PROGETTO VALUTAESPRESSIONE

Gaia Assunta Bertolino
Mat. 209507 A.A. 2020/2021
Corso di Programmazione orientata agli oggetti

Organizzazione delle classi:

Il progetto è stato organizzato secondo tre classi:
1) classe Algoritmo
2) classe Regex
3) classe ValutazioneGUI

Class Regex

La classe ha lo scopo di definire dei metodi che possano analizzare una stringa e verificare che sia riconducibile ad una espressione aritmetica corretta. Nonostante vi siano già delle verifiche all'interno dell'algoritmo principale, i metodi della classe Regex compiono ulteriori controlli e verificano anche, ad esempio, che le parentesi siano accoppiate, evitando dunque di invocare il metodo di calcolo principale inutilmente. I metodi da cui è composto sono:

- **static boolean corretta(String s) ->** metodo statico che verifica per prima cosa se l'ultimo carattere della stringa è un operatore. In caso contrario, continua analizzando tutta la stringa e incrementando o decrementando un'opportuna variabile contatore quando si incontra, rispettivamente, una parentesi aperta o una chiusa. Se terminato il ciclo, si riconosce che la variabile non è pari a 0 e dunque le parentesi non sono accoppiate, allora restituisce false. Infine, viene verificata la condiziona necessaria ma non sufficiente (tuttavia rafforzata dai controlli precedenti), secondo la quale la stringa è una espressione correttamente formulata, attraverso un match con una espressione regolare precedentemente impostata.
- **static boolean isNumero(String s) ->** metodo statico che verifica se la stringa passata come argomento sia riconducibile ad un numero attraverso il match con la classe di caratteri numerici.

Class Algoritmo

La classe ha lo scopo di implementare quello che è l'algoritmo di valutazione di una espressione. Ha come unica variabile d'istanza la stringa *risultato* con modificatore *protected* in modo che possa essere accessibile dalla GUI implementata attraverso la classe ValutazioneGUI presente nello stesso package. La classe è composta dai seguenti metodi:

- **static String valutaEspressione(StringTokenizer s) ->** metodo statico il cui scopo è di analizzare uno StringTokenizer, il quale a sua volta è impostato in ValutazioneGUI in modo da dividere la stringa-espressione quando incontra un operatore matematico o delle parentesi ("ricordandosi" tuttavia quale simbolo si è incontrato grazie al terzo parametro posto a true). Fin quando sono disponibili token, la funzione li preleva e opera delle verifiche: se lo riconosce come numero lo aggiunge allo stack degli operandi; se lo riconosce come una parentesi aperta allora richiama ricorsivamente se stesso e pusha nello stack degli operandi il risultato di questa invocazione poiché, infatti, quando il metodo stesso incontra una parentesi chiusa, "ripulisce" lo stack di operatori e operandi

creato dalla chiamata ricorsiva (calcolando le operazioni rimaste) e restituisce dunque il risultato finale dell'espressione fra parentesi, così che la prima chiamata del metodo possa dunque recuperarlo e aggiungerlo agli operandi; infine si è sicuri che il token incontrato è un operatore dunque si esegue la parte essenziale dell'algoritmo per cui si pusha l'operatore se il relativo stack è vuoto o esso è prioritario rispetto alla cima o altrimenti si procede col calcolo dell'operazione precedente (che sarà prioritaria dunque) per poi reinserire il risultato nello stack degli operandi; l'operazione viene eseguita tramite la funzione *calcola(String o1, String o2, char op)* mentre la valutazione della priorità viene eseguita tramite il metodo *prioritario(char op1, char op2)*. Dopo aver eseguito questa operazione, si opera nuovamente la verifica precedente così da eliminare eventuali operatori a loro volta prioritari rispetto a quello appena prelevato (si eliminano così anche eventuali sottrazioni). Terminata l'analisi dello StringTokenizer, si procede a svuotare lo stack degli operatori eseguendo tutte le operazioni rimaste (che ora avranno tutte la stessa priorità). Terminato il calcolo finale, se lo stack degli operandi non dovesse contenere un solo valore (cioè il risultato dell'espressione) allora verrebbe lanciata una IllegalArgumentException.

- **static String calcola (String o1, String o2, char op) ->** metodo statico che riconosce l'operatore passato come argomento e calcola l'operazione corrispondente. Eventuali errori di battitura o di definizione dell'espressione vengono gestiti direttamente nella GUI
- **static boolean prioritario(char op1, char op2) ->** metodo che valuta se il primo operatore passato come argomento sia prioritario rispetto al secondo

Class ValutazioneGUI

La classe ha lo scopo di implementare una GUI in cui l'utente può digitare l'espressione aritmetica da calcolare. La Gui è strutturata in modo che ad ogni digitazione venga calcolata la validità dell'espressione digitata così che si possa opportunamente disattivare il bottone che richiama il calcolo effettivo se si prova a digitare un'espressione incompleta, incorretta o dove sono presenti dei caratteri non riconosciuti. Dunque, non è possibile cliccare sul bottone di calcolo fin quando l'espressione digitata non sia correttamente strutturata. Questa impostazione è gestita dall'invocazione continua della funzione *checker()*. In particolare, la classe ValutazioneGUI ha la sola funzione di generare una finestra di tipo FinestraGUI attraverso un main.

Class FinestraGUI extends JFrame implements ActionListener

La classe ha lo scopo di struttura la finestra effettiva di interazione con l'utente. Fornisce i seguenti metodi:

- **FinestraGUI() ->** costruttore che definisce le specifiche della finestra e rende visibile un'area di testo in cui inserire l'espressione da calcolare
- **void actionPerformed(ActionEvent e) ->** metodo che viene richiamato ogni qualvolta si interagisca con uno dei componenti a cui è stato aggiunto l'ActionListener di this attraverso il comando *addActionListener(this)*. Nel momento in cui si clicca sul bottone "Calcola", viene creata una finestra di tipo FinestraRis in cui è mostrato il risultato. Per

l'implementazione precedentemente spiegata che gestisce eventuali malformazioni, la creazione di tale finestra avviene solo quando è effettivamente possibile calcolare un risultato.

- **boolean uscita() ->** metodo che viene richiamato quando si prova a chiudere una finestra. Esso visualizza una ulteriore finestra di conferma dell'uscita e, eventualmente, permette di non chiudere realmente la finestra.
- **void checker() ->** metodo che ha lo scopo di valutare se è stato digitato del testo e, eventualmente, valutare se effettivamente l'espressione inserita è calcolabile. In caso negativo, disattiva il bottone di calcolo della GUI mentre lo rende cliccabile se è possibile calcolare un risultato.
- **private class FinestraRis extends JFrame ->** classe privata il cui scopo è generare, tramite l'invocazione del suo costruttore, una finestra che visualizzi il risultato del calcolo dell'espressione aritmetica digitata.