

Relazione SecureDataContainer

Giovanni Solimeno

1 dicembre 2018

1 Scelte progettuali

In generale, si è deciso di creare una classe contenitrice `Element<E>` per incapsulare il tipo generico, in modo da rendere più facili le operazioni sui permessi, e una classe `User`, rappresentante il singolo utente.

1.1 `Element<E>`

La classe `Element<E>` contiene tre campi privati, con relativi metodi setter/getter:

- Il campo `owner` è una stringa contenente il proprietario del dato. Non è accessibile direttamente, ma è possibile controllare se un particolare utente è proprietario del dato tramite il metodo `ownedBy(who)`.
- Il campo `allowed` è una lista di stringhe, contenente gli utenti autorizzati ad accedere al dato (escluso il proprietario). Non è possibile accedere direttamente alla lista, ma è possibile indicare che un utente deve essere autorizzato/disautorizzato tramite i rispettivi metodi `allowUser(other)` (che lancia `UserAlreadyAllowedException` se l'utente è già autorizzato) e `denyUser(other)` (che lancia `UserNotAllowedException` in caso l'utente non sia presente tra gli utenti autorizzati).
- Il campo `e1` è un riferimento ad una istanza del tipo generico `E`. È possibile accedervi tramite il metodo `getE1()`, che restituisce un riferimento `e1`, mentre non è possibile cambiarne il valore.

Inoltre, è possibile controllare se un utente può accedere a un dato tramite il metodo `canBeAccessedBy(other)`, che restituisce `true` se e solo se `other` è il proprietario oppure è presente nella lista degli utenti autorizzati. La classe sovrascrive il metodo `Object.equals(other)`, in modo da ritornare `true` se e solo se `other.owner.equals(this.owner)` è `true` e `other.getE1().equals(this.getE1())` restituisce `true`. Viene generata l'eccezione unchecked `NullPointerException` se `other` è nullo (la scelta di chiamare `equals` su `other.owner` e su `other.getE1()` è stata fatta in modo da lanciare in automatico l'eccezione se `other` è `null`).

1.2 `User`

La classe `User` contiene due campi privati:

- Il campo `userName` contiene il nome utente dell'utente, ed è possibile accedervi tramite il metodo `getUserName()`. Non è possibile in alcun modo modificarne il valore.
- Il campo `userPass` contiene la password dell'utente, ed è possibile soltanto modificarla, tramite il metodo `setUserPass(newPass)`, mentre non è possibile accedervi in alcun modo.

Inoltre, la classe implementa come meccanismo di login la sovrascrittura del metodo `Object.equals(other)`, che restituisce `true` se e solo se `other.getUserName().equals(this.getUserName)` e `other.userPass.equals(this.userPass)` (si è deciso di effettuare il confronto tramite i metodi/campi di `other` per lo stesso motivo di `Element<E>.equals()`), e implementa l'interfaccia `Comparable<T>`, in modo da ordinare gli utenti in base al nome (proprietà che viene usata nella classe `TreeMapSecureDataContainer`).

2 Scelte specifiche

Si è scelto di non imporre vincoli sulle classi/interfacce del tipo generico di `SecureDataContainer`, in modo da rendere l'utilizzo dell'interfaccia facile e non creare problemi di compatibilità, e di non criptare gli elementi salvati (per lo stesso motivo).

2.1 ListSecureDataContainer

Si è deciso di fornire le due implementazioni usando due metodologie diverse: La prima (`ListSecureDataContainer`) si appoggia su due liste non ordinate, la prima contenente gli utenti registrati, mentre la seconda contenente i dati degli utenti.

2.2 TreeMapSecureDataContainer

La seconda (`TreeMapSecureDataContainer`) si basa su una `TreeMap` (ordinata tramite `User.compareTo()`), che associa ad ogni utente una lista con gli elementi da lui posseduti. Nonostante ciò, la ricerca/rimozione/condivisione di un elemento non si avvale delle proprietà di un albero, in quanto bisogna scorrere ogni associazione utente/lista elementi, alla ricerca degli elementi condivisi (oltre a quelli posseduti, se non ci fosse bisogno di cercare tali elementi basterebbe controllare nella lista associata all'utente che ha richiesto l'operazione), ma permette un login/aggiunta di un utente in modo più efficace.

3 Esecuzione della batteria di test

Compilare il progetto ed eseguire il main presente nella classe `SecureDataContainerTests`. Se il risultato è diverso da 0 (oppure se vengono generati a schermo messaggi contenenti [`TEST FALLITO`]), picchiare il sottoscritto.