# Sommario Homework

In questo documento è presente la descrizione degli elementi presenti nell'homework. Quest'ultimo è organizzato secondo la seguente gerarchia:

- **myLib**, dove sono presenti i file di JUnit 4.13 e hamcrest-core-1.3 per lo sviluppo dei test del codice sviluppato per l'assegnamento.
- **myAdapter**, libreria che comprende il codice per la realizzazione di ListAdapter.java e MapAdapter.java oltre a contenere anche le interfacce usate per la realizzazione degli adapter che sono state opportunatamente commentate in stile Javadoc direttamente dalla documentazione di Java 1.4.2.
- **myTest**, dove sono presenti i due tester ListTest.java e MapTest.java, codice che testano approfonditamente i due adattatori realizzati. Sono inoltre presenti il TestRunner.java e il TestSuite.java, che servono per eseguire i test di entrambe le classi adapter insieme e vederne i risultati
- **myDocu**, dove sono presenti le documentazioni dell'Homework in formato PDF.

L'obiettivo dell'homework era quello di **realizzare un Adapter della classe List e Map tramite gli Adaptee di tipo Vector e Hashtable**, dovendo utilizzare una versione di java (CLDC 1.1) in cui non erano ancora presenti le strutture dati da realizzare. Usando quindi **solo** i metodi delle classi Adaptee sono state quindi realizzate le classi ListAdapter e MapAdapter tramite i metodi delle interfacce e in particolare:

- ListAdapter che implementa l'interfaccia HList.java che a sua volta estende HCollection.java. Questa lista può quindi contenere qualsiasi tipo di oggetto ad eccezione di *null* che non è permesso dalla documentazione. La lista usa un Vector (l'adattatore) per tutta la gestione dei dati. In particolare, vengono usati gli Iteratori per navigare all'interno della struttura, avanti o indietro, per aggiungere o togliere, cercare o rimuovere oggetti (*object*), indipendentemente dal contenitore in cui si trova. All'interno di ListAdapter sono presenti **due iteratori**:

1. Un iteratore che implementa l'interfaccia HIterator.java, un iteratore che serve solo ad attraversare la lista in un verso (avanti), e aggiungere o togliere elementi.
2. Un secondo iteratore che implementa l'interfaccia HListIterator.java che permette invece di eseguire le operazioni potendo tornare anche indietro.

Sono state poi create 3 classi interne in ListAdapter, due classi sono gli iteratori sopra descritti e l'ultima è una **classe privata SubList**. Questa sottoclasse privata permette di ritagliarsi una porzione della lista di tipo ListAdapter su cui si può lavorare con gli stessi metodi della lista originale. È anche fornito lo stesso set di iteratori che è presente nella classe madre e quindi posso operare come nella superclasse. Nella SubList è presente il backing, ossia **ogni operazione che viene effettuata nella sottoclasse viene anche effettuata nella lista originale e viceversa**.

- MapAdapter che implementa l'interfaccia HMap.java. Questa mappa può quindi contenere qualsiasi tipo di oggetto ad eccezione di *null* che non è permesso dalla documentazione. La mappa usa un Hashtable (l'adattatore) per tutta la gestione dei dati attraverso la struttura chiave-valore. In particolare, vengono usati gli iteratori per navigare all'interno della struttura, avanti o indietro, per aggiungere o togliere, cercare o rimuovere oggetti (*object*), indipendentemente dal contenitore in cui si trova. All'interno di MapAdapter sono presenti **4 classi interne**:
    1. EntrySet, che implementa HSet (che a sua volta estende HCollection) rappresenta una struttura in grado di contenere dati di tipo Entry, ossia associando ad elementi presenti nella struttura una coppia chiave-valore, che realizza poi i metodi della HSet (che ha la stessa struttura dell'interfaccia HCollection ma **senza la possibilità di avere valori doppi** all'interno della struttura che la realizza).
    2. KeySet, che implementa HSet rappresenta una struttura in grado di contenere elementi di tipo Key, ossia, realizzando i metodi di HSet, opererà solamente tramite le chiavi presenti nella mappa.
    3. ValueCollection, che implementa HCollection (*e non HSet*) rappresenta una struttura in grado di contenere solamente elementi di tipo Value, e nella realizzazione di HSet, userà, quindi

solamente i Value contenuti nella mappa (si ricorda che a differenza delle chiavi, ci possono essere valori uguali associati a chiavi diverse).

4. MyEntry, che implementa HEntry, opera direttamente su un singolo caso di coppia chiave-valore della mappa

Nelle prime 3 sottoclassi è presente un Iteratore che estende HIterator (è quindi un iteratore "semplice") e che opera in ogni sottoclasse in modo coerente con il tipo di dato che ne è contenuto: su Entry, su Key o su Values. È importante specificare che **quando si opera su una sottoclasse che realizza i metodi di HSet o HCollection** non è possibile richiamare i metodi *add()* o *addAll()* in quanto, come specificato dalla documentazione non sono supportati e vengono gestiti tramite il lancio dell'eccezione *UnsupportedOperationException()*. Si ricorda inoltre che nella mappa è presente il **backing**: gli oggetti EntrySet, KeySet e ValueCollection che vengono creati dai metodi modificando i loro attributi modificheranno anche la mappa "madre" e viceversa, quindi nel caso in cui io voglia operare su KeySet sto operando indirettamente anche sugli elementi della mappa che sono stati passati