[Relazione sul Progetto di PRL2]

[Nome]: Loris [Cognome]: Giunta [Matricola]: 560451 [Corso]: B

Breve descrizione delle scelte stilistiche e progettuali

Ho usato due implementazioni differenti: MySecureDataContainer sfrutta una coppia formata da: un vettore di utenti e da un altra coppia: un dato e un vettore con il proprietario in prima posizione , e il resto occupato dagli altri utenti ai quali è permesso di accedere a quel dato. Questa struttura dati viene definita nella classe StructureData. Solo il proprietario può condividere , eliminare, vedere e inserire un dato. Se il proprietario condivide il suo dato con un altro utente, quest'ultimo ne diventa automaticamente proprietario alla pari del primo. Se un proprietario decide di eliminare un suo dato, allora verrà eliminato anche per gli utenti ai quali quel dato era stato condiviso. La rimozione riguarda un singolo dato, se ci sono più copie di quel dato, ne verrà cancellato soltanto uno alla volta. In generale in questa implementazione si da priorità al dato.
La seconda implementazione, MySecureDataContainerHashMap, utilizza due HashMap e un vettore di dati. L'HashMap Utenti rappresenta l'utente, associando al suo Id la sua password. L'HashMap DataMap associa ad un Id un vettore di tutti i dati accessibili alla chiave Id. Chi ha accesso al dato può eseguire le varie operazioni possibili. Anche qui rimozione e condivisione avvengono allo stesso modo che in MySecureDataContainer. In questa implementazione ha priorità l'utente.

Descrizione dei metodi utilizzati in MySecureDataContainer

METODI UTILIZZATI	DESCRIZIONE DEL METODO
public void createUser(String Id, String passw)	Crea un nuovo utente con Id e passw come credenziali
public int getSize(String Owner, String passw)	Restituisce il numero di dati posseduti dall' utente Owner
public boolean put(String Owner, String passw, E data)	Restituisce true se l'inserimento del dato è andato a buon fine, false altrimenti
private boolean AccessPermitted(String owner, String passw)	Restituisce true se l'utente supera i controlli di identità, false altrimenti
private boolean UserExists(String IdUser);	Restituisce true se l'utente IdUser esiste , false altrimenti
public void share(String Owner, String passw, String Other, E data)	Owner condivide il suo data con l 'utente Other, se Other esiste e vengono superati i controlli di identità
public E get(String Owner, String passw, E data)	Restituisce una copia di data se i controlli vengono superati, null altrimenti
public void copy(String Owner, String passw, E data)	Copia data se vengono superati i controlli
public E remove(String Owner, String passw, E data)	Rimuove data dalla collezione di ciascun utente che può accedervi e lo restituisce. Se la rimozione non va a buon fine ritorna null

private StructureData<E> getStructureData(String Owner,E data)	Restituisce data se viene trovato e Owner può accedervi, null altrimenti
public Iterator<E> getIterator(String Owner, String passw)	Restituisce un iteratore dei dati posseduti da Owner, se vengono superati i controlli di identità

Descrizione dei metodi utilizzati in MySecureDataContainerHashMap

METODI UTILIZZATI	DESCRIZIONE DEL METODO
public void createUser(String Id, String passw)	Crea un nuovo utente con Id e passw come credenziali
private boolean UserExists(String IdUser);	Restituisce true se l'utente IdUser esiste , false altrimenti
public int getSize(String Owner, String passw)	Restituisce il numero di dati posseduti dall' utente Owner
public boolean put(String Owner, String passw, E data)	Restituisce true se l'inserimento del dato è andato a buon fine, false altrimenti
private boolean AccessPermitted(String owner, String passw)	Restituisce true se l'utente supera i controlli di identità, false altrimenti
public void share(String Owner, String passw, String Other, E data)	Owner condivide il suo data con l 'utente Other, se Other esiste e vengono superati i controlli di identità
public E get(String Owner, String passw, E data)	Restituisce una copia di data se i controlli vengono superati, null altrimenti
public E remove(String Owner, String passw, E data)	Rimuove data dalla collezione di ciascun utente che può accedervi e lo restituisce. Se la rimozione non va a buon fine ritorna null
public void copy(String Owner, String passw, E data)	Copia data se vengono superati i controlli di identità
private boolean DataExists(String Owner, E data)	Restituisce true se data è contentuo nella collezione di Owner, false altrimenti
private E removeAll(E data)	Viene usata in Remove per cercare data nella collezione dei vari utenti ai quali era stato condiviso, eliminarlo e restituirlo
private boolean DataForAll(E data)	Restituisce true se data è presente nella collezione di tutti i dati, false altrimenti
public Iterator<E> getIterator(String Owner, String passw)	Restituisce un iteratore dei dati posseduti da Owner, se vengono superati i controlli di identità

Istruzioni su come eseguire il programma

Il main è provvisto di una batteria di test completa, nella quale vengono testati i metodi nei differenti casi in cui si presentano. Nel main è già stato istanziato l'oggetto con le due classi che implementano l'interfaccia, di cui una è commentata e l'altra no. Per eseguire occorre scegliere quale delle due classi usare: se la scelta è MySecureDataContainer basta eseguire il programma, altrimenti se la scelta è MySecureDataContainerHashMap occorre decommentarla e commentare l' altra istanza.