered="1"/>
```

```
<du var="offset" def="103" use="104" target="104" covered="1"/>
<du var="offset" def="103" use="104" target="108" covered="1"/>
<du var="offset" def="103" use="108" covered="1"/>
<du var="offset" def="103" use="110" covered="1"/>
<du var="offset" def="122" use="123" target="125" covered="1"/>
<du var="offset" def="122" use="123" target="132" covered="0"/>
<du var="offset" def="122" use="132" covered="0"/>
<du var="offset" def="122" use="133" covered="0"/>
<du var="this.currentSegmentIdx" def="125" use="132" covered="1"/>
<du var="this.currentSegmentIdx" def="125" use="133" covered="1"/>
<du var="offset" def="128" use="132" covered="1"/>
<du var="offset" def="128" use="133" covered="1"/>
<counter type="DU" missed="8" covered="70"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>
```

*Figura\_5:* Percentuale e analisi della coverage del metodo get (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">get(long, long)</a>		100%		100%

```

139.     public ByteBuf get(long ledgerId, long entryId) {
140.         lock.readLock().lock();
141.
142.         try {
143.             // We need to check all the segments, starting from the current one and looking
144.             // backward to minimize the
145.             // checks for recently inserted entries
146.             int size = cacheSegments.size();
147.             for (int i = 0; i < size; i++) {
148.                 int segmentIdx = (currentSegmentIdx + (size - i)) % size;
149.
150.                 LongPair res = cacheIndexes.get(segmentIdx).get(ledgerId, entryId);
151.                 if (res != null) {
152.                     int entryOffset = (int) res.first;
153.                     int entryLen = (int) res.second;
154.
155.                     ByteBuf entry = allocator.buffer(entryLen, entryLen);
156.                     entry.writeBytes(cacheSegments.get(segmentIdx), entryOffset, entryLen);
157.                     return entry;
158.                 }
159.             }
160.         } finally {
161.             lock.readLock().unlock();
162.         }
163.
164.         // Entry not found in any segment
165.         return null;
166.     }

```

*Figura\_6:* Analisi della mutation coverage del metodo get (Pit)

[\[torna su\]](#)

```

139     public ByteBuf get(long ledgerId, long entryId) {
140.         lock.readLock().lock();
141.
142.         try {
143.             // We need to check all the segments, starting from the current one and looking
144.             // backward to minimize the
145.             // checks for recently inserted entries
146.             int size = cacheSegments.size();
147.             for (int i = 0; i < size; i++) {
148.                 int segmentIdx = (currentSegmentIdx + (size - i)) % size;
149.
150.                 LongPair res = cacheIndexes.get(segmentIdx).get(ledgerId, entryId);
151.                 if (res != null) {
152.                     int entryOffset = (int) res.first;
153.                     int entryLen = (int) res.second;
154.
155.                     ByteBuf entry = allocator.buffer(entryLen, entryLen);
156.                     entry.writeBytes(cacheSegments.get(segmentIdx), entryOffset, entryLen);
157.                     return entry;
158.                 }
159.             }
160.         } finally {
161.             lock.readLock().unlock();
162.         }
163.
164.         // Entry not found in any segment
165.         return null;
166.     }

```

Figura\_7: Analisi data flow coverage del metodo get (Badua)

[[torna su](#)]

```
▼<method name="get" desc="(JJ)Lio/netty/buffer/ByteBuf;">
  <du var="this" def="140" use="161" covered="1"/>
  <du var="this" def="140" use="148" covered="1"/>
  <du var="this" def="140" use="150" covered="1"/>
  <du var="this" def="140" use="155" covered="1"/>
  <du var="this" def="140" use="156" covered="1"/>
  <du var="this" def="140" use="161" covered="1"/>
  <du var="ledgerId" def="140" use="150" covered="1"/>
  <du var="entryId" def="140" use="150" covered="1"/>
  <du var="this.lock" def="140" use="161" covered="1"/>
  <du var="this.lock" def="140" use="161" covered="1"/>
  <du var="this.cacheSegments" def="140" use="156" covered="1"/>
  <du var="this.currentSegmentIdx" def="140" use="148" covered="1"/>
  <du var="this.cacheIndexes" def="140" use="150" covered="1"/>
  <du var="thisallocator" def="140" use="155" covered="1"/>
  <du var="size" def="146" use="147" target="148" covered="1"/>
  <du var="size" def="146" use="147" target="161" covered="1"/>
  <du var="size" def="146" use="148" covered="1"/>
  <du var="i" def="147" use="147" target="148" covered="1"/>
  <du var="i" def="147" use="147" target="161" covered="0"/>
  <du var="i" def="147" use="148" covered="1"/>
  <du var="i" def="147" use="147" covered="1"/>
  <du var="segmentIdx" def="148" use="156" covered="1"/>
  <du var="res" def="150" use="151" target="152" covered="1"/>
  <du var="res" def="150" use="151" target="147" covered="1"/>
  <du var="res" def="150" use="152" covered="1"/>
  <du var="res" def="150" use="153" covered="1"/>
  <du var="res.first" def="150" use="152" covered="1"/>
  <du var="res.second" def="150" use="153" covered="1"/>
  <du var="i" def="147" use="147" target="148" covered="1"/>
  <du var="i" def="147" use="147" target="161" covered="1"/>
  <du var="i" def="147" use="148" covered="1"/>
  <du var="i" def="147" use="147" covered="1"/>
  <counter type="DU" missed="1" covered="31"/>
  <counter type="METHOD" missed="0" covered="1"/>
</method>
```

*Figura\_8*: Percentuale e analisi della coverage del metodo listJournal (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
listJournalIds(File, Journal.JournalIdFilter)	█	100%	█	91%

```

103.     public static List<Long> listJournalIds(File journalDir, JournalIdFilter filter) {
104.         File[] logFiles = journalDir.listFiles();
105.         if (logFiles == null || logFiles.length == 0) {
106.             return Collections.emptyList();
107.         }
108.         List<Long> logs = new ArrayList<Long>();
109.         for (File f: logFiles) {
110.             String name = f.getName();
111.             if (!name.endsWith(".txn")) {
112.                 continue;
113.             }
114.             String idString = name.split("\\.")[0];
115.             long id = Long.parseLong(idString, 16);
116.             if (filter != null) {
117.                 if (filter.accept(id)) {
118.                     logs.add(id);
119.                 }
120.             } else {
121.                 logs.add(id);
122.             }
123.         }
124.         Collections.sort(logs);
125.         return logs;
126.     }

```

*Figura\_9*: Percentuale della coverage del metodo listJournal (Jacoco) post miglioramento

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
listJournalIds(File, Journal.JournalIdFilter)	█	100%	█	100%

*Figura\_10*: Analisi della mutation coverage del metodo listJournal (Pit)

[\[torna su\]](#)

```

103     public static List<Long> listJournalIds(File journalDir, JournalIdFilter filter) {
104         File[] logFiles = journalDir.listFiles();
105         if (logFiles == null || logFiles.length == 0) {
106             return Collections.emptyList();
107         }
108         List<Long> logs = new ArrayList<Long>();
109         for (File f: logFiles) {
110             String name = f.getName();
111             if (!name.endsWith(".txn")) {
112                 continue;
113             }
114             String idString = name.split("\\.")[0];
115             long id = Long.parseLong(idString, 16);
116             if (filter != null) {
117                 if (filter.accept(id)) {
118                     logs.add(id);
119                 }
120             } else {
121                 logs.add(id);
122             }
123         }
124         Collections.sort(logs);
125         return logs;
126     }

```

```
▼<class name="org/apache/bookkeeper/bookie/Journal">
  ▼<method name="listJournalIds" desc="(Ljava/io/File;Lorg/apache/bookk
    <du var="filter" def="104" use="116" target="117" covered="1"/>
    <du var="filter" def="104" use="116" target="121" covered="1"/>
    <du var="filter" def="104" use="117" target="118" covered="1"/>
    <du var="filter" def="104" use="117" target="109" covered="1"/>
    <du var="logFiles" def="104" use="105" target="105" covered="1"/>
    <du var="logFiles" def="104" use="105" target="106" covered="1"/>
    <du var="logFiles" def="104" use="105" target="106" covered="1"/>
    <du var="logFiles" def="104" use="105" target="108" covered="1"/>
    <du var="logFiles" def="104" use="109" covered="1"/>
    <du var="logs" def="108" use="124" covered="1"/>
    <du var="logs" def="108" use="125" covered="1"/>
    <du var="logs" def="108" use="121" covered="1"/>
    <du var="logs" def="108" use="118" covered="1"/>
    <du var="name" def="110" use="111" target="112" covered="1"/>
    <du var="name" def="110" use="111" target="114" covered="1"/>
    <du var="name" def="110" use="114" covered="1"/>
    <du var="id" def="115" use="121" covered="1"/>
    <du var="id" def="115" use="117" target="118" covered="1"/>
    <du var="id" def="115" use="117" target="109" covered="1"/>
    <du var="id" def="115" use="118" covered="1"/>
    <counter type="DU" missed="1" covered="30"/>
    <counter type="METHOD" missed="0" covered="1"/>
  </method>
```

*Figura\_12:* Percentuale e analisi della coverage del metodo scanJournal (Jacoco)

[[torna su](#)]

Element	Missed Instructions	Cov.	Missed Branches	Cov.
scanJournal(long, long, Journal.JournalScanner)	██████	70%	██████	50%
<pre>800.     public long scanJournal(long journalId, long journalPos, JournalScanner scanner) 801.     throws IOException { 802.         JournalChannel recLog; 803.         if (journalPos &lt;= 0) { 804.             recLog = new JournalChannel(journalDirectory, journalId, journalPreAllocSize, journalWriteBufferSize, 805.   conf, fileChannelProvider); 806.         } else { 807.             recLog = new JournalChannel(journalDirectory, journalId, journalPreAllocSize, journalWriteBufferSize, 808.   journalPos, conf, fileChannelProvider); 809.         } 810.         int journalVersion = recLog.getFormatVersion(); 811.         try { 812.             ByteBuffer lenBuff = ByteBuffer.allocate(4); 813.             ByteBuffer recBuff = ByteBuffer.allocate(64 * 1024); 814.             while (true) { 815.                 // entry start offset 816.                 long offset = recLog.fc.position(); 817.                 // start reading entry 818.                 lenBuff.clear(); 819.                 fullRead(recLog, lenBuff); 820.                 if (lenBuff.remaining() != 0) { 821.                     break; 822.                 } 823.                 lenBuff.flip(); 824.                 int len = lenBuff.getInt(); 825.                 if (len == 0) { 826.                     break; 827.                 } 828.                 boolean isPaddingRecord = false; 829.                 if (len &lt; 0) { 830.                     if (len == PADDING_MASK &amp;&amp; journalVersion &gt;= JournalChannel.V5) { 831.                         // skip padding bytes 832.                         lenBuff.clear(); 833.                         fullRead(recLog, lenBuff); 834.                         if (lenBuff.remaining() != 0) { 835.                             break; 836.                         } 837.                         lenBuff.flip(); 838.                         len = lenBuff.getInt(); 839.                         if (len == 0) { 840.                             continue; 841.                         } 842.                         isPaddingRecord = true; 843.                     } else { 844.                         LOG.error("Invalid record found with negative length: {}", len); 845.                         throw new IOException("Invalid record found with negative length " + len); 846.                     } 847.                 } 848.                 recBuff.clear(); 849.                 if (recBuff.remaining() &lt; len) { 850.                     recBuff = ByteBuffer.allocate(len); 851.                 } 852.                 recBuff.limit(len); 853.                 if (fullRead(recLog, recBuff) != len) { 854.                     // This seems scary, but it just means that this is where we 855.                     // left off writing 856.                     break; 857.                 } 858.                 recBuff.flip(); 859.                 if (!isPaddingRecord) { 860.                     scanner.process(journalVersion, offset, recBuff); 861.                 } 862.             } 863.             return recLog.fc.position(); 864.         } finally { 865.             recLog.close(); 866.         } 867.     }</pre>				

**Figura\_13:** Analisi della mutation coverage del metodo scanJournal (Pit)

[\[torna su\]](#)

```
800     public long scanJournal(long journalId, long journalPos, JournalScanner scanner)
801         throws IOException {
802         JournalChannel recLog;
803 [2]     if (journalPos <= 0) {
804             recLog = new JournalChannel(journalDirectory, journalId, journalPreAllocSize, journalWriteBufferSize,
805                                         conf, fileChannelProvider);
806         } else {
807             recLog = new JournalChannel(journalDirectory, journalId, journalPreAllocSize, journalWriteBufferSize,
808                                         journalPos, conf, fileChannelProvider);
809         }
810         int journalVersion = recLog.getFormatVersion();
811         try {
812             ByteBuffer lenBuff = ByteBuffer.allocate(4);
813             ByteBuffer recBuff = ByteBuffer.allocate(64 * 1024);
814             while (true) {
815                 // entry start offset
816                 long offset = recLog.fc.position();
817                 // start reading entry
818                 lenBuff.clear();
819                 fullRead(recLog, lenBuff);
820 [1]             if (lenBuff.remaining() != 0) {
821                 break;
822             }
823             lenBuff.flip();
824             int len = lenBuff.getInt();
825 [1]             if (len == 0) {
826                 break;
827             }
828             boolean isPaddingRecord = false;
829 [2]             if (len < 0) {
830 [3]                 if (len == PADDING_MASK && journalVersion >= JournalChannel.V5) {
831                     // skip padding bytes
832                     lenBuff.clear();
833                     fullRead(recLog, lenBuff);
834 [1]                     if (lenBuff.remaining() != 0) {
835                         break;
836                     }
837                     lenBuff.flip();
838                     len = lenBuff.getInt();
839 [1]                     if (len == 0) {
840                         continue;
841                     }
842                     isPaddingRecord = true;
843                 } else {
844                     LOG.error("Invalid record found with negative length: {}", len);
845                     throw new IOException("Invalid record found with negative length " + len);
846                 }
847             }
848             recBuff.clear();
849 [2]             if (recBuff.remaining() < len) {
850                 recBuff = ByteBuffer.allocate(len);
851             }
852             recBuff.limit(len);
853 [1]             if (fullRead(recLog, recBuff) != len) {
854                 // This seems scary, but it just means that this is where we
855                 // left off writing
856                 break;
857             }
858             recBuff.flip();
859 [1]             if (!isPaddingRecord) {
860 [1]                 scanner.process(journalVersion, offset, recBuff);
861             }
862         }
863 [1]         return recLog.fc.position();
864     } finally {
865 [1]         recLog.close();
866     }
867 }
```

**Figura\_14:** Analisi data flow coverage del metodo scanJournal (Badua)

[[torna su](#)]

```

▼<method name="scanJournal" desc="(JJLorg/apache/bookkeeper/bookie/Journal$JournalScanner;)J">
<du var="this" def="803" use="807" covered="1"/>
<du var="this" def="803" use="804" covered="1"/>
<du var="journalId" def="803" use="807" covered="1"/>
<du var="journalId" def="803" use="804" covered="1"/>
<du var="journalPos" def="803" use="803" target="804" covered="1"/>
<du var="journalPos" def="803" use="803" target="807" covered="1"/>
<du var="journalPos" def="803" use="807" covered="1"/>
<du var="scanner" def="803" use="860" covered="1"/>
<du var="this.journalDirectory" def="803" use="807" covered="1"/>
<du var="this.journalDirectory" def="803" use="804" covered="1"/>
<du var="this.journalPreAllocSize" def="803" use="807" covered="1"/>
<du var="this.journalPreAllocSize" def="803" use="804" covered="1"/>
<du var="this.journalWriteBufferSize" def="803" use="807" covered="1"/>
<du var="this.journalWriteBufferSize" def="803" use="804" covered="1"/>
<du var="this.conf" def="803" use="807" covered="1"/>
<du var="this.conf" def="803" use="804" covered="1"/>
<du var="this.fileChannelProvider" def="803" use="807" covered="1"/>
<du var="this.fileChannelProvider" def="803" use="804" covered="1"/>
<du var="LOG" def="803" use="844" covered="0"/>
<du var="recLog" def="804" use="810" covered="1"/>
<du var="recLog" def="804" use="816" covered="1"/>
<du var="recLog" def="804" use="819" covered="1"/>
<du var="recLog" def="804" use="853" target="856" covered="0"/>
<du var="recLog" def="804" use="853" target="858" covered="1"/>
<du var="recLog" def="804" use="863" covered="1"/>
<du var="recLog" def="804" use="865" covered="1"/>
<du var="recLog" def="804" use="833" covered="0"/>
<du var="recLog.fc" def="804" use="816" covered="1"/>
<du var="recLog.fc" def="804" use="863" covered="1"/>
<du var="recLog" def="807" use="810" covered="1"/>
<du var="recLog" def="807" use="816" covered="1"/>
<du var="recLog" def="807" use="819" covered="1"/>
<du var="recLog" def="807" use="853" target="856" covered="1"/>
<du var="recLog" def="807" use="853" target="858" covered="0"/>
<du var="recLog" def="807" use="863" covered="1"/>
<du var="recLog" def="807" use="865" covered="1"/>
<du var="recLog" def="807" use="833" covered="0"/>
<du var="recLog.fc" def="807" use="816" covered="1"/>
<du var="recLog.fc" def="807" use="863" covered="1"/>
<du var="journalVersion" def="810" use="860" covered="1"/>
<du var="journalVersion" def="810" use="830" target="832" covered="0"/>
<du var="journalVersion" def="810" use="830" target="844" covered="0"/>
<du var="lenBuff" def="812" use="818" covered="1"/>
<du var="lenBuff" def="812" use="819" covered="1"/>
<du var="lenBuff" def="812" use="820" target="821" covered="0"/>
<du var="lenBuff" def="812" use="820" target="823" covered="1"/>
<du var="lenBuff" def="812" use="823" covered="1"/>
<du var="lenBuff" def="812" use="824" covered="1"/>
<du var="lenBuff" def="812" use="832" covered="0"/>
<du var="lenBuff" def="812" use="833" covered="0"/>
<du var="lenBuff" def="812" use="834" target="835" covered="0"/>
<du var="lenBuff" def="812" use="834" target="837" covered="0"/>
<du var="lenBuff" def="812" use="837" covered="0"/>
<du var="lenBuff" def="812" use="838" covered="0"/>
<du var="recBuff" def="813" use="848" covered="1"/>
<du var="recBuff" def="813" use="849" target="850" covered="1"/>
<du var="recBuff" def="813" use="849" target="852" covered="1"/>
<du var="recBuff" def="813" use="852" covered="1"/>
<du var="recBuff" def="813" use="853" target="856" covered="0"/>
<du var="recBuff" def="813" use="853" target="858" covered="1"/>
<du var="recBuff" def="813" use="858" covered="1"/>
<du var="recBuff" def="813" use="860" covered="1"/>
<du var="offset" def="816" use="860" covered="1"/>
<du var="len" def="824" use="825" target="826" covered="1"/>
<du var="len" def="824" use="825" target="828" covered="1"/>
<du var="len" def="824" use="829" target="830" covered="0"/>
<du var="len" def="824" use="829" target="848" covered="1"/>
<du var="len" def="824" use="849" target="850" covered="1"/>
<du var="len" def="824" use="849" target="852" covered="1"/>
<du var="len" def="824" use="852" covered="1"/>
<du var="len" def="824" use="853" target="856" covered="1"/>
<du var="len" def="824" use="853" target="858" covered="1"/>

```

```

<du var="len" def="824" use="850" covered="1"/>
<du var="len" def="824" use="830" target="830" covered="0"/>
<du var="len" def="824" use="830" target="844" covered="0"/>
<du var="len" def="824" use="844" covered="0"/>
<du var="len" def="824" use="845" covered="0"/>
<du var="isPaddingRecord" def="828" use="859" target="860" covered="1"/>
<du var="isPaddingRecord" def="828" use="859" target="862" covered="0"/>
<du var="len" def="838" use="839" target="840" covered="0"/>
<du var="len" def="838" use="839" target="842" covered="0"/>
<du var="len" def="838" use="849" target="850" covered="0"/>
<du var="len" def="838" use="849" target="852" covered="0"/>
<du var="len" def="838" use="852" covered="0"/>
<du var="len" def="838" use="853" target="856" covered="0"/>
<du var="len" def="838" use="853" target="858" covered="0"/>
<du var="len" def="838" use="850" covered="0"/>
<du var="isPaddingRecord" def="842" use="859" target="860" covered="0"/>
<du var="isPaddingRecord" def="842" use="859" target="862" covered="0"/>
<du var="recBuff" def="850" use="852" covered="1"/>
<du var="recBuff" def="850" use="853" target="856" covered="1"/>
<du var="recBuff" def="850" use="853" target="858" covered="0"/>
<du var="recBuff" def="850" use="858" covered="0"/>
<du var="recBuff" def="850" use="848" covered="0"/>
<du var="recBuff" def="850" use="849" target="850" covered="0"/>
<du var="recBuff" def="850" use="849" target="852" covered="0"/>
<du var="recBuff" def="850" use="860" covered="0"/>
<counter type="DU" missed="37" covered="60"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>

```

*Figura\_15:* Percentuale e analisi della coverage del metodo handleCallbackException (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
handleCallbackExceptions(Exception[], int)	██████	51%	██████	50%

```

888.     private void handleCallbackExceptions(Exception[] excep, int mode) {
889.         if (excep.length == 0 || (mode & CALLBACK_IGNORE) != 0)
890.             return;
891.
892.         OpenJPAException ce;
893.         if (excep.length == 1) {
894.             // If the exception is already a wrapped exception throw the
895.             // exception instead of wrapping it with a callback exception
896.             if (excep[0] instanceof WrappedException)
897.                 throw (WrappedException)excep[0];
898.             else
899.                 ce = new CallbackException(excep[0]);
900.         } else {
901.             ce = new CallbackException(_loc.get("callback-err"))
902.                 .setNestedThrowables(excep);
903.         }
904.         if ((mode & CALLBACK_ROLLBACK) != 0 && (_flags & FLAG_ACTIVE) != 0) {
905.             ce.setFatal(true);
906.             setRollbackOnlyInternal(ce);
907.         }
908.         if ((mode & CALLBACK_LOG) != 0 && _log.isWarnEnabled())
909.             _log.warn(ce);
910.         if ((mode & CALLBACK_RETHROW) != 0)
911.             throw ce;
912.     }

```

*Figura\_16:* Analisi della coverage del metodo handleCallbackException (Jacoco) post miglioramento

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
handleCallbackExceptions(Exception[], int)	██████	95%	██████	94%

```

888.     private void handleCallbackExceptions(Exception[] excep, int mode) {
889.         if (excep.length == 0 || (mode & CALLBACK_IGNORE) != 0)
890.             return;
891.
892.         OpenJPAException ce;
893.         if (excep.length == 1) {
894.             // If the exception is already a wrapped exception throw the
895.             // exception instead of wrapping it with a callback exception
896.             if (excep[0] instanceof WrappedException)
897.                 throw (WrappedException)excep[0];
898.             else
899.                 ce = new CallbackException(excep[0]);
900.         } else {
901.             ce = new CallbackException(_loc.get("callback-err"))
902.                 .setNestedThrowables(excep);
903.         }
904.         if ((mode & CALLBACK_ROLLBACK) != 0 && (_flags & FLAG_ACTIVE) != 0) {
905.             ce.setFatal(true);
906.             setRollbackOnlyInternal(ce);
907.         }
908.         if ((mode & CALLBACK_LOG) != 0 && _log.isWarnEnabled())
909.             _log.warn(ce);
910.         if ((mode & CALLBACK_RETHROW) != 0)
911.             throw ce;
912.     }

```

*Figura\_17*: Analisi della mutation coverage del metodo handleCallbackException (Pit)

[\[torna su\]](#)

```
888     private void handleCallbackExceptions(Exception[] excepts, int mode) {  
889         3 if (excepts.length == 0 || (mode & CALLBACK_IGNORE) != 0)  
890             return;  
891  
892         OpenJPAException ce;  
893         1 if (excepts.length == 1) {  
894             // If the exception is already a wrapped exception throw the  
895             // exception instead of wrapping it with a callback exception  
896         1 if (excepts[0] instanceof WrappedException)  
897             throw (WrappedException)excepts[0];  
898         else  
899             ce = new CallbackException(excepts[0]);  
900     } else {  
901         ce = new CallbackException(_loc.get("callback-err")).  
902             setNestedThrowables(excepts);  
903     }  
904     4 if ((mode & CALLBACK_ROLLBACK) != 0 && (_flags & FLAG_ACTIVE) != 0) {  
905         ce.setFatal(true);  
906         1 setRollbackOnlyInternal(ce);  
907     }  
908     3 if ((mode & CALLBACK_LOG) != 0 && _log.isWarnEnabled())  
909         1 _log.warn(ce);  
910     2 if ((mode & CALLBACK_RETHROW) != 0)  
911         throw ce;  
912 }
```

*Figura\_18:* Percentuale e analisi della coverage del metodo throwNestedExceptions (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">throwNestedExceptions(List, boolean)</a>		91%		81%
<pre>2662.     private void throwNestedExceptions(List&lt;Exception&gt; excep, boolean datastore) { 2663.         if (excep == null    excep.isEmpty()) 2664.             return; 2665.         if (datastore &amp;&amp; excep.size() == 1) 2666.             throw (RuntimeException) excep.get(0); 2667. 2668.         boolean fatal = false; 2669.         Throwable[] t = excep.toArray(new Throwable[excep.size()]); 2670.         for (Throwable throwable : t) { 2671.             if (throwable instanceof OpenJPAException 2672.                 &amp;&amp; ((OpenJPAException) throwable).isFatal()) 2673.                 fatal = true; 2674.         } 2675.         OpenJPAException err; 2676.         if (datastore) 2677.             err = new StoreException(_loc.get("nested-excep")); 2678.         else 2679.             err = new UserException(_loc.get("nested-excep")); 2680.         throw err.setNestedThrowables(t).setFatal(fatal); 2681.     }</pre>				
<a href="#">throwNestedExceptions(List, boolean)</a>		100%		100%

*Figura\_19:* Percentuale della coverage del metodo throwNestedExceptions (Jacoco) post miglioramento [\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">throwNestedExceptions(List, boolean)</a>		100%		100%

*Figura\_20:* Analisi della mutation coverage del metodo throwNestedExceptions (Pit)

[\[torna su\]](#)

2662	private void throwNestedExceptions(List<Exception> excep, boolean datastore) {			
2663 2	if (excep == null    excep.isEmpty())			
2664	return;			
2665 2	if (datastore && excep.size() == 1)			
2666	throw (RuntimeException) excep.get(0);			
2667				
2668	boolean fatal = false;			
2669	Throwable[] t = excep.toArray(new Throwable[excep.size()]);			
2670	for (Throwable throwable : t) {			
2671 1	if (throwable instanceof OpenJPAException			
2672 1	&& ((OpenJPAException) throwable).isFatal())			
2673	fatal = true;			
2674	}			
2675	OpenJPAException err;			
2676 1	if (datastore)			
2677	err = new StoreException(_loc.get("nested-excep"));			
2678	else			
2679	err = new UserException(_loc.get("nested-excep"));			
2680	throw err.setNestedThrowables(t).setFatal(fatal);			
2681	}			

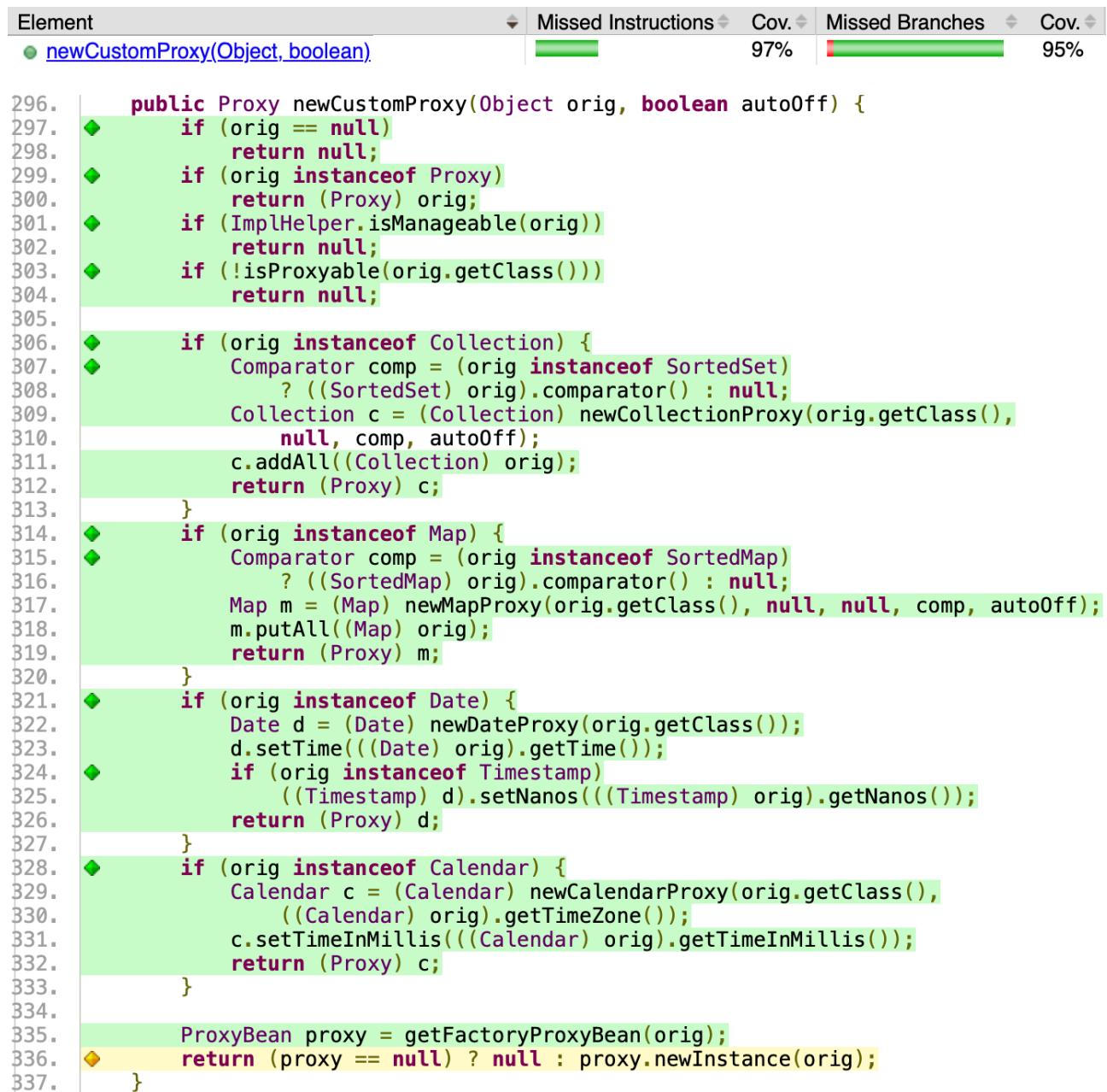
*Figura\_21:* Percentuale e analisi della coverage del metodo newCustomProxy (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
newCustomProxy(Object, boolean)		13%		25%
<pre> 296.     public Proxy newCustomProxy(Object orig, boolean autoOff) { 297.         if (orig == null) 298.             return null; 299.         if (orig instanceof Proxy) 300.             return (Proxy) orig; 301.         if (ImplHelper.isManageable(orig)) 302.             return null; 303.         if (!isProxyable(orig.getClass())) 304.             return null;  305.         if (orig instanceof Collection) { 306.             Comparator comp = (orig instanceof SortedSet) 307.                 ? ((SortedSet) orig).comparator() : null; 308.             Collection c = (Collection) newCollectionProxy(orig.getClass(), 309.                 null, comp, autoOff); 310.             c.addAll((Collection) orig); 311.             return (Proxy) c; 312.         } 313.         if (orig instanceof Map) { 314.             Comparator comp = (orig instanceof SortedMap) 315.                 ? ((SortedMap) orig).comparator() : null; 316.             Map m = (Map) newMapProxy(orig.getClass(), null, null, comp, autoOff); 317.             m.putAll((Map) orig); 318.             return (Proxy) m; 319.         } 320.         if (orig instanceof Date) { 321.             Date d = (Date) newDateProxy(orig.getClass()); 322.             d.setTime(((Date) orig).getTime()); 323.             if (orig instanceof Timestamp) 324.                 ((Timestamp) d).setNanos(((Timestamp) orig).getNanos()); 325.             return (Proxy) d; 326.         } 327.         if (orig instanceof Calendar) { 328.             Calendar c = (Calendar) newCalendarProxy(orig.getClass(), 329.                 ((Calendar) orig).getTimeZone()); 330.             c.setTimeInMillis(((Calendar) orig).getTimeInMillis()); 331.             return (Proxy) c; 332.         } 333.     } 334. 335.     ProxyBean proxy = getFactoryProxyBean(orig); 336.     return (proxy == null) ? null : proxy.newInstance(orig); 337. }</pre>				

*Figura\_22:* Analisi della coverage del metodo newCustomProxy (Jacoco) post miglioramento

[torna su]



*Figura\_23*: Analisi della mutation coverage del metodo newCustomProxy (Pit)

[\[torna su\]](#)

```
296     public Proxy newCustomProxy(Object orig, boolean autoOff) {  
297         1 if (orig == null)  
298             return null;  
299         1 if (orig instanceof Proxy)  
300             1 return (Proxy) orig;  
301         1 if (ImplHelper.isManageable(orig))  
302             return null;  
303         1 if (!isProxyable(orig.getClass()))  
304             return null;  
305  
306         1 if (orig instanceof Collection) {  
307             1 Comparator comp = (orig instanceof SortedSet)  
308                 ? ((SortedSet) orig).comparator() : null;  
309             Collection c = (Collection) newCollectionProxy(orig.getClass(),  
310                 null, comp, autoOff);  
311             c.addAll((Collection) orig);  
312             1 return (Proxy) c;  
313         }  
314         1 if (orig instanceof Map) {  
315             1 Comparator comp = (orig instanceof SortedMap)  
316                 ? ((SortedMap) orig).comparator() : null;  
317             Map m = (Map) newMapProxy(orig.getClass(), null, null, comp, autoOff);  
318             1 m.putAll((Map) orig);  
319             1 return (Proxy) m;  
320         }  
321         1 if (orig instanceof Date) {  
322             Date d = (Date) newDateProxy(orig.getClass());  
323             1 d.setTime(((Date) orig).getTime());  
324             1 if (orig instanceof Timestamp)  
325                 ((Timestamp) d).setNanos(((Timestamp) orig).getNanos());  
326             1 return (Proxy) d;  
327         }  
328         1 if (orig instanceof Calendar) {  
329             Calendar c = (Calendar) newCalendarProxy(orig.getClass(),  
330                 ((Calendar) orig).getTimeZone());  
331             1 c.setTimeInMillis(((Calendar) orig).getTimeInMillis());  
332             1 return (Proxy) c;  
333         }  
334  
335         ProxyBean proxy = getFactoryProxyBean(orig);  
336         2 return (proxy == null) ? null : proxy.newInstance(orig);  
337     }
```

*Figura\_24:* Analisi della mutation coverage del metodo newCustomProxy (Pit) post miglioramento

[\[torna su\]](#)

```
296     public Proxy newCustomProxy(Object orig, boolean autoOff) {  
297         if (orig == null)  
298             return null;  
299         if (orig instanceof Proxy)  
300             return (Proxy) orig;  
301         if (ImplHelper.isManageable(orig))  
302             return null;  
303         if (!isProxyable(orig.getClass()))  
304             return null;  
305  
306         if (orig instanceof Collection) {  
307             Comparator comp = (orig instanceof SortedSet)  
308                 ? ((SortedSet) orig).comparator() : null;  
309             Collection c = (Collection) newCollectionProxy(orig.getClass(),  
310                     null, comp, autoOff);  
311             c.addAll((Collection) orig);  
312             return (Proxy) c;  
313         }  
314         if (orig instanceof Map) {  
315             Comparator comp = (orig instanceof SortedMap)  
316                 ? ((SortedMap) orig).comparator() : null;  
317             Map m = (Map) newMapProxy(orig.getClass(), null, null, comp, autoOff);  
318             m.putAll((Map) orig);  
319             return (Proxy) m;  
320         }  
321         if (orig instanceof Date) {  
322             Date d = (Date) newDateProxy(orig.getClass());  
323             d.setTime(((Date) orig).getTime());  
324             if (orig instanceof Timestamp)  
325                 ((Timestamp) d).setNanos(((Timestamp) orig).getNanos());  
326             return (Proxy) d;  
327         }  
328         if (orig instanceof Calendar) {  
329             Calendar c = (Calendar) newCalendarProxy(orig.getClass(),  
330                     ((Calendar) orig).getTimeZone());  
331             c.setTimeInMillis(((Calendar) orig).getTimeInMillis());  
332             return (Proxy) c;  
333         }  
334  
335         ProxyBean proxy = getFactoryProxyBean(orig);  
336         return (proxy == null) ? null : proxy.newInstance(orig);  
337     }
```

*Figura\_25:* Percentuale della coverage del metodo copyArray (Jacoco)

[\[torna su\]](#)

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">copyArray(Object)</a>	■	100%	■	100%

*Figura\_26:* Analisi della mutation coverage del metodo copyArray (Pit)

[\[torna su\]](#)

```

179     public Object copyArray(Object orig) {
180     1     if (orig == null)
181         return null;
182
183     try {
184         int length = Array.getLength(orig);
185         Object array = Array.newInstance(orig.getClass().
186             getComponentType(), length);
187
188     1     System.arraycopy(orig, 0, array, 0, length);
189     1     return array;
190     } catch (Exception e) {
191         throw new UnsupportedOperationException(_loc.get("bad-array",
192             e.getMessage()), e);
193     }
194 }
```

*Figura\_27:* Analisi della mutation coverage del metodo copyArray (Pit) post miglioramento

[\[torna su\]](#)

```

179     public Object copyArray(Object orig) {
180     1     if (orig == null)
181         return null;
182
183     try {
184         int length = Array.getLength(orig);
185         Object array = Array.newInstance(orig.getClass().
186             getComponentType(), length);
187
188     1     System.arraycopy(orig, 0, array, 0, length);
189     1     return array;
190     } catch (Exception e) {
191         throw new UnsupportedOperationException(_loc.get("bad-array",
192             e.getMessage()), e);
193     }
194 }
```