Relazione Progetto Java

Anno 2018-2019

Selene Gerali - MATRICOLA: 546091

1 Introduzione:

Il progetto richiede di sviluppare una collezione chiamata *SecureDataContainer*, la quale riesce a memorizzare oggetti di un determinato tipo generico E e garantisce l'accesso ai dati, solo agli utenti registrati alla collezione.

2 Scelte Progettuali:

2.1 STRUTTURE DATI

Ho utilizzato due implementazioni differenti: Vector e HashTable.

- Nella prima implementazione: MyVectorSecureDataContainer ho utilizzato un Vector<*User*> nella quale *User* è una classe che contiene due variabili d'istanza nomeId e password e un Vector <*DataCollection*> la quale *DataCollection* è un ulteriore classe che contiene il dato generico e un Vector<String>, che rappresenta l'insieme degli utenti a cui è stato condiviso quel dato.
- Nella seconda implementazione: MyHashSecureDataContainer ho utilizzato un Vector<*UserReg*> , dove *UserReg* è una classe composta da nomeId e password utilizzata per memorizzare gli utenti registrati e una HashTable composta dal nomeid come chiave univoca il cui valore è rappresentato da un Vector<*DataCollection*>. (DataCollection è la classe descritta precedentemente).

Per quanto riguarda le differenze computazionali delle due diverse implementazioni, utilizzando i Vector impiego tempo lineare nelle ricerche massive, mentre utilizzando una struttura dati dinamica come può essere l'HashTable, attraverso la chiave accedo al vettore di dati con accesso diretto.

2.2 IMPLEMENTAZIONE

La specifica richiede che un Utente deve essere registrato alla collezione (metodo **createUser** aggiunge un utente) e identificato tramite username e password per poter accedere ai dati di cui è proprietario e poter effettuare qualsiasi operazione su di essi. Solo

dopo questa fase l'utente può aggiungere un nuovo dato (metodo **put** che aggiunge un dato se non è già presente), può copiare il dato, ovvero può diventarne il proprietario (metodo **copy** che aggiunge il dato nella collezione dell'utente solo se un'altro utente ha condiviso quel dato con lui), può decidere di rimuovere un dato (metodo **remove**: rimuove un dato solo se ne è proprietario, ovvero solo se è presente nella collezione dei dati), può decidere di condividere un dato con un altro utente anch'esso registrato alla stessa collezione (metodo **share** che va ad aggiungere l'utente con cui voglio condividere il dato nella lista degli utenti condivisi in corrispondenza di quel dato).
Inoltre può ottenere una copia del valore del dato (metodo **get** che restituisce una copia solo se esso è stato condiviso con l'utente che ha effettuato l'accesso oppure se l'utente è direttamente il proprietario di quel dato) oppure può ottenere il numero di dati presenti (metodo **getSize**). Per restituire l'insieme dei dati di un determinato utente, utilizzo il metodo **getIterator.** Infine ho aggiunto un metodo **size** che permette di restituire il numero di utenti registrati alla collezione.

Sono presenti inoltre altre classi che codificano le eccezioni utilizzate, tra cui: "IllegalOwnerException" viene lanciata se un utente non supera i controlli di identità, "NoDataException" viene lanciata in caso di ricerca di un elemento non presente, "DuplicateDataException" viene lanciata quando si vuole effettuare una copia di un dato già presente nella collezione, "DuplicateIdException" viene lanciata quando registro un utente già registrato alla ollezione , "NoOtherException" lanciata se l'utente con cui voglio condividere il dato non è presente tra gli utenti registrati, e infine "BadSharedException" lanciata se provo a condividere un dato ad un utente che ha lo stesso nomeId dell'utente che ha effettuato l'accesso.

Sono contenuti poi due Main: "TestMainV", "TestMainH" che una volta lanciati eseguono un test che verifica le funzioni anche in casi limite.

- aggiungo 3 utenti: (ut1, ut2, ut3) con esito positivo, successivamente provo ad aggiungere un utente già presente e otterrò "id già presente".

- inserisco 3 dati (d1,d2,d3) a ut1 e 2 dati (d1,d4) a ut3; provo ad inserire un dato a ut3 con password errata ottenendo "accesso negato" e infine provo ad inserire un dato nullo: "parametri nulli"

- richiedo il numero di dati di ut1: "3" di ut2: "0" e di ut3: "2"

- condivido il dato d3 di ut1 con ut2 e il dato d4 di ut3 con ut2; provo a condividere un dato non presente: "dato non presente" ; provo a condividere un dato con un utente non registrato: "utente con cui condividere il dato non presente" e infine provo a condividere un dato con un utente uguale all'utente che ha effettuato l'accesso : "Utente con cui condividere il dato uguale all'utente che ha effettuato l'accesso"

- copio un dato che è stato condiviso con lui aggiungendolo nella lista dei dati personali; copio un dato che non è presente ne nei dati personali ne negli utenti condivisi allora: “dato non presente”; copio un dato che è già presente nei dati personali allora: “Dato già presente “.

- richiedo numero di elementi di u2: “1”.

- richiedo la copia del valore di d3 dell’ut2 : “dato3”; richiedo la copia del valore di d4 dell’ut2 : “dato4”; richiedo la copia del valore di d7 dell’ut2 ma il dato non è presente: “null”;

- rimuovo d3 da ut1 : “dato3”; rimuovo d8 da ut1 ma il dato non è presente: “null”;

- itero sui dati di ut1: “dato1-dato2”; itero sui dati di ut2: “dato3”; itero sui dati di ut3: “dato1-dato4”;

- restituisco il numero di utenti registrati: “3”