PROGETTO POLINOMI

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

Organizzazione delle classi:
Il progetto è stato organizzato secondo sei classi (di cui le prime quattro elencate sono state sviluppate a lezione)

1) classe Monomio che implementa Comparable<Monomio>
2) classe Polinomio che estende Iterable<Monomio>
3) classe astratta PolinomioAstratto che implementa Polinomio
4) classe PolinomioSet che estende PolinomioAstratto
5) classe Regex
6) classe PolinomioGUI che estende JFrame e implementa ActionListener

Nella seguente relazione, vengono tralasciati il funzionamento delle prime quattro classi e si tratterà solamente della classe Regex, creata per poter valutare il polinomio passato attraverso una espressione regolare, e la classe PolinomioGUI, creata per generare una interfaccia grafica utente. Una precisazione importante va fatta sulla scelta di usare di PolinomioSet anziché PolinomioLL in quanto permette, per la natura intrinseca della variabile d'istanza *monomi* (un Set<Monomio>) risulta avere un metodo add più snello ed efficiente.

class Regex

La classe ha l'utilità di raccogliere i metodi di valutazione di una stringa. Essi sono:

- **static boolean riconoscimento(String s) ->** metodo statico che verifica la condizione necessaria ma non sufficiente affinché la stringa s passata sia un polinomio. Infatti, l'espressione regolare *POLINOMIO* comprende tutti i casi possibili, valutando opportunamente che un coefficiente possa essere composto da più numeri, che il primo coefficiente possa avere segno negativo (-) ma non positivo (+) in quanto non avrebbe senso, che l'incognita ci ciascun monomio possa essere composta da x o da x^ ciascuno ripetuto massimo una volta (es. 5xx^3 o 5x^^2 non vengono riconosciuti) e che se viene digitato un monomio con variabile x^ questa debba essere seguita per forza da un esponente
- **static Monomio monomio(String s, int i) ->** metodo statico che, data una stringa per argomento, costruisce un oggetto di tipo Monomio leggendo la stringa a partire dall'indice i e interpretando il primo carattere eventualmente come segno e i rimanenti come coefficienti fin quando non incontra la variabile x per cui non riconosce più i caratteri trovati come coefficiente ma come esponente del monomio, gestendo lo smistamento attraverso una flag booleana posta a true quando si incontra la x; se invece rincontra un segno, vuol dire che è terminata la lettura di un monomio. Successivamente, nel caso in cui non vi sia una variabile x allora il grado viene posto a 0 se non è stata trovata la x altrimenti è posto ad 1; nel caso in cui invece non siano stati trovati numeri al coefficiente, esso è sottinteso che sia 1 e dunque viene aggiunto. Il metodo può alla fine costruire un monomio con i parametri
-

ottenuti e viene utilizzato dal metodo *divisore(String s)* per dividere il polinomio stringa passato per costruire i monomi da cui è costituito.

- **static PolinomioSet divisore(String s) ->** metodo statico che costruisce un polinomio spezzando la stringa passata come argomento nel caso del primo indice (infatti il primo monomio di un polinomio può sottintendere il segno +) e ogni volta che incontra un segno + o – richiamando il metodo monomio(String, i) che costruisce il monomio come sopra descritto.

class PolinomioGUI

La classe ha l'utilità di generare una GUI dove è possibile inserire un polinomio nella barra in basse della finestra, visualizzare e selezionare i polinomi inseriti nella parte centrale della finestra e scegliere una operazione da eseguire nella barra dei menu in alto. La classe PolinomioGUI richiama nel proprio main la costruzione della finestra strutturata attraverso un'ulteriore classe chiamata FinestraGUI

class FinestraGUI extends JFrame implements ActionListener

La classe definisce come deve essere creata la finestra iniziale e presenta una LinkedList di PolinomioSet, permettendo così di avere sempre raccolti in una lista i polinomi salvati; l'accorgimento rende più facile l'esecuzione delle operazioni. In particolare, i metodi e le classi private da cui è costituito sono:

- **FinestraGUI()** -> costruttore dell'oggetto FinestraGUI. Imposta tutte le specifiche della finestra, compresi i bottoni, i menu, i messaggi da visualizzare e i pannelli che li comprendono. Inoltre, richiama il metodo *menuIniziale()* che ha lo scopo di disattivare tutti i bottoni dei menu chiaramente non utilizzabili visto che non sono presenti ancora polinomi
- **void refresh() ->** metodo che viene richiamato più volte nei metodi successivi e il cui scopo è, attraverso l'invocazione della funzione *counter()* che restituisce il numero di polinomi selezionati, valutare, dopo l'operazione in cui è invocato, quale tipo di operazioni rendere selezionali (es. se si selezionano 3 polinomi, la funzione renderà possibile calcola le derivate ma non renderà attiva l'opzione di addizione). Inoltre, richiama dei metodi di aggiornamento della finestra così da visualizzare le modificate apportate.
- **boolean uscita() ->** metodo che viene richiamato quando si prova a chiudere una finestra. Esso visualizza una ulteriore finestra di conferma dell'uscita ed, eventualmente, permette di non chiudere realmente la finestra.
- **void menuIniziale() ->** metodo che, come descritto prima, ha lo scopo di disattivare tutti i bottoni dei menu chiaramente non utilizzabili visto che non sono presenti ancora polinomi o non vi sono polinomi selezionati. Viene richiamato quando il metodo *counter()* restituisce 0
-

- **void menuOne() ->** metodo simile a *menuIniziale()* che però rende cliccabili quelle opzioni della barra del menu che possono essere eseguite su un solo polinomio (es. soluzione e derivata)
- **void menuTwo() ->** metodo simile a *menuIniziale()* che però rende cliccabili quelle opzioni della barra del menu che possono essere eseguite su due polinomi (es. derivata e addizione)
- **void menuMore() ->** metodo simile a *menuIniziale()* che però rende cliccabili quelle opzioni della barra del menu che possono essere eseguite su più polinomi (es. rimuovi e derivata)
- **void actionPerformed(ActionEvent e) ->** metodo che deriva dall'implementazione di ActionListener. Il metodo viene richiamato ogni volta che si interagisce con un componente della finestra a cui è stato aggiunto un ActionListener attraverso il metodo component.addActionListener(this). In base al componente cliccato, si rientra in uno dei casi definiti dagli if che, in base all'azione richiesta, richiamo un metodo dedicato. Nel caso in cui si si interagisca con dei componenti che a loro volta riguardano la gestione dei file (dunque componenti per il salvataggio o l'apertura di un file) allora viene operata una gestione try-catch che permette di evitare una interruzione del programma. Se infatti vi è una IOException, viene letto il blocco catch che richiama la costruzione e visualizzazione di una FinestraErrore che è un oggetto con lo scopo di visualizzare una finestra di errore con un messaggio personalizzato passato come argomento del costruttore.
- **void salvaFile() throws IOException ->** metodo che opera il salvataggio dei polinomi selezionati aprendo una finestra di dialogo dove scegliere il file di destinazione. Il salvataggio dei polinomi avviene controllando fra i polinomi quali sono stati selezionati; poiché essi sono gli unici componenti contenuti nel pannello *scroll*, è facile utilizzare la univocità degli indici per prelevare il corrispondente polinomio dalla LinkedList variabile d'istanza. I polinomi vengono scritti riga per riga attraverso un BufferedWriter. Viene inoltre gestita l'eccezione FileNotFoundException circondando tutto il codice con un blocco try-catch ed eventualmente viene generata una FinestraErrore come spiegato precedentemente.
- **void creaFile() throws IOException() ->** metodo che viene richiamato prima di *salvaFile()* ma dopo il metodo *nuovo()* se l'utente ha scelto di creare un nuovo file prima di effettuare il salvataggio. Il file viene creato se non ne esiste uno con lo stesso nome e percorso
- **void nuovo() ->** metodo che chiede all'utente, attraverso una finestra di dialogo, se vuole creare un nuovo file prima del salvataggio. In caso affermativo, viene visualizzata una finestra di tipo FinestraInput2 che permette l'inserimento del percorso del file e viene poi invocato *creaFile()*, altrimenti viene richiamato il metodo *salvaFile()*
- **void apriFile() throws IOException->** metodo che viene invocato quando si vuole caricare dei polinomi salvati precedentemente in un file. Legge i polinomi (che sono stati salvati ciascuno su una riga) attraverso un BufferedReader che verifica la correttezza di ciascuno attraverso il metodo divisore della classe Regex e li aggiunge alla schermata nel caso in cui sia corretto. Parte del codice è circondato da un blocco try-catch che permette di gestire il caso in cui il file cercato non esista, se si sta cercando di inserire dei polinomi ma il pannello ha superato il massimo contenibile

- (fissato per comodità a 64) o se nel file vengono trovati dei caratteri non riconducibili ad un polinomio e genera delle finestre di tipo FinestraErrore in tutti i casi
- **final class TxtFileFilter extends FileFilter->** classe privata e final che rende visibili solo cartelle e file di text (con estensione .txt) quando si prova a selezionare un file su cui salvare i polinomi nel metodo *salvaFile()*
- **void soluzione() ->** metodo che viene richiamato dopo il metodo *finestraSoluzione()* che chiede all'utente di inserire un valore quando si clicca sull'opzione di calcolo della soluzione di un polinomio. Esso genera una finestra di tipo FinestraOutput() su cui viene visualizzato il risultato ottenuto dall'invocazione della funzione *valore(int i)* sul polinomio selezionato. Nel caso in cui venga inserito un valore non numerico, grazie alla gestione tramite blocco try-catch viene catturata la NumberFormatException e viene generata una finestra di errore FinestraErrore
- **void addizione() ->** metodo che viene invocato quando si richiede il calcolo dell'addizione fra due polinomi. L'individuazione dei due polinomi selezionati avviene tramite un ciclo while e una variabile boolean di supporto in modo da evitare un meno efficiente doppio ciclo di for per analizzare i componenti. Alla fine, somma i due polinomi e aggiunge il risultato fra i precedenti polinomi, rendendo possibile l'esecuzione di operazioni su di esso.
- **int counter() ->** metodo che, come precedentemente detto, verifica quanti polinomi sono selezionati in un determinato istante e restituisce il numero
- **void derivazione() ->** metodo che viene invocato quando si vuole calcola la derivata di tutti i polinomi selezionati. Genera una finestra di errore se si vuole calcolare la derivata di una costante in quanto darebbe come risultato 0
- **void moltiplicazione() ->** metodo che viene invocato quando si vuole calcolare la moltiplicazione fra due polinomi. Come per il metodo *addizione()* viene utilizzato un ciclo while e una variabile booleana di supporto.
- **void rimuovi() ->** metodo che rimuove i polinomi selezionati dalla variabile d'istanza lista dei polinomi e anche dal pannello che li visualizza
- **void creaPolinomio() ->** metodo che viene richiamato ogni volta che viene inserito un polinomio nella barra in basso della schermata e si clicca il bottone di aggiunta. Esso verifica innanzitutto che la finestra non sia completa (cioè siano già presenti 64 polinomi), che la stringa inserita non sia vuota e che eventualmente il polinomio inserito sia corretto richiamando il metodo *divisore()* della classe Regex e gestisce tramite dei blocchi try-catch le eventuali eccezioni lanciate costruendo delle finestre di errore FinestraErrore
- **private class FinestraErrore extends JFrame ->** classe che genera un oggetto di tipo finestra attraverso il costruttore *FinestraErrore(String s)* che genera una finestra con il messaggio di errore passato come argomento e permette di usare una unica classe per generare dunque messaggi di diverso tipo
- **private void finestraSoluzione() ->** metodo che viene richiamato quando si vuole calcolare la soluzione ad un polinomio. Invoca a sua volta una finestra di tipo FinestraInput1 che permette di inserire il valore per cui si vuole calcolare la soluzione e richiama poi il metodo *soluzione()* indirettamente attraverso il bottone calcola visualizzato sulla finestra. E' il metodo *soluzione()* ad effettuare eventuali controlli sul valore inserito.

- **private class FinestraInput1 extends JFrame ->** classe che definisce un oggetto finestra che viene creato dal metodo *finestraSoluzione()* per inserire un valore da cui calcolare una soluzione per il polinomio selezionato
- **private class FinestraInput2 extends JFrame ->** metodo simile al precedente ma che viene invocato da *nuovo()* per generare una finestra in cui inserire il percorso di un file da creare nel caso in cui si voglia effettuare un salvataggio dei polinomi