Documentazione test

1.Test di SetAdapter e SetAdapterIterator

Nome del test	testSize
Test del metodo	Size
Descrizione del test	Si controlla che l'oggetto appena inizializzato abbia dimensione uguale a 0 e dopo aver inserito un elemento essa venga incrementata di uno
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare la dimensione dell'oggetto

Nome del test	testIsEmpty
Test del metodo	isEmpty
Descrizione del test	Si controlla che l'oggetto appena inizializzato sia vuoto e dopo aver aggiunto un elemento che non lo sia più.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se l'oggetto è vuoto, false altrimenti

Nome del test	testContains
Test del metodo	Contains
Descrizione del test	Si controlla che inizialmente SetAdapter non contenga un elemento "o" e dopo averlo inserito che lo contenga. Se provo a controllare che contenga l'oggetto "null" viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se l'elemento è presente, false altrimenti

Nome del test	testToArray
Test del metodo	toArray
Descrizione del test	Si controlla che inserendo tre elementi e richiamando il metodo "toArray", venga creato un array con gli elementi che erano stati aggiunti.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.

Post-condizioni	Il metodo deve ritornare un array con tutti gli elementi dell’oggetto

Nome del test	testAdd
Test del metodo	Add
Descrizione del test	Si controlla che inizialmente l'oggetto SetAdapter non contenga un elemento “o” e dopo averlo inserito che lo contenga e che la dimensione sia uguale a 1. Inoltre se inserisco un altro elemento la dimensione è uguale a 2 e se provo ad inserire un oggetto che è già presente il metodo dia correttamente false in quanto il SetAdapter garantisce l’unicità degli elementi. Infine si controlla che il SetAdapter non sia vuoto. Il metodo add inserisce all.inizio del Set. Se provo ad inserire un elemento null viene lanciata l’eccezione NullPointerException
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se inserisce l’elemento, false se si prova a inserire un elemento già contenuto.

Nome del test	testRemove
Test del metodo	Remove
Descrizione del test	Si controlla che dopo aver aggiunto un elemento “o” ed averlo eliminato, il SetAdapter non lo contenga più e la dimensione sia quindi uguale a 0. Se si prova ad eliminare un elemento che non è contenuto il metodo ritorna giustamente false. Se si prova ad eliminare un elemento null viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina un oggetto, false altrimenti

Nome del test	testContainsAll
Test del metodo	containsAll
Descrizione del test	Si controlla che aggiungendo due elementi a SetAdapter e creando una Collection con gli stessi elementi, il SetAdapter contenga tutti gli elementi della Collection. Dopo aver aggiunto un ulteriore elemento alla Collection se si riprova a fare il controllo risulterà false. Se la collection è uguale a null, viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se contiene tutti gli elementi della Collection, false altrimenti

Nome del test	testAddAll

Test del metodo	AddAll
Descrizione del test	Si controlla che creando una collection con due elementi e usando il metodo addAll vengano aggiunti al SetAdapter tutti gli elementi contenuti dalla collection. Quindi si controlla che SetAdapter contenga gli elementi appena aggiunti e che la dimensione sia uguale a 2. Infine se si prova a inserire nuovamente gli elementi della collection ma SetAdapter già gli contiene tutti il metodo ritorna false. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se aggiunge almeno un elemento della collection, false altrimenti

Nome del test	testRetainAll
Test del metodo	retainAll
Descrizione del test	Si controlla che inserendo due elementi in una collection e gli stessi elementi più un terzo nel SetAdapter, usando il metodo retainAll il SetAdapter contenga infine solo gli elementi contenuti anche dalla Collection, mentre gli elementi non contenuti anche dalla collection vengano eliminati. Quindi la dimensione finale deve essere uguale a 2. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina almeno un elemento di SetAdapter, altrimenti false

Nome del test	testRemoveAll
Test del metodo	removeAll
Descrizione del test	Si controlla che inserendo due elementi in una collection e gli stessi elementi più un terzo nel SetAdapter, usando il metodo removeAll il SetAdapter contenga infine solo gli elementi non contenuti anche dalla Collection, mentre gli elementi contenuti anche dalla collection vengano eliminati. Quindi la dimensione finale deve essere uguale a 1. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina almeno un elemento di SetAdapter, altrimenti false

Nome del test	testClear
Test del metodo	Clear

Descrizione del test	Si controlla che dopo aver inserito due elementi nel setadapter e usato il metodo clear, l'oggetto sia vuoto, quindi la dimensione sia uguale a 0.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se l'oggetto è vuoto, false altrimenti.

Nome del test	testEquals
Test del metodo	Equals
Descrizione del test	Si controlla che aggiungendo due elementi al SetAdapter e successivamente creando un altro oggetti di tipo Setadapter e inserendo gli stessi elementi del primo, essi siano uguali. Se si aggiunge un terzo elemento ad uno dei due il metodo ritornerà quindi false.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se i due oggetti di tipo SetAdapter contengono gli stessi elementi, false altrimenti

Nome del test	testHashCode
Test del metodo	hashCode
Descrizione del test	Si controlla che due oggetti di tipo SetAdapter con gli stessi elementi abbiano lo sesso hashcode, quindi se su uno dei due aggiungo un elemento che non è contenuto anche nell'altro il metodo ritorna false
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se due oggetti di tipo SetAdapter hanno lo stesso hashcode, false altrimenti

Nome del test	testHasNextIterator
Test del metodo	HasNext della classe SetAdapterIterator
Descrizione del test	Si controlla che inserendo tre elementi nel SetAdapter , richiamando il metodo iterator che crea un iteratore del SetAdapter e usando il metodo next alternato a hasNext si controlla se c'è un elemento successivo. Quindi alla terza chiamata i metodo deve ritornare false perché non è presente un elemento successivo.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se è presente un elemento successivo, false altrimenti.

Nome del test	testNextIterator

Test del metodo	Next della classe SetAdapterIterator
Descrizione del test	Si controlla che inserendo tre elementi nel SetAdapter , richiamando il metodo iterator che crea un iteratore del SetAdapter e usando il metodo next vengano restituiti uno alla volta i tre oggetti del SetAdapter.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna l'oggetto successivo se presente

Nome del test	testRemovetIterator
Test del metodo	Remove della classe SetAdapterIterator
Descrizione del test	Si controlla che inserendo tre elementi nel SetAdapter , richiamando il metodo iterator che crea un iteratore del SetAdapter, usando il metodo next e successivamente il metodo remove venga eliminato l'oggetto individuato da next e quindi la dimensione venga diminuita a 2. Se si ripete il next-remove la dimensione sarà 1. Se si chiama due volte il remove di seguito viene lanciata l'eccezione IllegalStateException.
Pre-condizioni	Un oggetto di tipo SetAdapter correttamente inizializzato.
Post-condizioni	Il metodo elimina l'oggetto individuato da next

2. Test di ListAdapter e ListAdapterListIterator

Nome del test	testSize
Test del metodo	size
Descrizione del test	Si controlla che l'oggetto appena inizializzato abbia dimensione uguale a 0 e dopo aver inserito un elemento essa venga incrementata di uno
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare la dimensione dell'oggetto

Nome del test	testIsEmpty
Test del metodo	isEmpty
Descrizione del test	Si controlla che l'oggetto appena inizializzato sia vuoto e dopo aver aggiunto un elemento che non lo sia più.

Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se l’oggetto è vuoto, false altrimenti

Nome del test	testContains
Test del metodo	Contains
Descrizione del test	Si controlla che inizialmente ListAdapter non contenga un elemento “o” e dopo averlo inserito che lo contenga. Se provo a controllare che contenga l’oggetto “null” viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se l’elemento è presente, false altrimenti

Nome del test	testToArray
Test del metodo	toArray
Descrizione del test	Si controlla che inserendo tre elementi e richiamando il metodo “toArray”, venga creato un array con gli elementi che erano stati aggiunti.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare un array con tutti gli elementi dell’oggetto

Nome del test	testAdd1
Test del metodo	Add con un parametro (Object)
Descrizione del test	Si controlla che inizialmente l'oggetto ListAdapter non contenga un elemento “o” e dopo averlo inserito che lo contenga e che la dimensione sia uguale a 1. Inoltre se inserisco un altro elemento la dimensione è uguale a 2. Come ultimo elemento inserisco nuovamente l’elemento “o” e quindi la dimensione diventa 3. Infine si controlla che il ListAdapter non sia vuoto. Gli oggetti vengono inseriti alla fine della lista. Se provo ad inserire un elemento null viene lanciata l’eccezione NullPointerException
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se inserisce l’elemento

Nome del test	testRemove
Test del metodo	remove con un parametro (Object)

Descrizione del test	Si controlla che dopo aver aggiunto due volte un elemento “o” ed averlo usato il metodo remove venga eliminata la prima occorrenza di “o” e quindi che la dimensione sia 1. Infine se si invoca ancora il remove su “o” il ListAdapter non lo contiene più e la dimensione è quindi uguale a 0. Se si prova ad eliminare un elemento che non è contenuto il metodo ritorna giustamente false. Se si prova ad eliminare un elemento null viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina un oggetto, false altrimenti

Nome del test	testContainsAll
Test del metodo	ContainsAll
Descrizione del test	Si controlla che aggiungendo due elementi a listAdapter e creando una Collection con gli stessi elementi, il ListAdapter contenga tutti gli elementi della Collection. Dopo aver aggiunto un ulteriore elemento alla Collection se si riprova a fare il controllo risulterà false. Se la collection è uguale a null, viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se contiene tutti gli elementi della Collection, false altrimenti

Nome del test	testAddAll1
Test del metodo	AddAll con un parametro (Collection)
Descrizione del test	Si controlla che creando una collection con due elementi e usando il metodo addAll vengano aggiunti al ListAdapter tutti gli elementi contenuti dalla collection. Quindi si controlla che SetAdapter contenga gli elementi appena aggiunti e che la dimensione sia uguale a 2. Infine si usa nuovamente il metodo addAll che inserisce i due elementi della Collection in coda alla lista facendo diventare la dimensione uguale a 4. Se la collection è uguale a null, viene lanciata l’eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se aggiunge almeno un elemento della collection, false se la collection è vuota

Nome del test	testAddAll2
Test del metodo	AddAll con due parametri (index, Collection)

Descrizione del test	Si controlla che creando una collection con due elementi e aggiungendo due elementi al ListAdapter, usando il metodo AddAll(1, collection) si inseriscono i due elelementi della collection nella lista a partire dall'indice 1. Quindi la dimensione diventerà 4 e i due elementi della collection si troveranno in mezzo a quelli che erano presenti inizialmente nel ListAdapter. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException, mentre se index è maggiore della dimensione o minore di 0 viene lanciata IndexOutOfBoundsException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se aggiunge almeno un elemento della collection, false se la collection è vuota

Nome del test	testRemoveAll
Test del metodo	removeAll
Descrizione del test	Si controlla che inserendo due elementi in una collection e gli stessi elementi più un terzo nel listAdapter, usando il metodo removeAll il ListAdapter contenga infine solo gli elementi non contenuti anche dalla Collection, mentre gli elementi contenuti anche dalla collection vengano eliminati. Quindi la dimensione finale deve essere uguale a 1. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina almeno un elemento di ListAdapter, altrimenti false

Nome del test	testRetainAll
Test del metodo	RetainAll
Descrizione del test	Si controlla che inserendo due elementi in una collection e gli stessi elementi più un terzo nel ListAdapter, usando il metodo retainAll il ListAdapter contenga infine solo gli elementi contenuti anche dalla Collection, mentre gli elementi non contenuti anche dalla collection vengano eliminati. Quindi la dimensione finale deve essere uguale a 2. Se la collection è uguale a null, viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se elimina almeno un elemento di ListAdapter, altrimenti false

Nome del test	testClear
Test del metodo	Clear

Descrizione del test	Si controlla che dopo aver inserito due elementi nel ListAdapter e usato il metodo clear, l'oggetto sia vuoto, quindi la dimensione sia uguale a 0.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se l'oggetto è vuoto, false altrimenti.

Nome del test	testEquals
Test del metodo	Equals
Descrizione del test	Si controlla che aggiungendo due elementi al ListAdapter e successivamente creando un altro oggetti di tipo ListAdapter e inserendo gli stessi elementi del primo, essi siano uguali. Se si aggiunge un terzo elemento ad uno dei due il metodo ritornerà quindi false.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se i due oggetti di tipo ListAdapter contengono gli stessi elementi, false altrimenti

Nome del test	testHashCode
Test del metodo	HashCode
Descrizione del test	Si controlla che due oggetti di tipo ListAdapter con gli stessi elementi abbiano lo sesso hashcode, quindi se su uno dei due aggiungo un elemento che non è contenuto anche nell'altro il metodo ritorna false
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se due oggetti di tipo ListAdapter hanno lo stesso hashcode, false altrimenti

Nome del test	testGet
Test del metodo	Get
Descrizione del test	Si controlla che inserendo 4 elementi (si inserisce sempre in fondo alla lista) a ListAdapter, usando get(0) restituisce il primo elemento inserito mentre usando get(2) restituisce il terzo elemento inserito. Se si inserisce un indice maggiore o uguale alla dimensione il metodo lancia IndexOutOfBoundsException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	L'elemento corrispondente all'indice

Nome del test	testSet
Test del metodo	Set

Descrizione del test	Si controlla che inserendo tre elementi in ListAdapter e applicando il set sulle tre posizioni si possono cambiare i valori degli elementi precedentemente inseriti. Se l'indice è maggiore della dimensione il metodo lancia l'eccezione IndexOutOfBoundsException, mentre se si cerca si inserire un valore nullo il metodo lancia l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	L'elemento precedente alla posizione specificata

Nome del test	testAdd2
Test del metodo	Add con due parametri (index, Object)
Descrizione del test	Si controlla che inserendo un elemento alla posizione 0 e tre elementi alla posizione 1, la dimensione sia 4 e il primo elemento della lista sia quello inserito alla posizione zero e dopo in successione gli altri tre elementi partendo dall'ultimo inserito alla posizione 1. Se si prova ad inserire null il metodo lancia NullPointerException, mentre se provo ad inserire in un indice maggiore della dimensione il metodo lancia l'eccezione IndexOutOfBoundsException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Elemento inserito alla posizione specificata e dimensione aumentata di uno

Nome del test	testRemove2
Test del metodo	Remove con un parametro (index)
Descrizione del test	Si controlla che inserendo tre elementi alla lista ed eliminando quello in posizione 1 la dimensione sia 2 e che non contenga più l'elemento inserito per secondo. Se si prova ad eliminare un elemento in un indice maggiore della dimensione il metodo lancia l'eccezione IndexOutOfBoundsException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Elemento precedentemente presente in quella posizione

Nome del test	testIndexOf
Test del metodo	IndexOf
Descrizione del test	Si controlla che inserendo 4 elementi di cui due uguali(uno in pozione 1 e uno in posizione 3) se si usa il metodo indexOf dell'elemento che è presente due volte essa ritornerà 1. Mentre se si usa indexOf di un elemento non presente ritorna -1. Se si prova a cercare l'indice di null viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	La posizione della prima occorrenza dell'elemento specificato

Nome del test	testLastIndexOf
Test del metodo	LasIndexOf
Descrizione del test	Si controlla che inserendo 4 elementi di cui due uguali(uno in pozione 1 e uno in posizione 3) se si usa il metodo lastIndexOf dell'elemento che è presente due volte essa ritornerà 3. Mentre se si usa lasIndexOf di un elemento non presente ritorna -1. Se si prova a cercare l'indice di null viene lanciata l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	La posizione dell'ultima occorrenza dell'elemento specificato

Nome del test	testSubList
Test del metodo	subList
Descrizione del test	Aggiungo alla lista 7 elementi e uso il metodo subList(1, 5) che mi crea una sottolista della lista contenente gli elementi dall'indice 1 (incluso) al 5 (escluso). Quindi controllo che la dimensione della subList sia uguale a 4 e che effettivamente i valori corrispondano. Successivamente aggiungo un elemento alla fine della sublist quindi la dimensione di essa diventa 5 e controllo che effettivamente la sublist non sia vuota. Dopo elimino un certo valore della sublist e controllo che la dimensione sia decrementata di uno e che la sublist nn contenga più quell'elemento eliminato. Infine setto l'elemento all'indice 0 della sublist ad un certo valore e controllo che il get(0) della sublist restituisca effettivamente il valore appena settato. Creo quindi un iteratore della sublist e scorro tutti e 4 gli elementi controllando l'effettivo valore e accertando della presenza di un successivo con hasNext. Ricreo un iteratore della sublist e uso next-remove per eliminare il primo elemento della sublist facendo diventare la dimensione della sublist 3. Alla fine controllo che effettivamente la sublist sia backed quindi se aggiungo un elemento alla fine della sublist anche la List lo conterrà (in quanto la List e la sublist condividono lo stesso Vector, anche se gli indici non vengono aggiornati).
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Creazione di un oggetto di tipo subList

Nome del test	testHasNextListIterator
Test del metodo	HasNext della classe ListAdapterListIterator
Descrizione del test	Si controlla che inserendo tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter e usando il metodo next alternato a hasNext si controlla se c'è un elemento successivo. Quindi alla terza chiamata il metodo deve ritornare false perché non è presente un elemento successivo.

Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se è presente un elemento successivo, false altrimenti.

Nome del test	testNextListIterator
Test del metodo	Next della classe ListAdapterListIterator
Descrizione del test	Si controlla che inserendo tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter e usando il metodo next vengano restituiti uno alla volta i tre oggetti del listAdapter.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna l'oggetto successivo se presente

Nome del test	testHasPrevoiusListIterator
Test del metodo	HasPrevious della classe ListAdapterListIterator
Descrizione del test	Si controlla che inserendo tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter e usando il metodo next tre volte per raggiungere l'ultimo elemento, e usando il metodo previous alternato a hasPrevious si controlla se c'è un elemento precedente. Quindi alla terza chiamata il metodo deve ritornare false perché non è presente un elemento precedente.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se è presente un elemento precedente, false altrimenti.

Nome del test	testPreviousListIterator
Test del metodo	Previous della classe ListAdapterListIterator
Descrizione del test	Si controlla che inserendo tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter e usando il metodo next tre volte per raggiungere l'ultimo elemento, e usando il metodo previous vengano restituiti uno alla volta i tre oggetti del listAdapter, partendo dall'ultimo.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna l'oggetto precedente se presente

Nome del test	testRemoveListIterator
Test del metodo	Remove della classe ListIterator

Descrizione del test	Si controlla che inserendo tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter, usando il metodo next e successivamente il metodo remove venga eliminato l'oggetto individuato da next e quindi la dimensione venga diminuita a 2. Se si ripete il next-remove la dimensione sarà 1. Se si chiama due volte il remove di seguito viene lanciata l'eccezione IllegalStateException.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Il metodo elimina l'oggetto individuato da next o previous

Nome del test	testSetlistIterator
Test del metodo	Set della classe ListIterator
Descrizione del test	Inserisco tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter e usando il metodo next due volte raggiungo il secondo elemento e uso il set per cambiargli valore, successivamente uso il metodo previous per raggiungere il primo elemento e uso il set per cambiargli valore. Alla fine controllore siano contenuti i due valori cambiati dal set e il terzo elemento è rimasto invariato.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	L'elemento cambia valore

Nome del test	testAddListIterator
Test del metodo	Add della classe ListIterator
Descrizione del test	Inserisco tre elementi nel ListAdapter , richiamando il metodo iterator che crea un iteratore del ListAdapter, usando il metodo next e successivamente l'ade aggiungo in seconda pozione un nuovo elemento, quindi la dimensione diventa 4.
Pre-condizioni	Un oggetto di tipo ListAdapter correttamente inizializzato.
Post-condizioni	Aggiunta di un elemento

3. Test di MapAdapter

Nome del test	testSize
Test del metodo	Size

Descrizione del test	Si controlla che l'oggetto appena inizializzato abbia dimensione uguale a 0 e dopo aver inserito una coppia(key-value) di valori essa venga incrementata di uno
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare la dimensione dell'oggetto

Nome del test	testIsEmpty
Test del metodo	isEmpty
Descrizione del test	Si controlla che l'oggetto appena inizializzato sia vuoto e dopo aver aggiunto una coppia(key-value) di valori che non lo sia più.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il metodo deve ritornare true se l'oggetto è vuoto, false altrimenti

Nome del test	testContainsKey
Test del metodo	containsKey
Descrizione del test	Si controlla che l'oggetto appena inizializzato non contenga una chiave e dopo aver inserito tre coppie di valori contenga le tre chiavi inserite. Se si prova a cercare la presenza di una chiave null il metodo lancia l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se contiene la chiave, false altrimenti

Nome del test	testContainsValue
Test del metodo	Containsvalue
Descrizione del test	Si controlla che l'oggetto appena inizializzato non contenga un valore e dopo aver inserito tre coppie di valori contenga i tre valori inseriti. Se si prova a cercare la presenza di una valore null il metodo lancia l'eccezione NullPointerException.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il metodo ritorna true se contiene il valore, false altrimenti

Nome del test	testGet
Test del metodo	Get
Descrizione del test	Si controlla che dopo aver inserito tre coppie di valori, con il get inserendo le tre chiavi devono corrispondere i relativi valori. Se si cerca di fare get di una chiave null il metodo lancia l'eccezione NullPointerException.

Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il valore relativo alla chiave

Nome del test	testPut
Test del metodo	Put
Descrizione del test	Si controlla che dopo aver inserito tre coppie di valori, la size sia uguale a tre e che i metodo isEmpty ritorni false
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Inserimento di una coppia di valori(key-value)

Nome del test	testRemove
Test del metodo	Remove
Descrizione del test	SI controlla che dopo aver inserito tre coppie di valori e averne rimossa una identificandola con la chiave, la dimensione del MapAdapter sia uguale a 2 e che non contenga più il valore e la chiave rimossi. Successivamente, dopo aver eliminato i due restanti valori si controlla che il MapAdapter sia vuoto.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il valore della coppia appena rimossa

Nome del test	testPutAll
Test del metodo	puttAll
Descrizione del test	Dopo aver creato un nuovo MapAdapter e averci inserite tre coppie di valori, uso il metodo putAll e controllo che il Map Adapter istanziato nel @Before abbia dimensione uguale a 3. Se il MapAdapter appena creato è uguale a null il metodo lancia NullPointerException.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Inserimento dei valori di un altro MapAdapter

Nome del test	testClear
Test del metodo	Clear
Descrizione del test	Si controlla che dopo aver inserito tre coppie di valori e usato il metodo clear il MapAdapter abbia dimensione uguale a 0.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	MapAdapter vuoto

Nome del test	testEquals
Test del metodo	Equals
Descrizione del test	Dopo aver inserito tre coppie di valori ed aver creato un nuovo MapAdapter e averci inserite le stesse tre coppie di valori si controlla che i due MapAdapter siano uguali.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	True se sono uguali, false altrimenti

Nome del test	testHashCode
Test del metodo	HashCode
Descrizione del test	Dopo aver inserito tre coppie di valori ed aver creato un nuovo MapAdapter e averci inserite le stesse tre coppie di valori controllo che i due MapAdapter abbiano lo stesso hashCode.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	True se hanno lo stesso hashCode, false altrimenti.

Nome del test	testGetKeyEntry
Test del metodo	GetKey della classe Entry
Descrizione del test	Dopo aver creato un oggetto di tipo MapAdapter.Entry si controlla che la chiave restituita sia giusta
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	La chiave dell’entry

Nome del test	testGetValueEntry
Test del metodo	GetValue della classe Entry
Descrizione del test	Dopo aver creato un oggetto di tipo MapAdapter.Entry si controlla che il valore restituito sia giusto
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il valore dell’entry

Nome del test	testSetValueEntry
Test del metodo	SetValue della classe Entry

Descrizione del test	Dopo aver creato un oggetto di tipo MapAdapter.Entry e usato il metodo setvalue imposto un nuovo valore a value identificato tramite la chiave e si controlla che effettivamente il valore sia cambiato
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Il vecchio value

Nome del test	testEqualsEntry
Test del metodo	Equals della classe Entry
Descrizione del test	Dopo aver creato due oggetti uguali di tipo MapAdapter.Entry si controlla che abbiano lo stesso hashcode
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	True se sono uguali, false altrimenti

Nome del test	testHashCodeEntry
Test del metodo	hashCode della classe Entry
Descrizione del test	Dopo aver creato due oggetti uguali di tipo MapAdapter.Entry si controlla che abbiano lo stesso hashcode
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	True se hanno lo stesso hashcode, false altrimenti

Nome del test	testToString
Test del metodo	toString della classe Entry
Descrizione del test	Dopo aver creato un oggetto di tipo MapAdapter.Entry si controlla che mi crei una stringa con “key: value”
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Stringa con “key: value”

Nome del test	testEntrySet
Test del metodo	entrySet

Descrizione del test	Dopo aver inserito quattro coppie di valori(entry) creo un set con il metodo entrySet, e controllo che sia presente una Entry. Si rimuove quindi una entry e controllo che non sia più presente. Successivamente creo un altro set con le stesse tre entry rimaste nell'altro utilizzando il metodo entrySet e controllo che i due set siano uguali e che quindi il primo contenga tutti gli elementi del secondo. Inoltre uso removeAll eliminando tutti gli elementi del primo set e mi assicuro che la dimensione del primo set sia uguale a 0 e anche quindi anche quella di MapAdapter essendo backed. Infine dopo aver inserito nuovamente 4 coppie di valori in MapAdapter creo un nuovo set e ci creo un iteratore. Scorro tutto il set controllando i valori corrispondenti, rimuovo il penultimo elemento e controllo che non sia più contenuto nel nuovo set oltre che la dimensione sia 3.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Un insieme di Entry

Nome del test	testKeySet
Test del metodo	KeySet
Descrizione del test	Dopo aver inserito quattro coppie di valori creo un set delle chiavi di MapAdapter con il metodo keySet, e controllo che sia presente una chiave. Si rimuove quindi una chiave e controllo che non sia più presente e che la dimensione del set sia 3. Successivamente creo un altro set con le stesse tre chiavi rimaste nell'altro utilizzando il metodo keySet e controllo che i due set siano uguali e che quindi il primo contenga tutti gli elementi del secondo. Inoltre uso removeAll eliminando tutti gli elementi del primo set e mi assicuro che la dimensione del primo set sia uguale a 0, quindi sia vuoto, e anche quindi anche quella di MapAdapter essendo backed. Infine dopo aver inserito nuovamente 4 coppie di valori in MapAdapter creo un nuovo set sempre con keyset e ci creo un iteratore. Scorro tutto il set controllando i valori corrispondenti, rimuovo l'ultimo elemento e controllo che non sia più contenuto nel nuovo set oltre che la dimensione sia 3.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Un insieme di chiavi

Nome del test	testValues
Test del metodo	Values

Descrizione del test	Dopo aver inserito quattro coppie di valori creo una collection di valori di MapAdapter con il metodo Values, e controllo che sia presente una valore. Si rimuove quindi una valore e controllo che non sia più presente e che la dimensione della collection sia 3. Successivamente creo un'altra collection con i valori rimasti nell'altra utilizzando il metodo values e controllo che le due collection siano uguali e che quindi la prima contenga tutti gli elementi della seconda. Inoltre uso removeAll eliminando tutti gli elementi della prima collection e mi assicuro che la dimensione della prima collection sia uguale a 0, quindi sia vuota, e anche quindi anche quella di MapAdapter essendo backed. Infine dopo aver inserito nuovamente 4 coppie di valori in MapAdapter creo una nuova collection sempre con value e ci creo un iteratore. Scorro tutta la collection controllando i valori corrispondenti, rimuovo l'ultimo elemento e controllo che non sia più contenuto nella nuova collection oltre che la dimensione sia 3.
Pre-condizioni	Un oggetto di tipo MapAdapter correttamente inizializzato.
Post-condizioni	Una collection dei valori