***Bussolotto Raffaele, 1224718 HOMEWORK 1 – FEBBRAIO 2023***

**TEST SUITE DELLA CLASSE myAdapter.MapAdapter**

**Tutti i test sono stati fatti utilizzando** [***junit@4.13***](mailto:junit@4.13)***.2* e** [***hamcrest-core@1.3***](mailto:hamcrest-core@1.3) **quindi replicare i test con altre versioni di queste librerie potrebbe causare errori indesiderati.**

* **Summary**: prima di iniziare i test è necessario assicurarsi che la main directory del progetto contenga le cartelle *myAdapter*, *myTest* e *matcher* con al suo interno *junit-4.13.2.jar* e *hamcrest-core.1.3.jar*. In caso contrario è sufficiente aprire il progetto su *VS Code*, recarsi nel menù laterale a sinistra sulla voce *testing* e clickare sull’opzione *‘Enable Java Tests’*, successivamente in alto su *JUnit*. Infine, per avviare i test è sufficiente avviare il file bash test.sh che si trova nella main directory del progetto; questo file si occupa di compilare tutti i file .java necessari al funzionamento e ne avvia i test.
* **Test Suite Design**: ogni metodo legato a MapAdapter è testato in base agli oggetti e le eccezioni che dovrebbe restituire.
* **Pre-condition**:prima di eseguire tutti i test è necessario che tutte le interfacce e le classi dei pacchetti myAdapter e myTest siano compilati e matchati correttamente secondo l’ambiente costruito da *junit-4.13.2.jar* e *hamcrest-core.1.3.jar*.
* **Post-condition:** tutti i metodi sono stati analizzati sulle casistiche che potrebbero incontrare in una loro eventuale esecuzione.
* **Test Suite Execution Record**:i risultati dei test sono consultabili sul file *TestCasesExecutionRecord.html*.
* **Execution variables**: nei metodi di test le variabili sono gestite internamente perciò non è necessario passare alcun parametro.
* **Test cases**: per tutti i test case è stato sviluppato un metodo init(), eseguito prima dei test dei metodi e con lo scopo di “popolare” la mappa con valori di test. EntrySet è testato in una classe di test separata da MapAdapter.

**MapAdapter** (conforme a java.util.Map@1.4.2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Summary | Test Description | Pre-Condition | Post-Condition | Expected Results |
| size | metodo che restituisce la dimensione della mappa | inserisco degli elementi nella mappa e controllo che la dimensione vari | *mapAdapter* vuota, metodi *put* e *remove* funzionanti | *mapAdapter* con tre elementi | mi aspetto che dopo aver inserito un elemento la dimensioni incrementi e che decrementi se invece lo rimuovo |
| isEmpty | verifica se la mappa è vuota | inserisco un elemento nella mappa e invoco il metodo | *mapAdapter* vuota, metodo *put* funzionante | *mapAdapter* con un elemento | mi aspetto che la mappa all’inzio del test sia vuota, alla fine invece che non sia più vuota |
| containsKey | verifica che la mappa contenga quella chiave | prima invoco il metodo sulla mappa vuota, poi inserisco un elemento e invoco il metodo su quell’elemento | *mapAdapter* vuota, metodo *put* funzionante | *mapAdapter* con un elemento | mi aspetto che prima il metodo restituisca *false* con la lista vuota, invece che dopo aver inserito l’elemento restituisca *true* |
| containsKeyNull | le chiavi *null* non sono valide | invoco il metodo con una chiave nulla | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| containsValue | verifica che la mappa contenga quel valore | prima invoco il metodo sulla mappa vuota, poi inserisco un elemento e invoco il metodo su quell’elemento | *mapAdapter* vuota, metodo *put* funzionante | *mapAdapter* con un elemento | mi aspetto che prima il metodo restituisca *false* con la lista vuota, invece che dopo aver inserito l’elemento restituisca *true* |
| containsValueNull | i valori *null* non sono validi | invoco il metodo con un valore nullo | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| get | restituzione di un elemento data la chiave | invoco il metodo sulla lista vuota, successivamente inserisco un elemento e invoco di nuovo il metodo | *mapAdapter* vuota, metodo *put* funzionante | *mapAdapter* con un elemento | alla prima invocazione *null* essendo che non si elementi, alla seconda *notNull* |
| getNull | le chiavi *null* non sono valide | invoco il metodo con una chiave nulla | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| put | inserisco un elemento | inserisco degli elementi e man mano controllo che la dimensione della mappa | *mapAdapter* vuota, metodo *size* funzionante | *mapAdapter* con vari elementi | mi aspetto che la dimensioni vari a seconda di come inserisco o rimuovo elementi |
| putNull | la mappa non accetta elementi nulli | invoco il metodo con chiave e valore *null* | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| putNullValue | i valori *null* non sono validi | invoco il metodo con un valore nullo | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| putNullKey | le chiavi *null* non sono valide | invoco il metodo con una chiave nulla | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| remove | rimuove un elemento data la sua chiave | rimuovo gli elementi confrontandoli con quelli che ho inserito precedentemente | *mapAdapter* vuota, metodi *put* e *size* funzionanti | *mapAdapter* vuota | mi aspetto che gli elementi rimossi siano gli stessi con ho inserito con quella chiave e che la dimensione della mappa diminuisca |
| removeNull | le chiavi *null* non sono valide | invoco il metodo con una chiave nulla | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| putAll | inserimento di tutti gli elementi di un’altra mappa | invoco il metodo e verifico tutti gli elementi | *mapAdapter* vuota e *mapAdapterF* con 5 elementi, metodi *size* e *get* funzionanti | *mapAdapter* e *mapAdapterF* con gli stessi elementi | mi aspetto che le dimensioni e gli elementi delle due mappe siano uguali |
| putAllNull | non è possibile invocare il metodo con un parametro *null* perché sarebbe inutilizzabile | invoco il metodo con un parametro *null* | istanza di *mapAdapter* | *mapAdapter* invariata | NullPointerException |
| clear | rimozione di tutti gli elementi dalla mappa | invoco il metodo e verifico la dimensione della mappa | *mapAdapterF* non vuota, metodo *size* funzionante | *mapAdapterF* vuota | verifico che la dimensione della mappa sia 0 dopo l’invocazione del metodo |
| equals | verifica che due mappe siano uguali | confronto due mappe vuote | *mapAdapter* vuota, metodi *put*, *putAll* e *remove* funzionanti | *mapAdapter* con elementi all’interno | *equals* |
| confronto due mappe di dimensioni diverse | *not equals* |
| confronto di due mappe uguali | *equals* |
| hashCode | due oggetti con gli stessi elementi devono avere stessi hashCode | confronto tra due mappe uguali | listAdapter2 e listAdapter3 uguali, metodi *put*, *putAll* e *remove* funzionanti | listAdapter2 e listAdapter3 diverse | equals |
| confronto tra due mappe diverse | not equals |

**EntrySet** (conforme a java.util.Set@1.4.2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Summary | Test Description | Pre-Condition | Post-Condition | Expected Results |
| add | metodo non supportato | invoco il metodo | istanza di *entrySet* | *entrySet* invariata | UnsupportedOperationException |
| addAll | metodo non supportato | invoco il metodo | istanza di *entrySet* | *entrySet* invariata | UnsupportedOperationException |
| addParent | verifico che il set sia backed alla mappa | aggiungo un elemento alla mappa e verifico che vari anche l’entrySet | *mapAdapterF* con entry diverse da (kk,ee), metodo *size* funzionante | *mapAdapterF* con nuova entry (kk,ee) | mi aspetto che l’elemento aggiunto alla lista sia aggiunto anche a *entrySetF* |
| clear | svuotamento del Set | invoco semplicemente il metodo su una mappa non vutoa | *mapAdapterF* e *entrySetF* non vuoti | *mapAdapterF* e *entrySetF* vuoti | mi aspetto che si svuotino sia *entrySetF* e la mappa a esso collegata |
| contains | se il Set contiene quella determinata entry | inserisco una chiave nella lista e faccio alcune verifiche con diverse entry | *entrySet* vuota, metodo *put* funzionante | *entrySet* con un elemento | mi aspetto che restituisca *true* solo una volta e con la entry che ho appena inserito |
| containsClass | *contains* con un oggetto di un’altra classe | invoco il metodo con una stringa come parametro: non è un oggetto valido | *entrySet* vuota, metodo *put* funzionante | *entrySet* invariata | ClassCastException |
| containsNull | *contains* con un oggetto *null* | invoco il metodo con un null come parametro: non è un oggetto valido | *entrySet* vuota | *entrySet* invariata | NullPointerException |
| containsAll | verifico che il Set contenga tutti gli elementi presenti nel Set passato come argomento | aggiungo altri elementi a *mapAdapterF2* oltre a quelli che hanno in comune | *mapAdapterF* e *mapAdapterF2* vuote con gli stessi elementi, metodo *put* funzionante | *mapAdapterF2* con altri elementi nuovie *mapAdapterF* con un solo elemento nuovo | true |
| aggiungo un altro elemento (non presente in *mapAdapterF2*) solo a *mapAdapterF* | false |
| containsAllClass | *contains* con di una lista che non contiene HMap.Entry | invoco il metodo con una Collection con una stringa come elemento (non è un oggetto valido) | *entrySet* vuota | *entrySet* invariata | ClassCastException |
| containsAllNull | *contains* con un oggetto *null* | invoco il metodo con un null come parametro: non è un oggetto valido | *entrySet* vuota | *entrySet* invariata | NullPointerException |
| isEmpty | verifico se il Set contiene elementi | all’inizio ricevo un Set vuoto quindi il risultato deve essere *true*, successivamente aggiungo un elemento | *mapAdapter* vuota, metodo *put* funzionante | *mapAdapter* non vuota | mi aspetto che il metodo restituisca *false* solo dopo aver inserito un elemento |
| iterator | verifico che ritorni effettivamente un iterator e che i metodi base funzionino | l’iterator non deve essere nullo, essendo che il set ha degli elementi deve essere vero *hasNext* e *next* deve restituire un elemento, rimuovendo l’elemento a cui sono la lunghezza del Set devi diminuire di 1 | *entrySetF* non vuota, metodo *size* funzionante | *entrySetF* con un elemento rimosso | *hasNext==true*  *next* not null  *size==prevSize-1* |
| iteratorIll | condizione che si possa invocare *remove* | il *remove* dell’iteratore può essere invocato solo dopo aver chiamato il metodo *next* quindi, se l’iteratore è appena stato creato, deve lanciare un’eccezione | *entrySetF* non vuota | *entrySetF* invariata | IllegalStateException |
| remove | rimozione di un determinato elemento dal Set | rimuovo un elemento che ho appena aggiunto al Set | *entrySet* vuoto, metodo isEmptyfunzionante | *entrySet* invariata | *remove==true*  *isEmpty==true* |
| removeNull | rimozione di un elemento nullo | essendo che la classe non accetta elementi nulli mi aspetto che il metodo lanci eccezione | Istanza di *entrySet* vuoto | *entrySet* invariato | NullPointerException |
| removeClass | rimozione di un oggetto di un’altra classe | invoco il metodo con una stringa come parametro: non è un oggetto valido | *entrySet* vuota | *entrySet* invariata | ClassCastException |
| removeAll | rimozione di tutti gli elementi che sono contenuti nel Set passato come argomento | rimuovo tutti gli elementi di un Set uguale | *entrySetF* e *entrySetF2* con gli stessi elementi, metodo *isEmpty* funzionante | *entrySetF* vuoto | mi aspetto che dopo l’invocazione il metodo abbia rimosso tutti gli elementi |
| removeAllNull | un oggetto Set *null* è inutilizzabile | invoco il metodo con *null* come parametro | Istanza di *entrySet* | *entrySet* invariato | NullPointerException |
| retainAll | rimozione di tutti gli elementi che non sono contenuti nel Set passato come argomento | controllo la presenza o meno di tutti gli elementi che c’erano nella lista precedentemente alla chiamata del metodo | *entrySetF* e *entrySetF2* con una sola entry (k,e), metodo *put* funzionante | *entrySetF* che abbia solo elementi in comune con *entrySetF2* | mi aspetto che il metodo *contains* invocato con l’elemento di *entrySetF2*restituisca tutti *true*, invece sugli altri elementi tutti *false* |
| retainAllNull | un oggetto Set *null* è inutilizzabile | invoco il metodo con *null* come parametro | Istanza di *entrySet* | *entrySet* invariato | NullPointerException |
| toArray | creazione di un array dall’EntrySet | ricavo l’array dal Set e verifico che tutti gli elementi dell’array sia presenti nel Set | istanza di *entrySet*, metodi *contains* e *size* funzionanti | *entrySet* non vuoto | mi aspetto che tutte le invocazioni a *contains* siano *true* e che l’array abbia la stessa lunghezza del Set |
| toArrayParam | creazione di un array dal Set con riferimento a un altro array | passo come parametro al metodo un array di dimensione uguale a quella del Ser, quindi gli elementi devono essere inseriti in quell’array | *entrySet* vuoto, metodi *put* e *size* funzionanti | *entrySet* invariato | *true* |
| passo un array di dimensione maggiore e quindi deve essere rimpiazzato | false |
| equals | verifica che due Set contengano gli stessi elementi | verifico che i due Set siano uguali | *entrySetF* e *entrySetF2* uguali, metodo *clear* funzionante | *entrySetF* e *entrySetF2* diversi tra loro | *true* |
| svuoto *entrySetF* e invoco il metodo | *false* |
| altre verifiche con liste *null*, di istanze di oggetti diversi | *false* |
| hashCode | due oggetti con gli stessi elementi devono avere stessi hashCode | confronto tra due Set uguali | *entrySetF* e *entrySetF2* uguali, metodo *put* funzionante | *entrySetF* e *entrySetF2* diversi tra loro | *equals* |
| confronto tra due Set diversi | *notEquals* |