RELAZIONE PROGETTO SECURE DATA CONTAINER

Superbo Rebecca

Matricola 560504

Il progetto si compone di 10 file .java con le due implementazioni di SecureDataContainer che si basano su due strutture dati differenti: la prima utilizza una TreeMap e la seconda un'HashMap. Ci sono 4 file in comune tra le due implementazioni (UserInterface, User, Exceptions e SecureDataContainer), i due file con implementazione rispettivamente attraverso TreeMap (TreeSecureDataContainer) e HashMap (HashSecureDataContainer) e 4 TestMain.

L'implementazione di UserInterface, realizzata in User, è basata sull'utilizzo di due Linked List: la prima per la gestione degli elementi di tipo E appartenenti al singolo User e la seconda che gestisce l'elenco degli altri utenti autorizzati ad accedere ai dati di User (proprietario).

Di seguito vi è una descrizione delle classi utilizzate:

EXCEPTIONS: in questa classe sono state create delle apposite eccezioni per il programma.

USER: questa classe si occupa della creazione del singolo User e viene utilizzata sia da HashSecureDataContainer che da TreeSecureDataContainer. Ha come variabili d'istanza private un Id, una password ("passw"), una Linked List "DataElements" con elementi di tipo E per la gestione degli elementi posseduti dal singolo utente e una "AuthorizedUsers" con elementi di tipo String per l'elenco di utenti autorizzati dallo User. Il suo metodo costruttore public User inizializza tutte le variabili d'istanza descritte. Al suo interno ci sono i metodi che si occupano di eseguire le operazioni descritte in SecureDataContainer riferite però ad un singolo User anziché a molti (non sono infatti descritti i metodi createUser e RemoveUser, presenti invece nelle due implementazioni di SecureDataContainer). Ciò migliora la leggibilità del codice e permette di debuggare il codice in maniera più semplice poiché sarà necessario riferirsi alla sola classe User. In essa è presente anche una classe privata che estende Iterator<E> per gestire l'iteratore (senza remove, al suo posto sarà infatti lanciata un'eccezione). Una specifica dettagliata dei metodi utilizzati è presente nella UserInterface.

HASHSECUREDATACONTAINER E TREESECUREDATACONTAINER: queste due classi implementano l'interfaccia SecureDataContainer rispettivamente con un'HashMap e una TreeMap**.** L'interfaccia java.util.Map<K,V> specifica il tipo

Map con chiavi di tipo K e valori associati di tipo V, in questo caso K sarà di tipo String e V di tipo User<E>. Entrambe le implementazioni posseggono un metodo costruttore che inizializza rispettivamente, per la collezione di users, una tabella Hash (tramite la classe HashMap) e un albero binario di ricerca (red-black tree, tramite la classe TreeMap, il quale utilizza come comparatore l'ordinamento naturale delle stringhe). All'interno delle due implementazioni vi sono tutti i metodi descritti nell'interfaccia più due metodi aggiuntivi: **private boolean FoundId** che, dato un id di uno user, controlla se esso è già stato inserito nella collezione oppure no, e **public void authorizedUser** che controlla se l'id dell'altro utente è stato autorizzato e quindi inserito nella Linked List del proprietario.

TESTMAIN: queste quattro classi contengono gli stessi due tipi di test, TestMainT e TestMain2T per TreeSecureDataContainer , TestMainH e TestMain2H per HashSecureDataContainer. I primi simulano la corretta esecuzione del programma senza lanciare eccezioni, i secondi invece consentono di vedere come il programma lanci correttamente le eccezioni utilizzate e create. Per far ciò occorre eseguire per ogni comando un'azione di "try e catch".