

Università degli Studi di Napoli Federico II  
Facoltà di Ingegneria  
Corso di Laurea in Ingegneria Informatica LM

Testing dell'applicazione java Ricorda Password

Candidato:  
Sajmir Rusi

Relatore  
Prof. Porfirio Tramontana



<b>Introduzione</b>	<b>5</b>
<i>Requisiti funzionali</i>	<i>5</i>
<b>Debugging</b>	<b>6</b>
<b>Testing Black Box</b>	<b>8</b>
<b>Testing della GUI</b>	<b>12</b>
<b>Verifica della copertura White-Box</b>	<b>14</b>



# Introduzione

## L'applicazione Ricorda Password

Il presente elaborato ha come obiettivo il testing dell'applicazione "Ricorda Password". Tale applicazione consiste in un Tool che consente di:

- salvare delle informazioni come Nome, Username e relativa password in un file .txt
- Recuperare e visualizzare tali informazioni
- Eliminare un specifico Username
- Effettuare un backUp dei account salvati in precedenza.

**Ricorda Password**

**Visualizza Password**

sajmir *Seleziona*

sajrus *Username*

popolo *Password*

Elimina elemento

**Aggiungi un nuovo elemento**

*Nome*

*Username*

*Password*

Pulisci Salva

Esegui backup

## Requisiti funzionali

E' possibile quindi identificare i seguenti requisiti:

- Rappresentazione grafica della app con i campi necessari per la visualizzazione e l'inserimento delle informazioni.
- Possibilità di scegliere una delle password da visualizzare, tramite un menu a tendina.
- Possibilità di eliminare un specifico account e visualizzazione di un messaggio di conferma.
- Salvataggio di un nuovo account e visualizzazione di un messaggio di conferma.
- Possibilità di resettare i campi inseriti.
- Salvataggio di un backUp dei account memorizzati in precedenza.

## Debugging

Si è notato da subito un difetto nell'interfaccia dell'applicazione. La rappresentazione del nome utente e della password invertiti.

Per questo motivo si è pensato di risolvere questo bug prima di andare avanti, aiutati anche dal debugger (ovviamente cercando per parole chiavi come [Aggiungi un nuovo elemento](#) all'interno del file la posizione dove venivano definiti gli elementi della UI) si è notato che gli elementi della UI compromessi nel bug vengono definite all'interno del metodo initComponents(), nelle righe 105-111 invece viene assegnato una label statica (non viene utilizzato l18N) a tutti gli componenti della UI.

```
jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel4.setText("Aggiungi un nuovo elemento");

user.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel6.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel6.setText("Username");

psw.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel5.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel5.setText("Password");

nome.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel7.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel7.setText("Nome");
```

per risolvere l'errore è stato sufficiente rinominare gli elementi jLabel15 e jLabel16 della GUI, rispettivamente jLabel15 --> jLabel16 e viceversa jLabel16 --> jLabel15.

```
jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel4.setText("Aggiungi un nuovo elemento");

user.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel5.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel5.setText("Username");

psw.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel6.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel6.setText("Password");

nome.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

jLabel7.setFont(new java.awt.Font("Tahoma", 3, 11)); // NOI18N
jLabel7.setText("Nome");
```

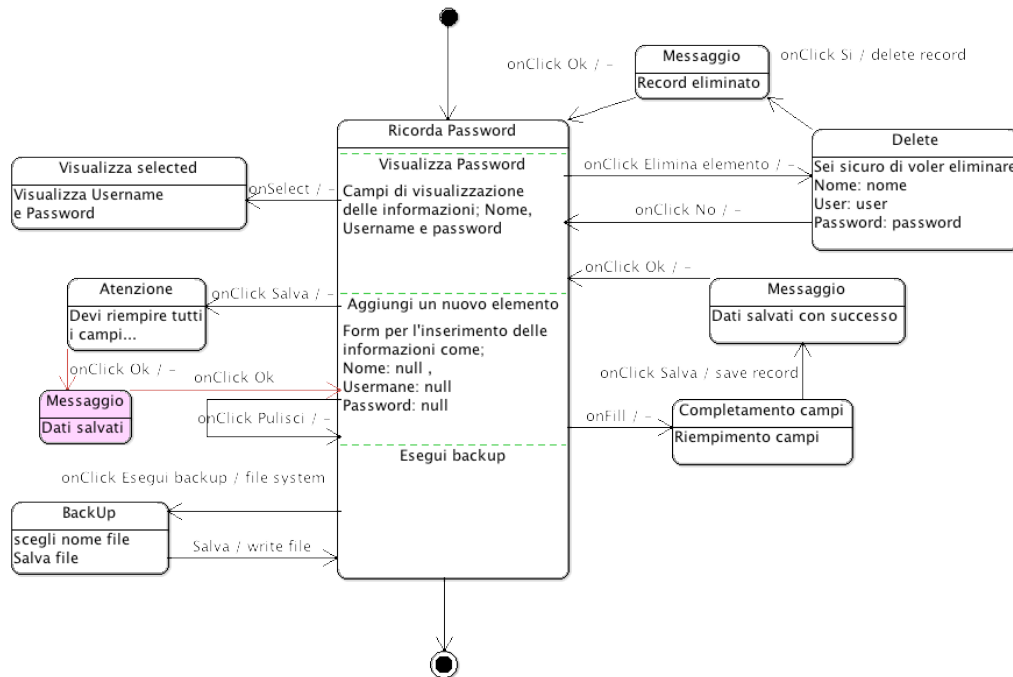
La GUI prima della correzione del bug, si vede chiaramente che gli elementi “Password” e “Username” risultano invertiti, questo dettaglio spinge l’utente di memorizzare in modo errato l’username e la password relativa.



Dopo la correzione del bug l’ordine degli elementi della GUI nel form di salvataggio della password è lo stesso come nel form di visualizzazione delle password salvate.

## Testing Black Box

Una volta corretto il bug si è passati alla definizione della Finite State Machine (FSM) tramite reverse engineering a partire dalla User Interface effettivamente implementata.



Tale diagramma FSM è caratterizzato da:

- Stati: corrispondenti alle interfacce grafiche (o parte di loro) presentabili all'utente
- Transazioni: sono identificate dagli archi e consentono il passaggio da uno stato all'altro

Per ciascuna transazione si può notare la presenza di tre parametri:

**trigger [guardia]/azione**

il primo detto trigger o evento, indica l'evento che attiva il passaggio di stato; il secondo, detto guardia, è un'espressione predicativa associata ad un evento, che stabilisce una condizione booleana che deve essere verificata affinché scatti la transizione; il terzo, rappresenta il risultato, l'output o l'operazione che segue un evento.

Nel nostro caso sulle transizioni è presente esclusivamente il trigger perciò bisogna testare il funzionamento dell'evento che consente il passaggio del stato.

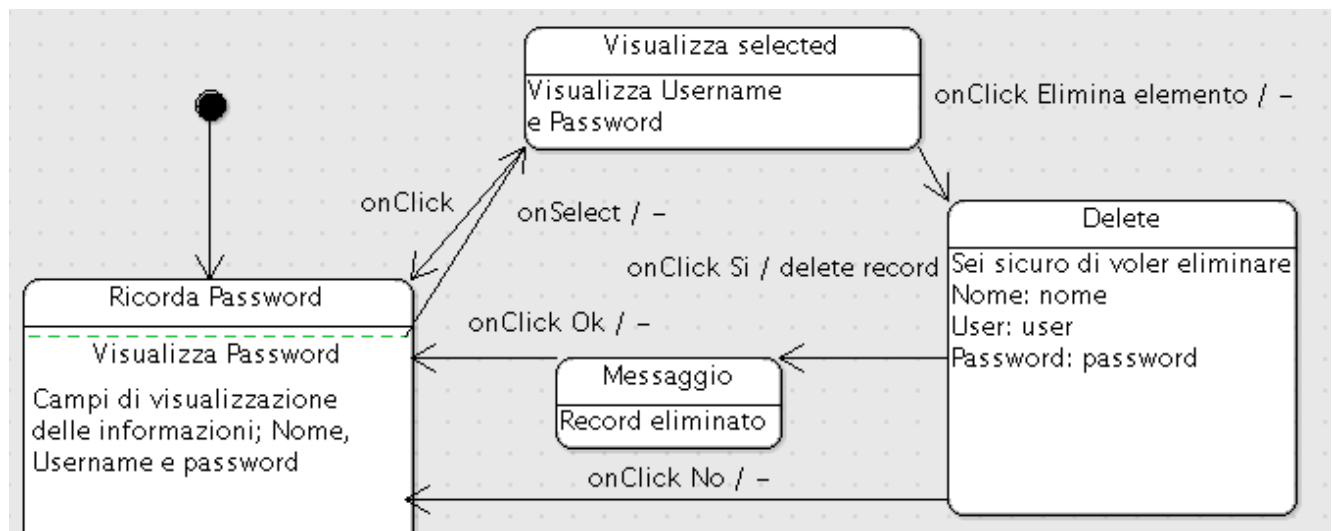


Di seguito andiamo a elencare i casi di test che coprono tutte le transizioni:

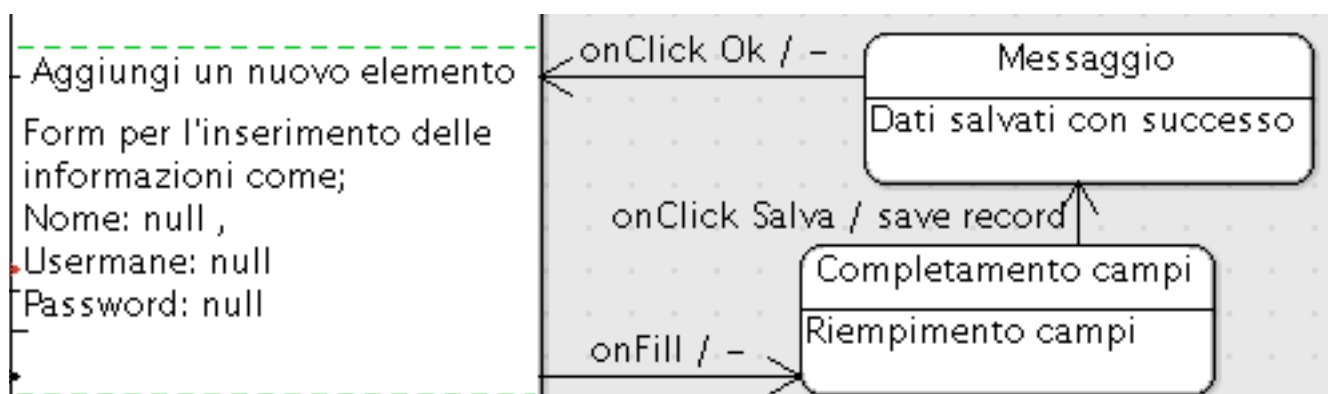
TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale
testRpswGUI1	VisualizzaPass	-onScroll down -onClick		Visualizza: -Username -Password

TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale	Post condizione
testRpswGUI2	VisualizzaPass	-onScroll down -onClick -onClick Elimina -onClick Si -onClick Si	-Visualizza selected -Delete -Messaggio	Visualizza Password	-Record eliminato dal db

TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale	Post condizione
testRpswGUI3	VisualizzaPass	-onScroll down -onClick -onClick Elimina -onClick No	-Visualizza selected	Visualizza Password	-Record non eliminato dal db

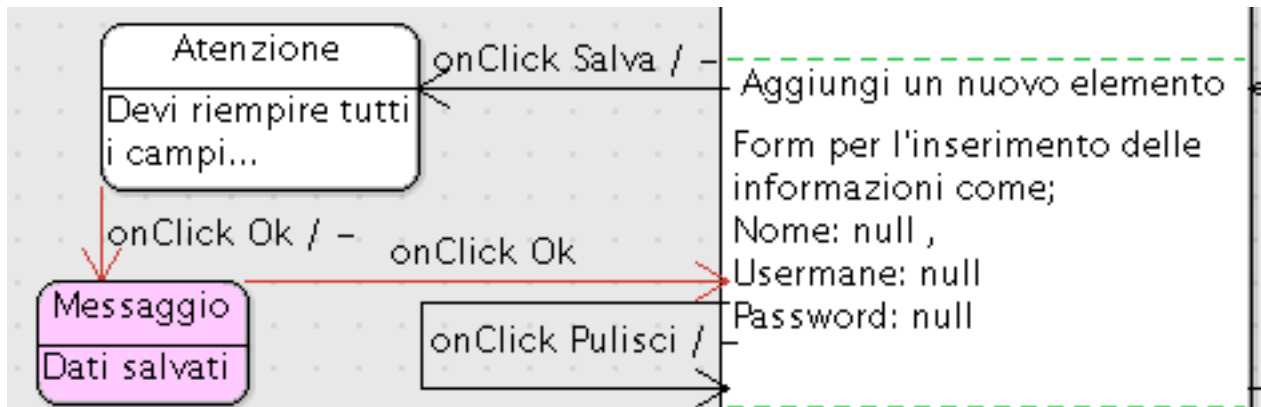


TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale
testRpswGUI4	Aggiungi un nuovo elemento	-onFill -onClick Salva -onClick Ok	-Completamento campi -Messaggio	Aggiungi un nuovo elemento

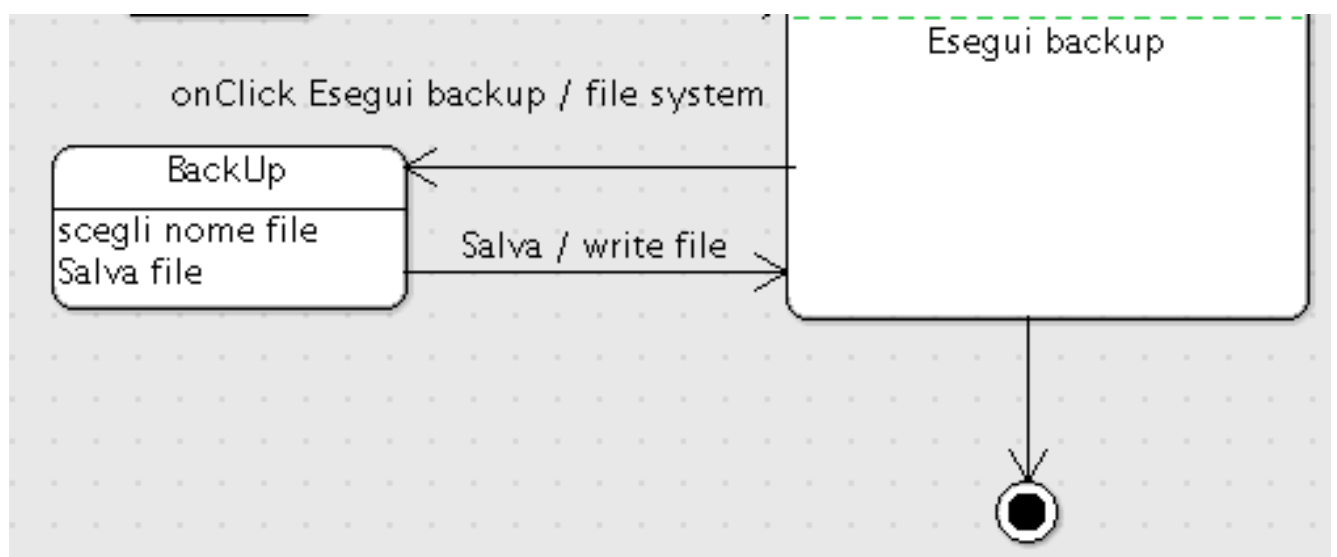


TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale
testRpswGUI5	Aggiungi un nuovo elemento	-onClick Salva -onClick Ok -onClick Ok	-Attenzione -Messaggio	Aggiungi un nuovo elemento

TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale
testRpswGUI6	Aggiungi un nuovo elemento	-onClick Salva		Aggiungi un nuovo elemento

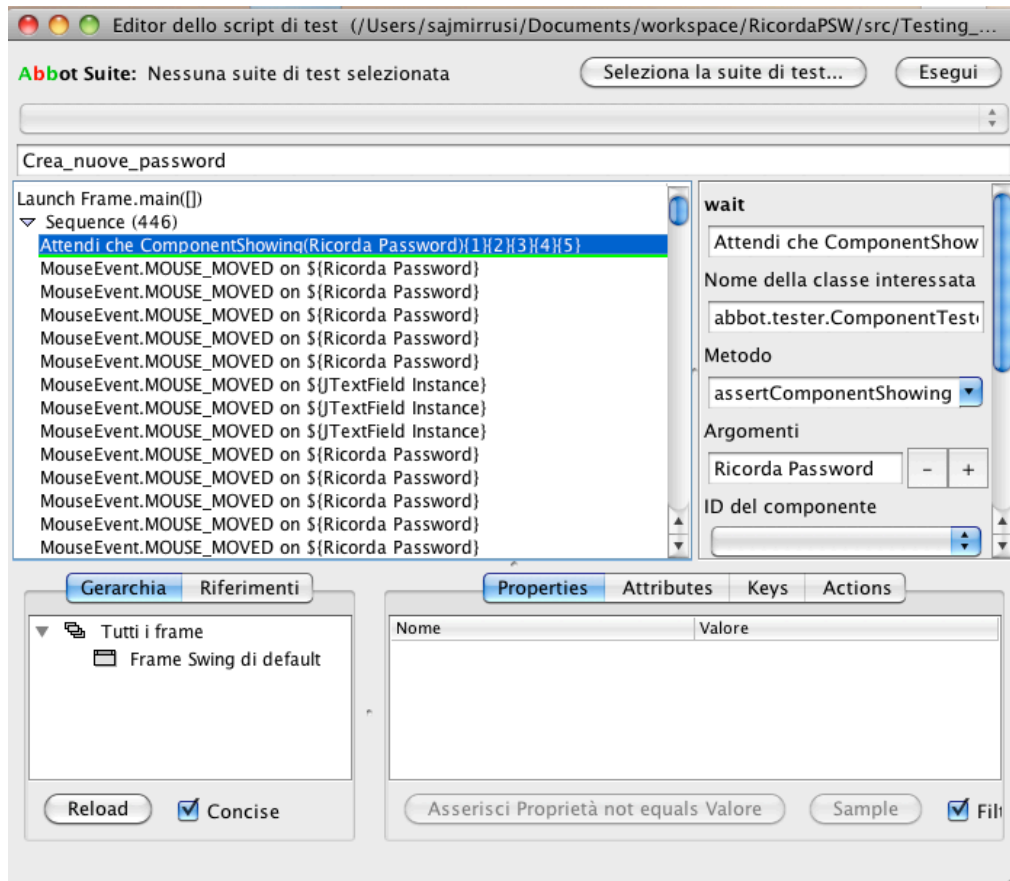


TestCase	Stato Iniziale	Sequenza di eventi	Stati Intermedi	Stato Finale	Post condizioni
testRpswGUI 7	Esegui backup	-onClick Esegui backup -onClick Salva	BackUp	Esegui backup	



## Testing della GUI

Per testare l'interfaccia grafica dell'applicazione è stato utilizzato lo strumento Abbot-Costello che ha consentito di creare diversi script di test cases registrando l'interazione con la GUI.



Sono stati dunque registrati interazioni al fine di testare:

- Creazione di nuovo account Utente
- Visualizzazione di un account esistente
- Eliminazione di un account
- Back Up dei account

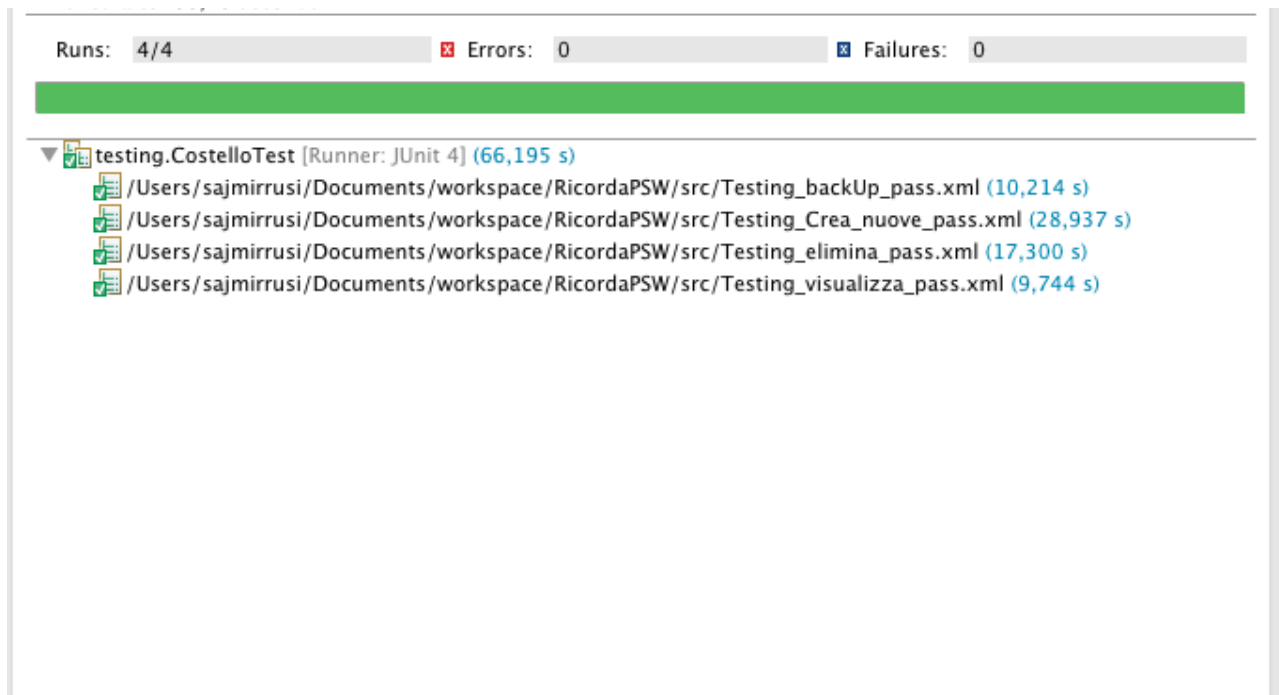
I test case generati sono stati quindi incorporati in una suite junit tramite la classe CostelloTest.java la quale estende da ScriptFixture.java, messa a disposizione da Abbot a tale scopo.

Sono stati scelti dei test case per creare lo script:

Test Case	Azione	Precondizioni	Input Fill/Click	Post condizione	Esito
tc1	Nuovo Utente	Utente non presente in db	form Salva Ok	Account presente in db	false
tc2	Visualizza Utente	-	Select View	-	false
tc3	Elimina Utente	Utente presente in db	Select Elimina Si Ok	Utente non più presente in db	true
tc4	Backup	file backup non presente	Esegui Backup Fill name Salva	File backUp presente nella directory	false

### Esecuzione della test Suite

Riportiamo di seguito i risultati dell'esecuzione in Costello dei Test Cases relativi alla GUI



Runs: 4/4    Errors: 0    Failures: 0

testing.CostelloTest [Runner: JUnit 4] (66,195 s)

- ✓ /Users/sajmirrusi/Documents/workspace/RicordaPSW/src/Testing\_backUp\_pass.xml (10,214 s)
- ✓ /Users/sajmirrusi/Documents/workspace/RicordaPSW/src/Testing\_Crea\_nuove\_pass.xml (28,937 s)
- ✓ /Users/sajmirrusi/Documents/workspace/RicordaPSW/src/Testing\_elimina\_pass.xml (17,300 s)
- ✓ /Users/sajmirrusi/Documents/workspace/RicordaPSW/src/Testing\_visualizza\_pass.xml (9,744 s)

## Verifica della copertura White-Box

A partire dai Test Cases realizzati ho verificato la copertura White-Box mediante lo strumento CodeCover. Le statistiche fanno riferimento alla copertura dei comandi, delle condizioni e delle decisioni

☐ Show methods with Statement Coverage <= 90,5 %

Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
▼ RicordaPSW	96,6 %	58,3 %	33,3 %	70,0 %	-	-
▼ Frame	96,6 %	58,3 %	33,3 %	70,0 %	-	-
Frame	88,9 %	50,0 %	-	-	-	-
Record	100,0 %	-	-	-	-	-
backMouseClicked	93,8 %	50,0 %	33,3 %	75,0 %	-	-
comboltemStateChanged	100,0 %	100,0 %	-	100,0 %	-	-
deletMouseClicked	100,0 %	66,7 %	33,3 %	75,0 %	-	-
deleteMouseClicked	100,0 %	-	-	-	-	-
init	92,9 %	50,0 %	33,3 %	75,0 %	-	-
initComponents	100,0 %	-	-	-	-	-
main	100,0 %	50,0 %	-	-	-	-
saveMouseClicked	83,3 %	50,0 %	33,3 %	50,0 %	-	-

Come si può notare dal report, i test effettuati hanno consentito di ottenere un buona percentuale di copertura dei metodi della classe Frame.