Documento dei Requisiti

REQUISITI

Requisiti funzionali

1. La calcolatrice utilizza una struttura dati di tipo stack per la memorizzazione dei dati.
2. Lo stack deve supportare numeri complessi in notazione cartesiana.
3. L'interfaccia utente:
   3.1. mostra fino a 12 elementi memorizzati.
   3.2. ha un'area di testo dove l'utente inserirà l'input.
   3.3. consente la visualizzazione del valore variabili.
   3.4. Implementa bottoni per la manipolazione stack.
   3.5. Implementa bottoni numeri, variabili, operazioni binarie e j che rappresenta la parte immaginaria.
   3.6. Implementa bottoni operazioni unarie.
4. La calcolatrice deve supportare le seguenti operazioni:
   4.1. "+": Addizione.
   4.2. "-": Sottrazione.
   4.3. "*": Moltiplicazione.
   4.4. "/": Divisione.
   4.5. "sqrt": Radice Quadrata.
   4.6. "+-": Inversione del Segno.
5. L'esecuzione delle operazioni segue l'ordine di ingresso input.
6. Manipolazione stack
   6.1. Clear: rimuove tutti gli elementi dallo stack.
   6.2. Drop: rimuove l'ultimo elemento inserito.
   6.3. Dup: copia l'ultimo elemento e lo inserisce.
   6.4. Swap: inverte l'ordine degli ultimi due elementi.
   6.5. Over: inserisce una copia nello stack del penultimo elemento inserito.
7. La calcolatrice supporta 26 variabili (case insensitive).
8. Operazioni sulle variabili:
   8.1. ">x": prende l'ultimo elemento inserito e lo salva nella variabile x.
   8.2. "<x": prende il valore salvato in x e lo inserisce nello stack.
   8.3. "+x": prende l'ultimo elemento inserito nello stack, lo addiziona al valore della variabile x e salva il risultato dell'addizione nella variabile x.
   8.4. "-x": sottrae l'ultimo elemento dello stack al valore nella variabile x e salva il risultato nella variabile x.

Requisiti non funzionali*

9. Interfaccia utente intuitiva.
10. Gestire errori di calcolo e input (eg. divisione per 0), senza interrompere l'esecuzione e mostrando un opportuno messaggio di errore a video.
11. Le variabili in input sono case insensitive.
12. Il codice sorgente segue le convenzioni di codifica standard di Java.
13. Il software è compatibile con i sistemi operativi Windows, MacOS e Linux.

*La scelta dei requisiti non funzionali sopra elencati è stata fatta per garantire al software le seguenti proprietà: Usabilità, affidabilità, manutenibilità e compatibilità.

TABELLA DELLE PRIORITÀ

id	1	2	3.1	3.2	3.3	3.4	3.5	3.6	4.1	4.2	4.3	4.4	4.5	4.6	5
priorità	alta	alta	alta	alta	bassa	alta	media	media	alta	alta	alta	alta	alta	alta	alta

6.1	6.2	6.3	6.4	6.5	7	8.1	8.2	8.3	8.4	9	10	11	12	13
alta	alta	alta	alta	alta	alta	alta	alta	alta	alta	media	alta	alta	alta	media

MATRICE DI TRACCIABILITÀ

ID	Requisiti	Design	Implementazione	Testing
1	implementazione stack			
2	supporto della notazione cartesiana per i numeri complessi			
3.1	visualizzazione di 12 elementi sullo stack			
3.2	visualizzazione input utente			
3.3	visualizzazione variabili			
3.4	bottoni manipolazione stack			
3.5	bottoni di: numeri, variabili, operazioni binarie, "j"			
3.6	bottoni operazioni unarie			
4.1	addizione			
4.2	sottrazione			
4.3	moltiplicazione			
4.4	divisione			
4.5	radice quadrata			
4.6	inversione del segno			
5	esecuzione operazioni in ordine di inserimento input			
6.1	clear			
6.2	drop			
6.3	dup			
6.4	swap			
6.5	over			
7	supporto di 26 variabili			
8.1	>x			
8.2	<x			
8.3	+x			
8.4	-x			

USE CASES DESCRIPTION

ID: 1

NOME: Aggiunta Numero

PRE-CONDIZIONI: L'utente deve aver inserito un numero.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso il numero, in notazione cartesiana, inserito dall'utente.

| Step 2 |: Il numero viene aggiunto in testa allo stack.

| Step 3 |: Il sistema ritorna in attesa di un nuovo inserimento.

POST-CONDIZIONI: Lo stack contiene in testa il numero inserito.

ECCEZIONI:

| Step 1 |: Se l'utente inserisce il numero in un formato non valido viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

| Step 2 |: Se lo stack è pieno viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato

ID: 2

NOME: Operazione Binaria

PRE-CONDIZIONI: L'utente deve aver inserito un operatore binario.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso l'operatore binario inserito dall'utente.

| Step 2 |: Il sistema riconosce l'operatore e lo identifica:

| Caso 1 |: "+" = addizione;

| Caso 2 |: "-" = sottrazione;

| Caso 3 |: "*" = moltiplicazione;

| Caso 4 |: "/" = divisione;

| Step 3 |: Il sistema differenzia il funzionamento dell'operatore:

| Caso 1 |: Il sistema somma tra loro gli ultimi due elementi dello stack;

| Caso 2 |: Il sistema sottrae l'ultimo elemento dello stack al penultimo;

| Caso 3 |: Il sistema moltiplica tra loro gli ultimi due elementi dello stack;

| Caso 4 |: Il sistema divide il penultimo elemento dello stack per l'ultimo;

| Step 4 |: Il sistema rimuove gli ultimi due elementi dello stack e aggiunge in testa il risultato dell'operazione.

POST-CONDIZIONI: Allo stack è stato aggiunto in testa il risultato dell'operazione e sono stati rimossi gli elementi coinvolti nell'operazione.

ECCEZIONI:

| Step 2 |: Se l'operatore inserito non è compatibile viene mostrato a schermo un messaggio di errore che indica quale sia il problema e le use case viene cancellato.

| Step 3 |: Se lo stack non contiene almeno due elementi viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

|Step 3 | Caso 4 |: Se viene effettuata una divisione per 0 il programma mostra un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

ID: 3

NOME: Operazione Unaria

PRE-CONDIZIONI: L'utente deve aver inserito un operatore unario.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso l'operatore unario inserito dall'utente.

| Step 2 |: Il sistema riconosce l'operatore e lo identifica:

| Caso 1 |: "+-" = inversione del segno;

| Caso 2 |: "sqrt" = radice quadrata;

| Step 3 |: Il sistema differenzia il funzionamento dell'operatore:

| Caso 1 |: Il sistema inverte il segno dell'ultimo elemento dello stack;

| Caso 2 |: Il sistema calcola la radice quadrata dell'ultimo elemento dello stack;

| Step 4 |: Il sistema rimuove l'ultimo elemento dello stack e aggiunge in testa il risultato dell'operazione.

POST-CONDIZONI: Allo stack è stato aggiunto in testa il risultato dell'operazione ed è stato rimosso l'elemento coinvolto nell'operazione.

ECCEZIONI:

| Step 2 |: Se l'operatore inserito non è compatibile viene mostrato un messaggio di errore che indica quale sia il problema questo use case viene cancellato.

| Step 3 |: Se lo stack non contiene almeno un elemento viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

ID: 4

NOME: Comando Binario

PRE-CONDIZIONI: L'utente deve aver inserito un comando binario.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso il comando binario inserito dall'utente.

| Step 2 |: Il sistema riconosce il comando e lo identifica:

| Caso 1 |: "swap" = scambia ultimo con penultimo;

| Caso 2 |: "over" = copia penultimo elemento;

| Step 3 |: Il sistema differenzia il funzionamento del comando:

| Caso 1 |: Il sistema scambia di posto l'ultimo e il penultimo elemento;

| Caso 2 |: Il sistema fa una copia del penultimo elemento e lo aggiunge in testa allo stack;

POST-CONDIZONI:

| Caso 1 |: Allo stack sono stati scambiati di posto ultimo e penultimo elemento inserito;

| Caso 2 |: Allo stack è stata aggiunta in testa una copia del penultimo elemento inserito;

ECCEZIONI:

| Step 2 |: Se il comando inserito non è compatibile viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

| Step 3 |: Se lo stack non contiene almeno due elementi viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

| Step 3 | Caso 2 |: Se lo stack è pieno viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

ID: 5

NOME: Comando Unario

PRE-CONDIZIONI: L'utente deve aver inserito un comando unario.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso il comando binario inserito dall'utente.

| Step 2 |: Il sistema riconosce il comando e lo identifica:

| Caso 1 |: "clear" = svuota lo stack.

| Caso 2 |: "drop" = rimuove l'ultimo.

| Caso 3 |: "dup" = duplica l'ultimo.

| Step 3 |: Il sistema differenzia il funzionamento del comando:

| Caso 1 |: Il sistema rimuove tutti gli elementi dello stack.

| Caso 2 |: Il sistema rimuove l'ultimo elemento dello stack.

| Caso 3 |: Il sistema fa una copia dell'ultimo elemento e lo inserisce in testa.

POST-CONDIZONI:

| Caso 1 |: Lo stack non contiene nessun elemento.

| Caso 2 |: Allo stack è stato rimosso l'ultimo elemento.

| Caso 3 |: L'ultimo elemento dello stack è stato duplicato.

ECCEZIONI:

| Step 2 |: Se il comando inserito non è compatibile viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

| Step 3 |: Se lo stack non contiene almeno un elemento viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

| Step 3 | Caso 3 |: Se lo stack è pieno viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

ID: 6

NOME: Modifica Delle Variabili

PRE-CONDIZIONI: Il sistema riceve in ingresso il comando di modifica di una variabile.

SEQUENZA:

| Step 1 |: Il sistema riceve in ingresso il comando inserito dall'utente.

| Step 2 |: Il sistema riconosce l'operazione sulla variabile e la identifica.

| Caso 1 |: ">x" = salva nella variabile.

| Caso 2 |: "<x" = prendi dalla variabile.

| Caso 3 |: "+x" = addiziona all'interno della variabile.

| Caso 4 |: "-x" = sottrae all'interno della variabile.

| Step 3 |: Il sistema differenzia il funzionamento dell'operazione sulla variabile:

| Caso 1 |: Il sistema prende l'ultimo elemento inserito nello stack e lo salva all'interno della variabile indicata.

| Caso 2 |: Il sistema prende il valore all'interno della variabile indicata e lo copia aggiungendolo in testa allo stack.

| Caso 3 |: Il sistema aggiorna il valore della variabile indicata sommando al valore attuale il valore dell'ultimo elemento nello stack.

| Caso 4 |: Il sistema aggiorna il valore della variabile indicata sottraendo al valore attuale il valore dell'ultimo elemento nello stack.

POST-CONDIZONI:

| Caso 1 |: Il valore della variabile scelta è stato modificato

| Caso 2 |: Allo stack è stato aggiunto una copia del valore della variabile.

| Caso 3 |: La variabile è stata incrementata.

| Caso 4 |: La variabile è stata decrementata.

ECCEZIONI:

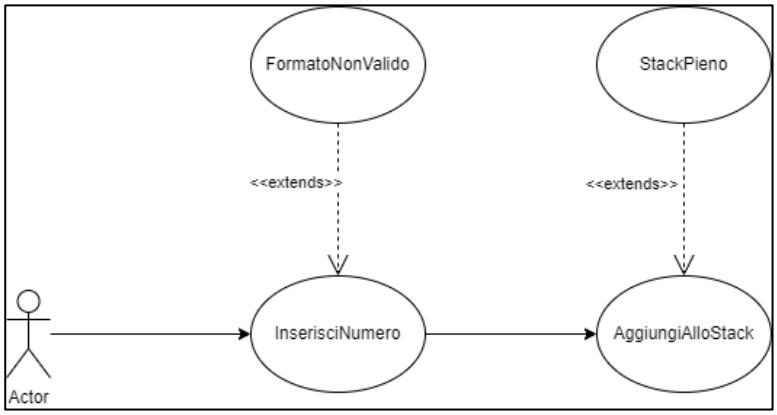
| Step 2 |: Se il comando inserito non è compatibile o la variabile non è compresa fra la A e la Z (Case Insensitive) viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

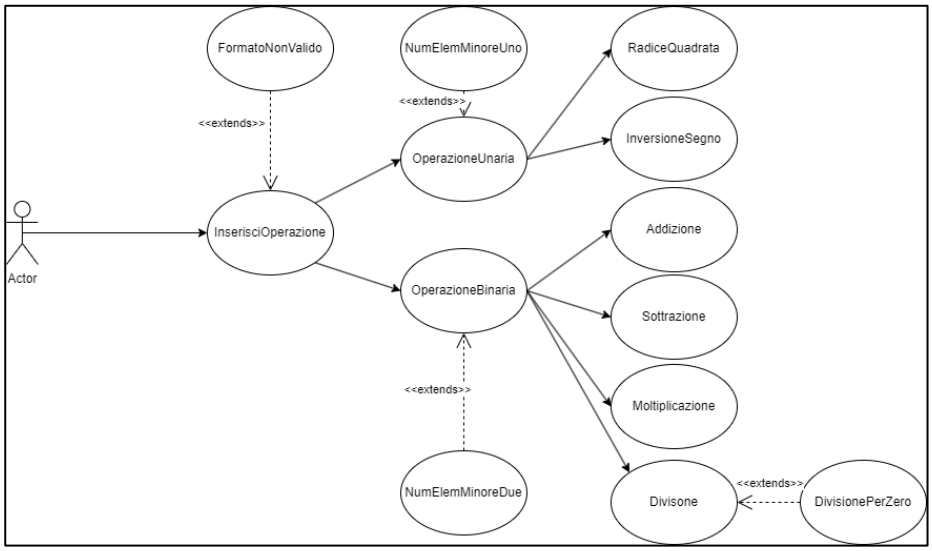
| Step 3 |: Se la variabile non è mai stata utilizza il sistema la inizializza a 0 prima di effettuare l'operazione su di essa e continuare la normale esecuzione del programma

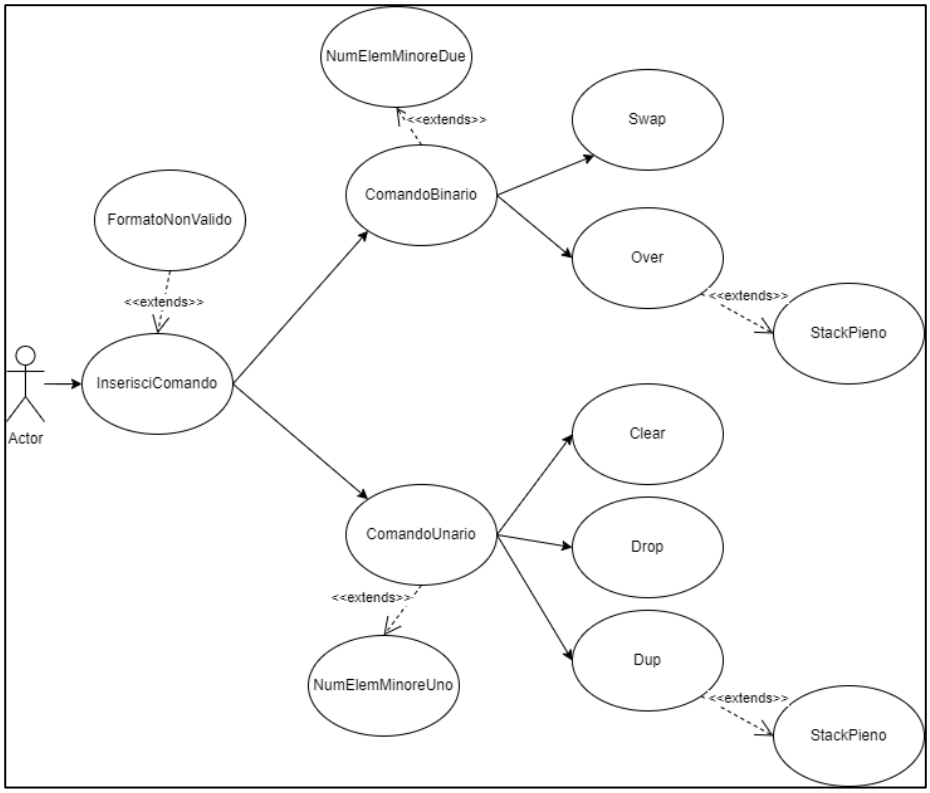
| Step 3 | Caso 2 |: Se lo stack è pieno viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

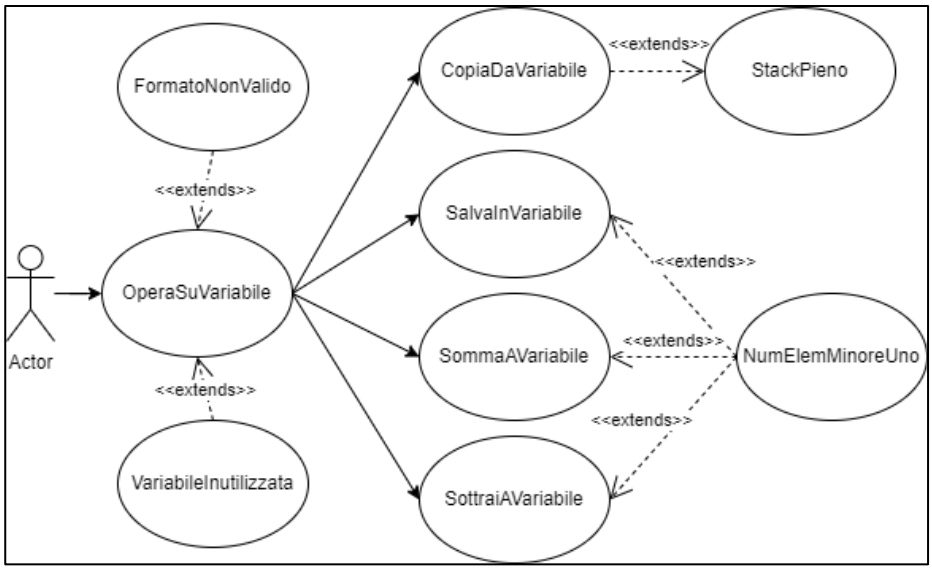
| Step 3 | Caso 1 | Caso 3 | Caso 4|: Se lo stack non contiene almeno un elemento viene mostrato un messaggio di errore che indica quale sia il problema e questo use case viene cancellato.

USE CASES DIAGRAMS

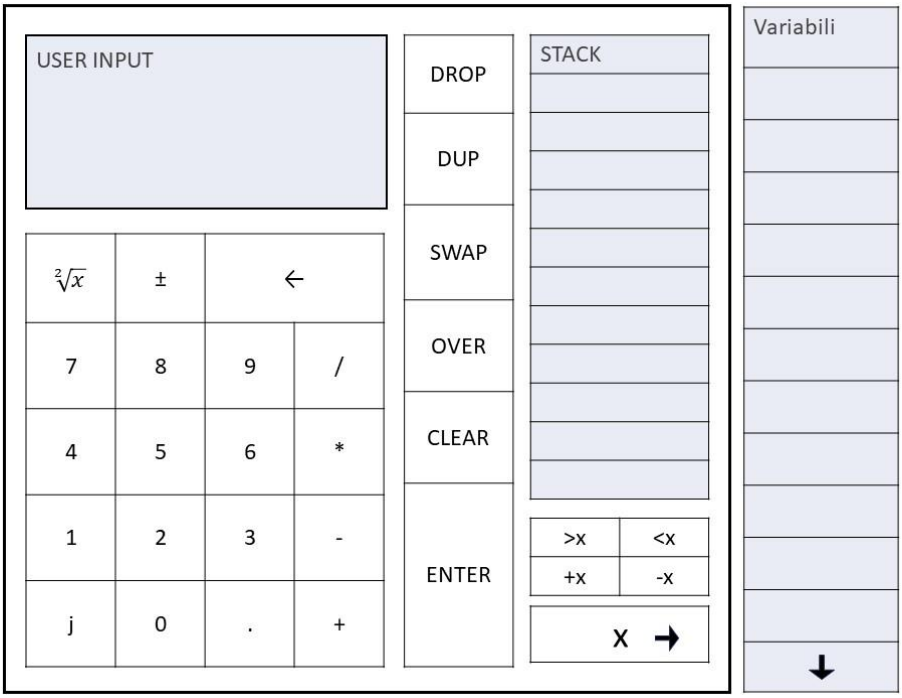







FormatoNonValido
<<extends>>
Actor
OperaSuVariabile
<<extends>>
VariabileInutilizzata
CopiaDaVariabile
<<extends>>
StackPieno
SalvaInVariabile
<<extends>>
SommaAVariabile
<<extends>>
NumElemMinoreUno
<<extends>>
SottraiAVariabile

INTERFACCIA DI MOCK-UP



*La freccia indica un menù a tendina, in particolare:

- : espande il menù delle variabili utilizzabili.
- : espande il menù di memorizzazione delle variabili, inizializzate e non, all'interno del programma.