CalcolProject

CALCOLATRICE AD ESPRESSIONI MULTINOTAZIONE

A cura di: Cristina Ochner, a.s. 2017-2018

SOMMARIO

INTRODUZIONE

ANALISI DEI REQUISITI

PROGETTAZIONE

SVILUPPO

TESTING

MANUTENZIONE

CONCLUSIONI

A cura di: Cristina Ochner, a.s. 2017-2018

Introduzione

STRUMENTI UTILIZZATI

- Sviluppo: Android Studio;
  - Java
  - Xml (interfaccia)
  - Test

- Versioning e gestione progetto:
  - Github

  - Github wiki e readme
    https://github.com/cristinaochner/CalculProject/wiki

  - Repository applicazione beta
    https://github.com/cristinaochner/LukasiewiczCalculator

  - Repository applicazione CalcolProject
    https://github.com/cristinaochner/CalculProject

  - Immagini presentazione:
    http://mshang.ca/syntree/

Introduzione

CHE COSA E' CALCOLPROJECT?

- Un'applicazione Android.

- Una calcolatrice multi-notazione.
  - Può gestire espressioni in notazione classica.
  - Può gestire espressioni in notazione polacca.
  - Può gestire espressioni in notazione RPN. (Polacca inversa)

- Può gestire espressioni intere, non un operatore alla volta come nelle classiche calcolatrici tascabili.

A cura di: Cristina Ochner, a.s. 2017-2018

Notazioni

CLASSICA – INFIX (Wiki)

- L'operatore è posto tra i due operandi;
- Necessita di parentesi in quanto gli operatori * e / hanno priorità di esecuzione;

CLA: (5 + 4) * 2 = 18

CLA: 5 + 4 * 2 = 13

Notazioni

POLACCA – PREFIX (Wiki)

- Gli operatori contenuti nella espressione sono collocati prima degli operandi.
- Non necessita di parentesi in quanto il modo in cui l'espressione viene inserita all'interno della calcolatrice non la rende equivoca;
- Permette di esprimere l'ordine delle operazioni.
- Esempi:

CLA: (5 + 4) * 2 = 18
POL: * + 5 4 2 = 18

CLA: 5 + 4 * 2 = 13
POL: + * 4 2 5 = 13

Notazioni

POLACCA INVERSA – POSTFIX – RPN (Wiki)

- E’ equivalente alla notazione polacca, ma l’ordine degli operatori è inverso, essi sono posti dopo gli operandi;
- Anch’essa non necessita di parentesi in quanto il modo in cui l’espressione viene inserita all’interno della calcolatrice non la rende equivoca;
- Permette di esprimere l’ordine delle operazioni.
- Esempi:

CLA: (5 + 4) * 2 = 18
RPN: 2 4 5 + * = 18

CLA: 5 + 4 * 2 = 13
RPN: 5 2 4 * + = 13

A cura di: Cristina Ochner, a.s. 2017-2018

Analisi Requisiti

OBIETTIVI DEL PROGETTO

- Realizzare un’applicazione ad alta usabilità.
- Che permetta l’accesso veloce alla calcolatrice.
- Con la possibilità di scegliere la notazione preferita in cui definire le espressioni.

Analisi Requisiti

VINCOLI DI PROGETTO

- L’applicazione deve funzionare sui sistemi Android.

- L’applicazione deve utilizzare come strumento di input un tastierino numerico customizzato.

Analisi Requisiti

REQUISITI

- L'applicazione deve permettere la scelta della notazione da adottare tra : Classica(infix), Polacca e Polacca Inversa.
- L'utente deve poter essere libero di utilizzare anche calcolatrici, la cui notazione a lui è sconosciuta, quindi deve essere integrato un manuale per ogni notazione.
- La calcolatrice deve supportare gli operatori : +, -, *, / .
- L'applicazione deve poter dare la possibilità all'utente di cancellare l'intero campo di immissione o un singolo carattere.
- L'utente deve poter inserire un'espressione completa che viene valutata complessivamente.

A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

ANDROID ACTIVITIES

- In Android le applicazioni sono strutturate in attività (activity).
- Le attività rappresentano una vista dell'applicazione.
- L'interfaccia è definita attraverso un documento di tipo xml che ne descrive i vari componenti.
- Il comportamento dei componenti e delle activity è definito all'interno di file Java.
- E' possibile definire transizioni tra le attività attraverso la creazione di Intent nel codice Java.

A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

INTERFACCIA: ATTIVITA' INIZIALE (Wiki)

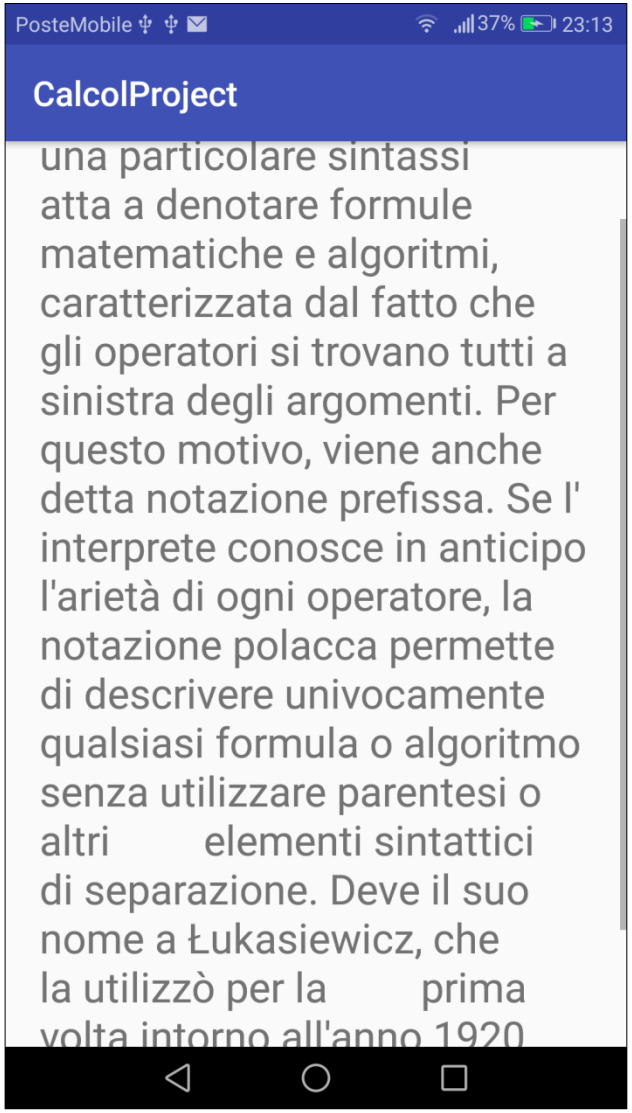
- Si può accedere all'attività di calcolatrice, specificando quale notazione utilizzare.
- Si può accedere all'attività di "Aiuto", in cui vengono descritte le funzionalità delle calcolatrici.

A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

INTERFACCIA: ATTIVITA' AIUTO (Wiki)

- Visualizza in modalità testuale informazioni sulla notazione di calcolo scelta.
- Fornisce un esempio di come inserire l'espressione nel calcolatore.



A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

INTERFACCIA: ATTIVITA' CALCOLATRICE (Wiki)

- Due campi testuali:
- Il primo visualizza l'espressione inserita dall'utente.
- Il secondo visualizza il risultato dell'espressione.

- Tastiera di input personalizzata:
- Numeri ed operatori gestiti.
- Tasti di cancellazione: elemento e reset.
- Tasto **Enter** e **Uguale** per l'input di numeri successivi e l'esecuzione dell'espressione.

A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

ESPERIENZA UTENTE

- Per migliorare l'esperienza utente si è deciso di minimizzare le transizioni tra le activity.
- È stata eliminata l'attività introduttiva di spiegazione, spostata nelle attività di aiuto.
- La scelta iniziale della notazione può essere evitata ricordando l'ultima scelta effettuata dall'utente.
- Il tastierino numerico custom ammette l'input dei soli caratteri ammissibili ottimizzando lo spazio e la funzionalità.
- La visualizzazione dell'espressione inserita dipende dal significato sintattico della stessa e appare più ordinata.

Progettazione

REQUISITI SOFTWARE

- E’ stata individuata la necessità di programmare un modulo che possa generare un’espressione dall'input dell’ utente ed eseguirla. Per questo sono stati individuati due componenti principali:
  - La Classe Divisore:
  - Prende l’input dal tastierino numerico.
  - Genera un array di elementi coerenti.
  - Fornisce la Stringa di visualizzazione dell’espressione.

- La Classe Espressione (e sottoclassi):
  - Prende array di elementi coerenti e lo trasforma in un’istanza di espressione.
  - Sull’istanza può essere chiamato il metodo esegui() che ricorsivamente calcola il valore dell’espressione.

A cura di: Cristina Ochner, a.s. 2017-2018

Progettazione

ARCHITETTURA DI SISTEMA

Attività

- Attività Principale
- Attività Calcolatrice
- Attività Aiuto

Divisore

- String toString()
- numero(String s)
- operatore(String s)
- invio()
- Double esegui()

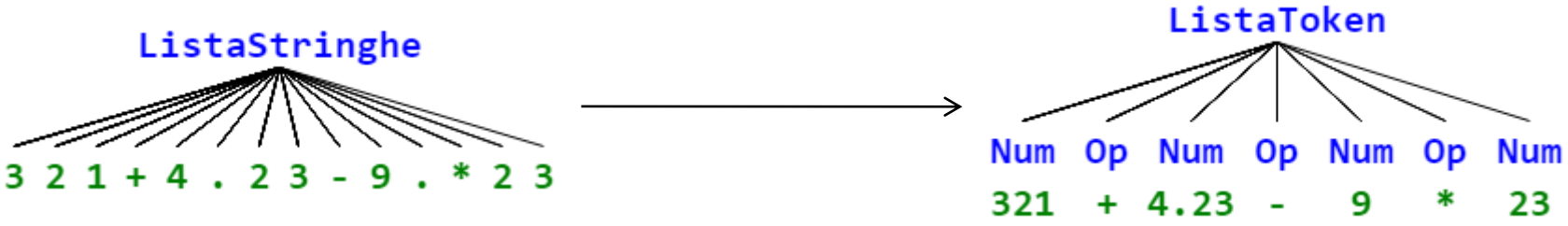
Espressione

- Expression creaCla (ArrayList<String>)
- Expression creaRPN (ArrayList<String>)
- Expression CreaPol (ArrayList<String>)
- Double esegui()

Sviluppo

CLASSE DIVISORE (Wiki)

- La Classe Divisore:
  - Prende l'input dal tastierino numerico.
  - Genera un array di elementi coerenti.
  - Fornisce la Stringa di visualizzazione dell'espressione che verrà visualizzata sullo schermo.

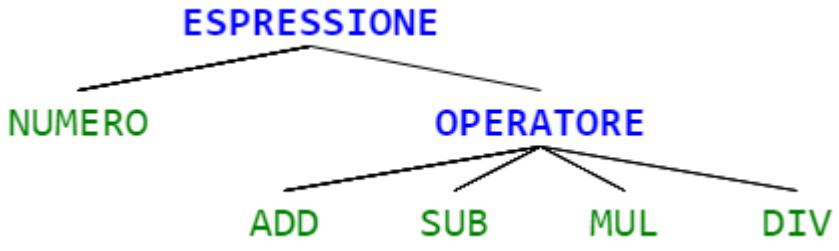


A cura di: Cristina Ochner, a.s. 2017-2018

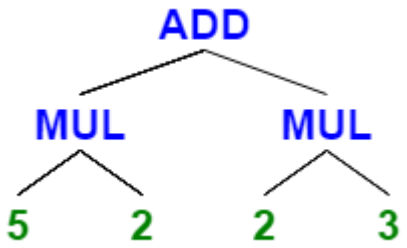
Sviluppo

CLASSE ESPRESSIONE (Wiki)

- La Classe Espressione (e sottoclassi) :
  - Trasforma l'array di elementi in un albero di espressioni.
  - Un metodo diverso per ogni notazione.
  - Sull'istanza può essere chiamato il metodo esegui() che ricorsivamente calcola il valore dell'espressione.
  - Può rappresentare qualsiasi espressione a prescindere dalla notazione.
  - Gerarchia classi espressione:



5 * 2 + 2 * 3



A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESEMPIO VALUTAZIONE ESPRESSIONE

- L'utente vuole calcolare il valore dell'espressione:
  - **CLA:** `1 + 2 * 3 – 4 / 5 * 6 + 7`
  - **RPN:** `1 2 3 * + 4 5 6 * / - 7 +`

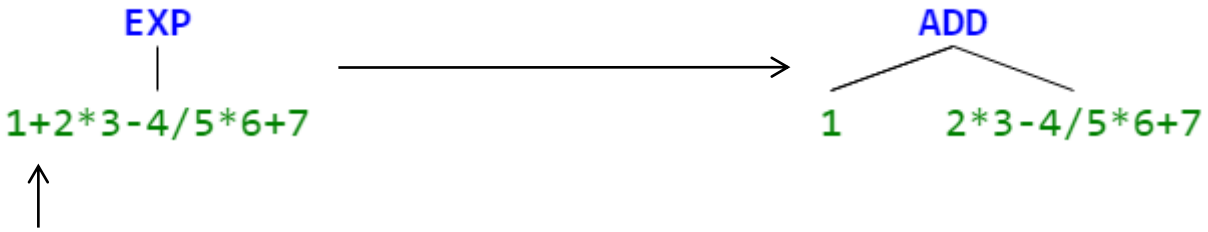
- Per la notazione classica l'utente premerà i tasti:
  `[1][+][2][*][3][-][4][/][5][*][6][+][7][=]`

- Il Divisore creerà un equivalente array di elementi:
  `[1] [+] [2] [*] [3] [-] [4] [/] [5] [*] [6] [+] [7]`

Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA

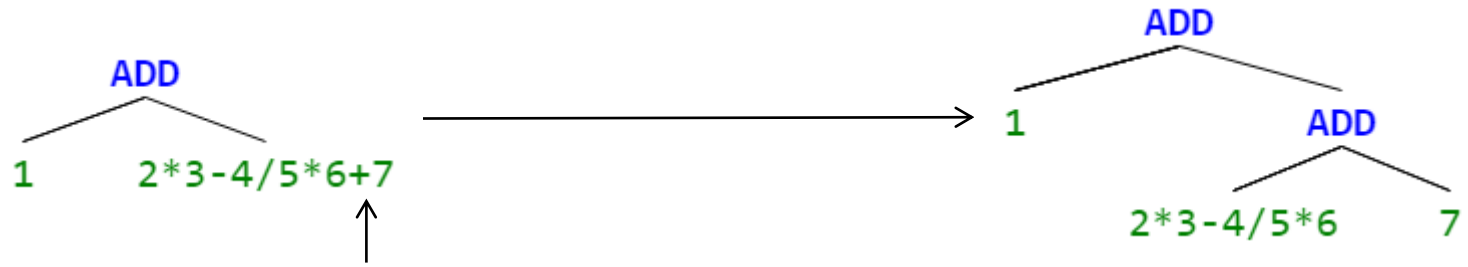
- Viene individuato l'indice del primo operatore con priorità (+ - * /).
- Si creano 2 sotto-array con la parte a sx e dx dell'operatore.
- Si applica ricorsivamente la procedura ai sotto array.
- Viene creato un oggetto somma con i risultati delle 2 chiamate.



Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA

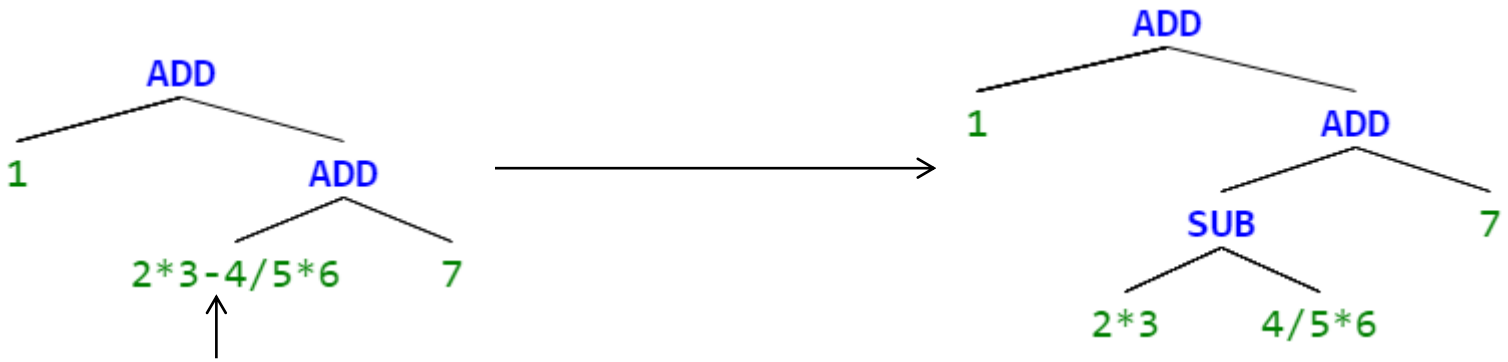
- Viene eseguita la procedura ricorsiva sui due sotto-array.
- L'array che contiene il singolo elemento "1" diventa un espressione di tipo numero.
- L'array più complesso viene suddiviso nuovamente cercando ancora il primo operatore con priorità (+ - * /)



Sviluppo

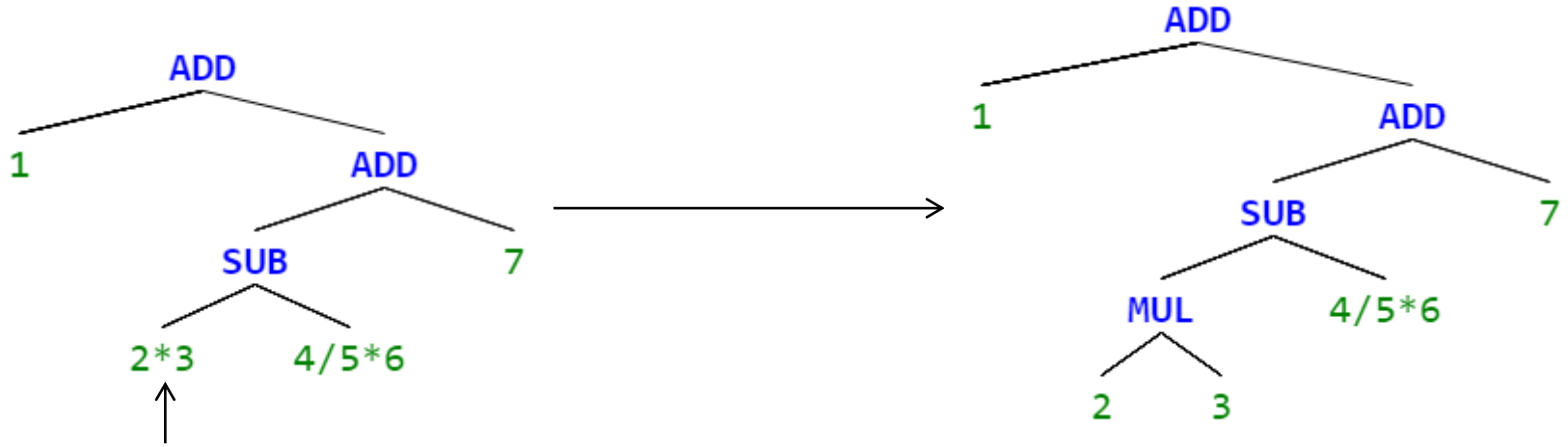
ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA

- Il procedimento continua ricorsivamente.
- Non ci sono più operatori + quindi si passa agli operatori con priorità inferiore (-).



Sviluppo

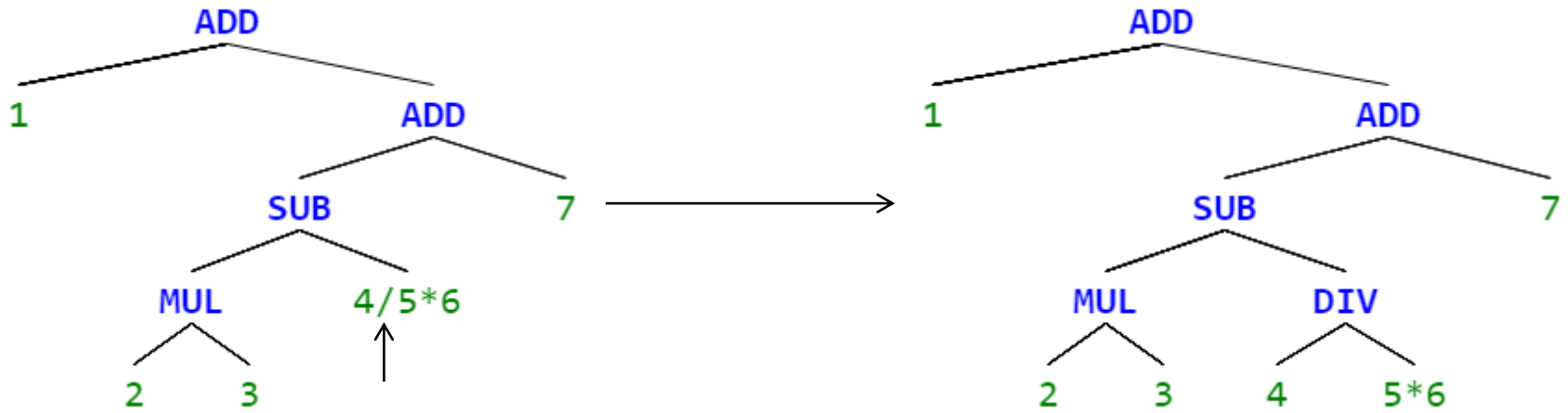
ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA



A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

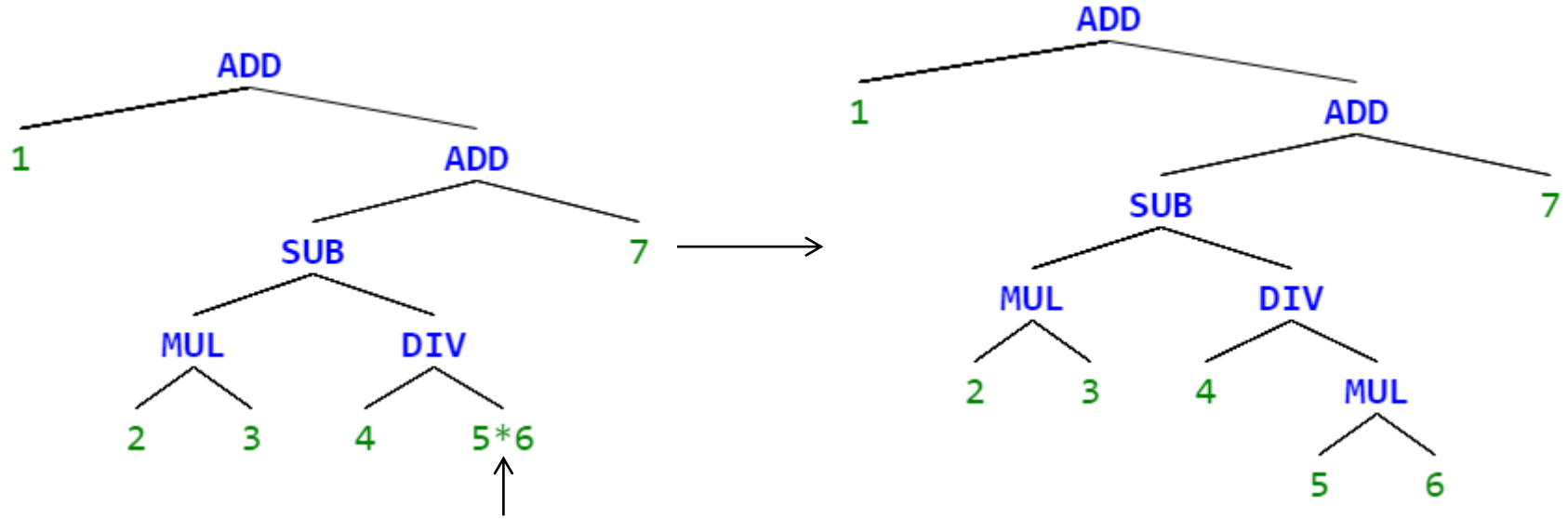
ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA



Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE CLASSICA

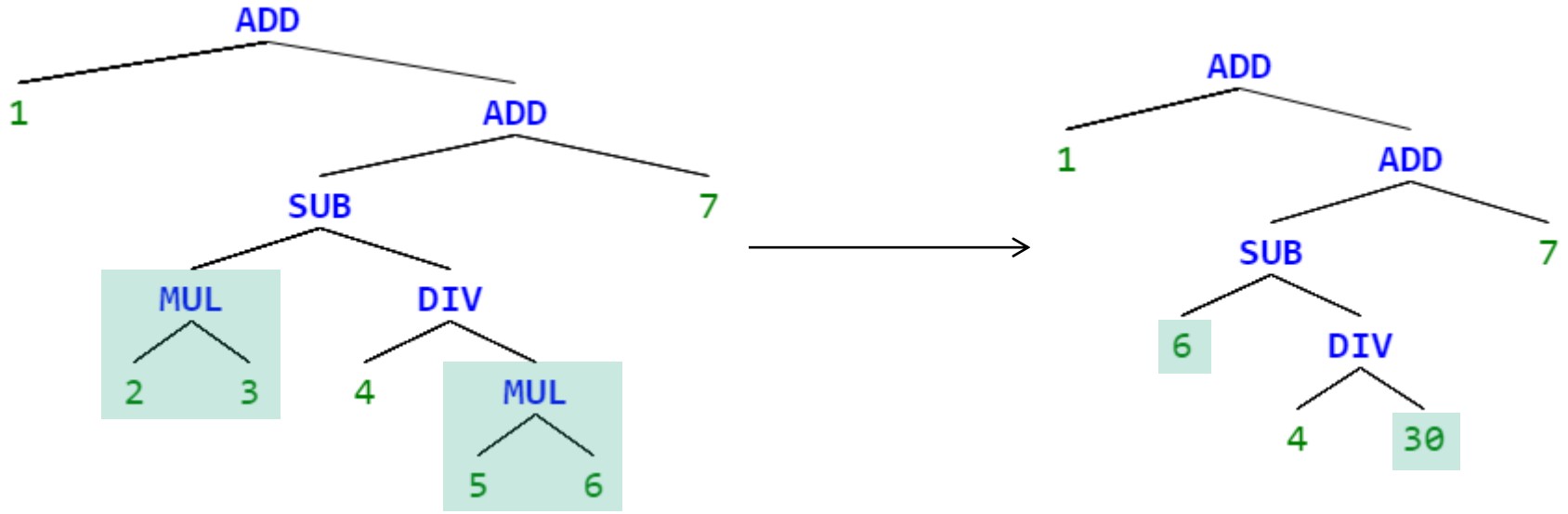
- La creazione dell'espressione è completa.
- Il metodo ritorna la prima espressione padre (ADD).
- Sulla quale può essere chiamato il metodo esegui() per ottenerne il risultato.



Sviluppo

ESECUZIONE

- La chiamata **esegui()** è ricorsiva, ogni operatore chiama la **esegui()** sui suoi operandi.
- Gli operatori che hanno operandi di tipo **Numero** possono eseguire l'operazione e quindi ritornare il risultato.



A cura di: Cristina Ochner, a.s. 2017-2018

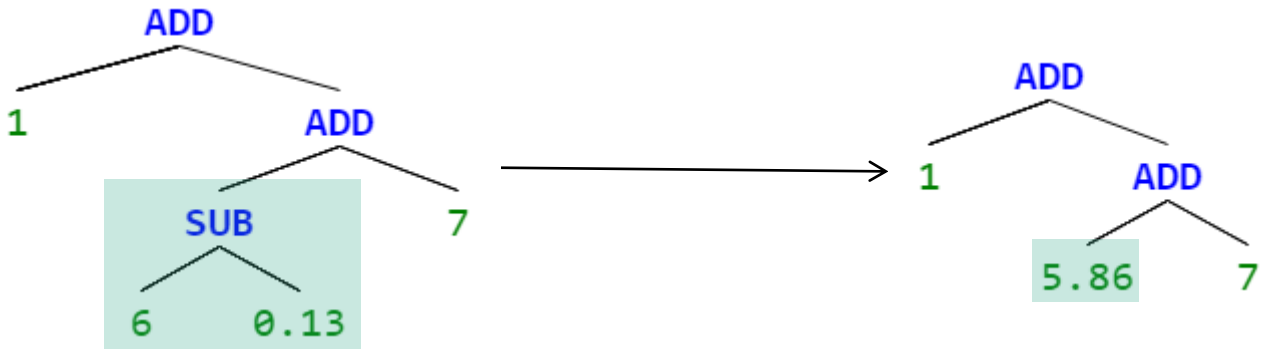
Sviluppo

ESECUZIONE

A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

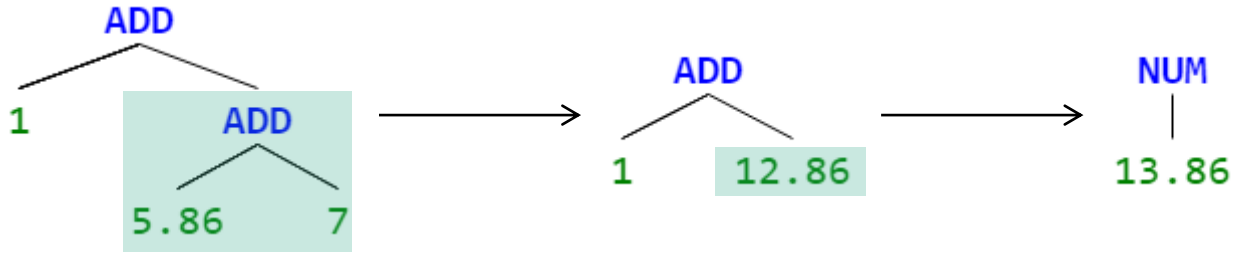
ESECUZIONE



A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESECUZIONE



A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

VALUTAZIONE ESPRESSIONE RPN

- L'utente vuole calcolare il valore dell'espressione:
  - **CLA:** 1 + 2 * 3 – 4 / 5 * 6 + 7
  - **RPN:** 1 2 3 * + 4 5 6 * / - 7 +

- Per la notazione RPN l'utente premerà i tasti:
  [1][E][2][E][3][*][+][4][E][5][E][6][*][/][-][7][+]
  E' necessario premere [E] per separare i numeri.

- Il Divisore creerà un array di elementi:
  [1] [2] [3] [*] [+] [4] [5] [6] [*] [/] [-] [7] [+]

A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

- La generazione dell'espressione a partire dalla notazione RPN è basata sull'utilizzo di uno stack.
- L'algoritmo è estremamente semplice:

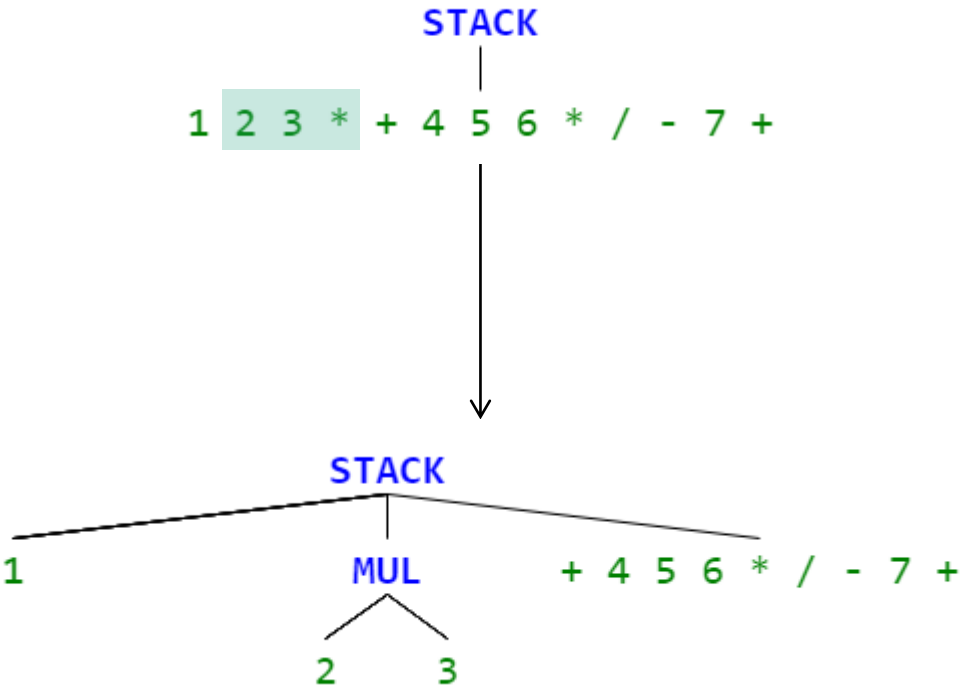
- Si cicla sull'array di elementi (numeri e operatori)
- Se si trova un numero lo si inserisce nello stack.
- Se si trova un operatore:
    - Si tolgono gli ultimi due elementi dallo stack.
    - Si inserisce una nuova istanza dell'operatore sullo stack con i due elementi come operandi.

- Alla fine, se l'espressione è valida, si otterrà sullo stack una singola espressione (Operatore o Numero) che potrà essere eseguita.

Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

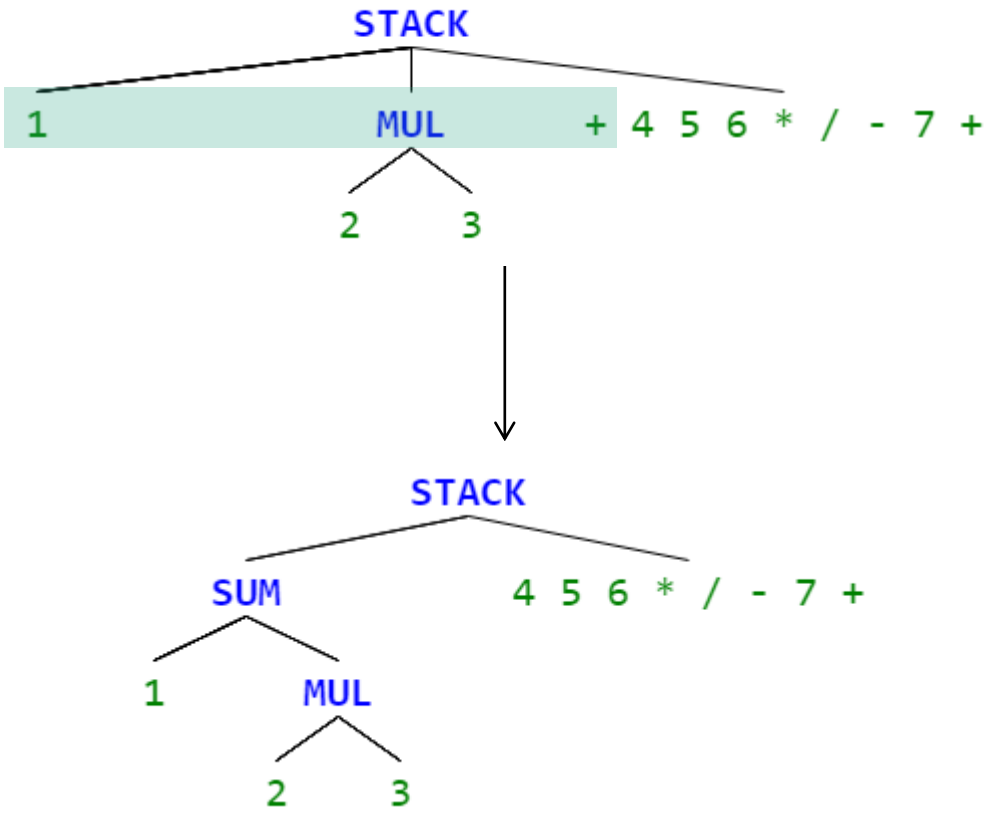
- Si inserisono sullo stack gli elementi [1][2][3] in quanto numeri.
- Il quarto elemento è una moltiplicazione quindi si rimuovono gli ultimi 2 elementi,
- E si inserisce l'istanza della classe con i due elementi come operandi.



Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

- Troviamo subito l'operatore [+]
- Si rimuovono gli ultimi 2 elementi dallo stack [1] e [MUL]
- Si aggiunge allo stack un nuovo oggetto [SUM] con operatori i due elementi.

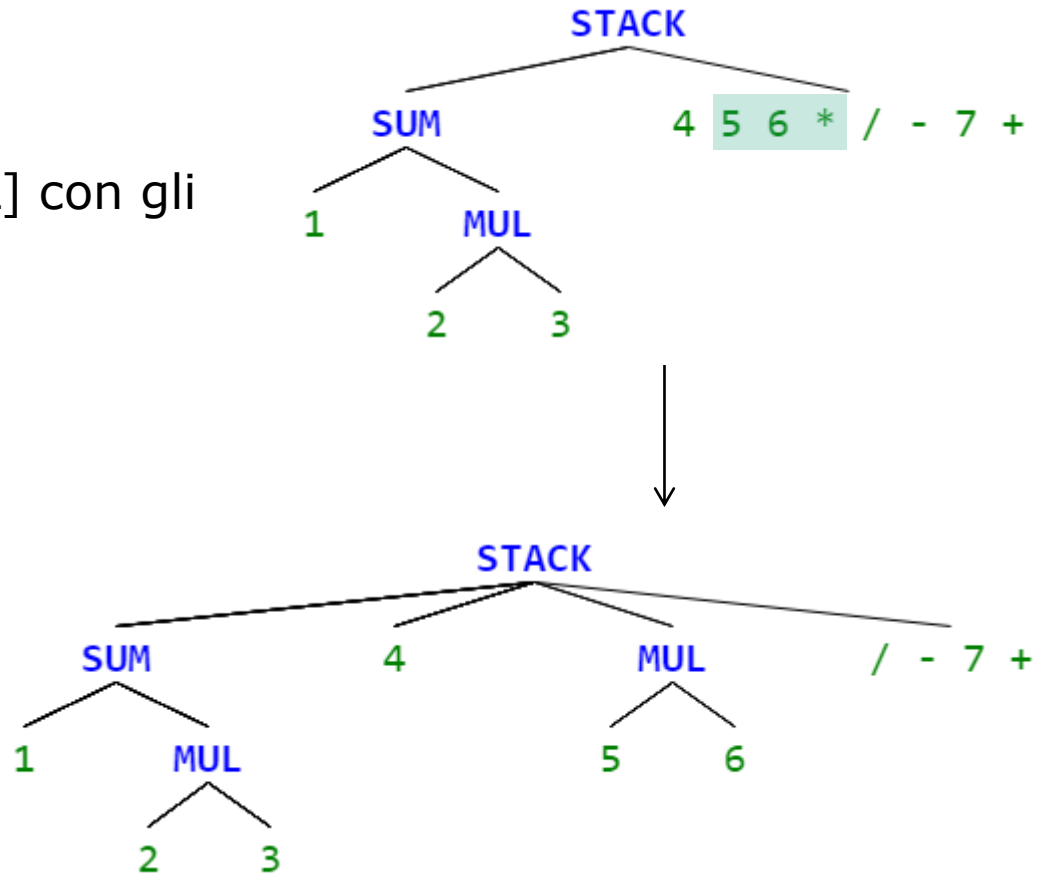


A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

- Si inserisce [4][5][6]
- Si Crea un oggetto [MUL] con gli oggetti [5] e [6]

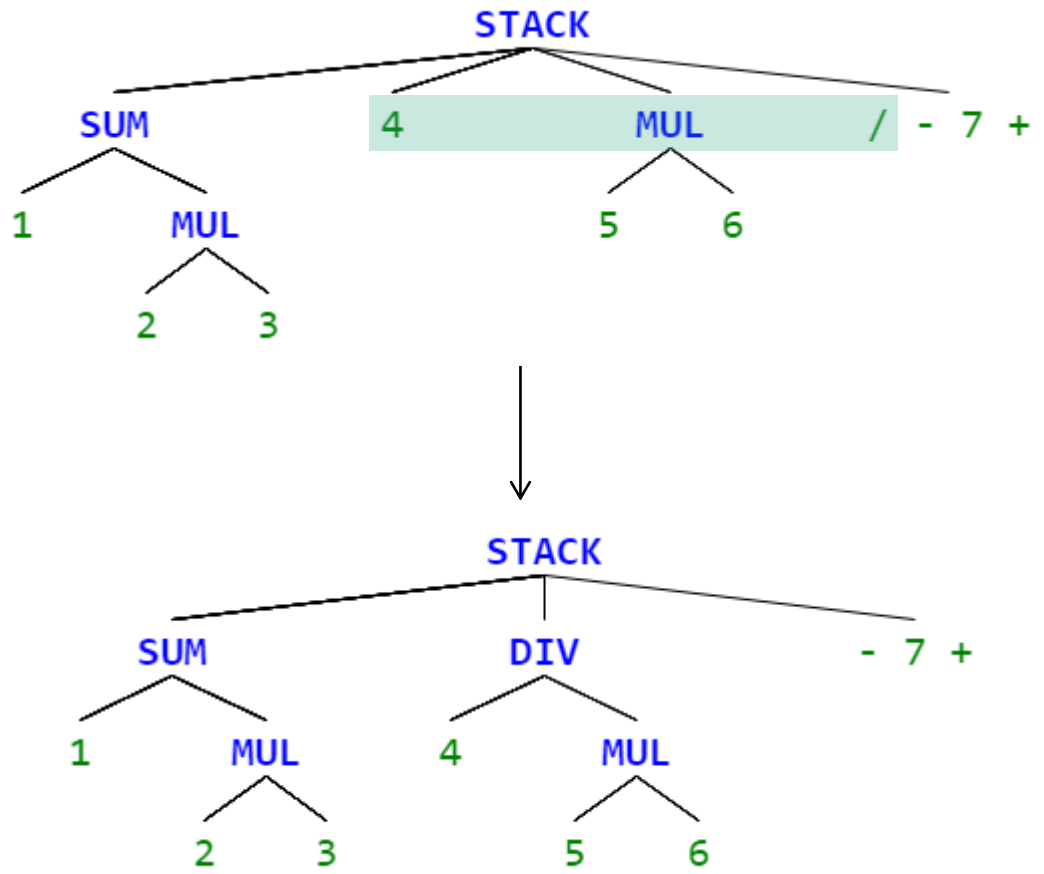


A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

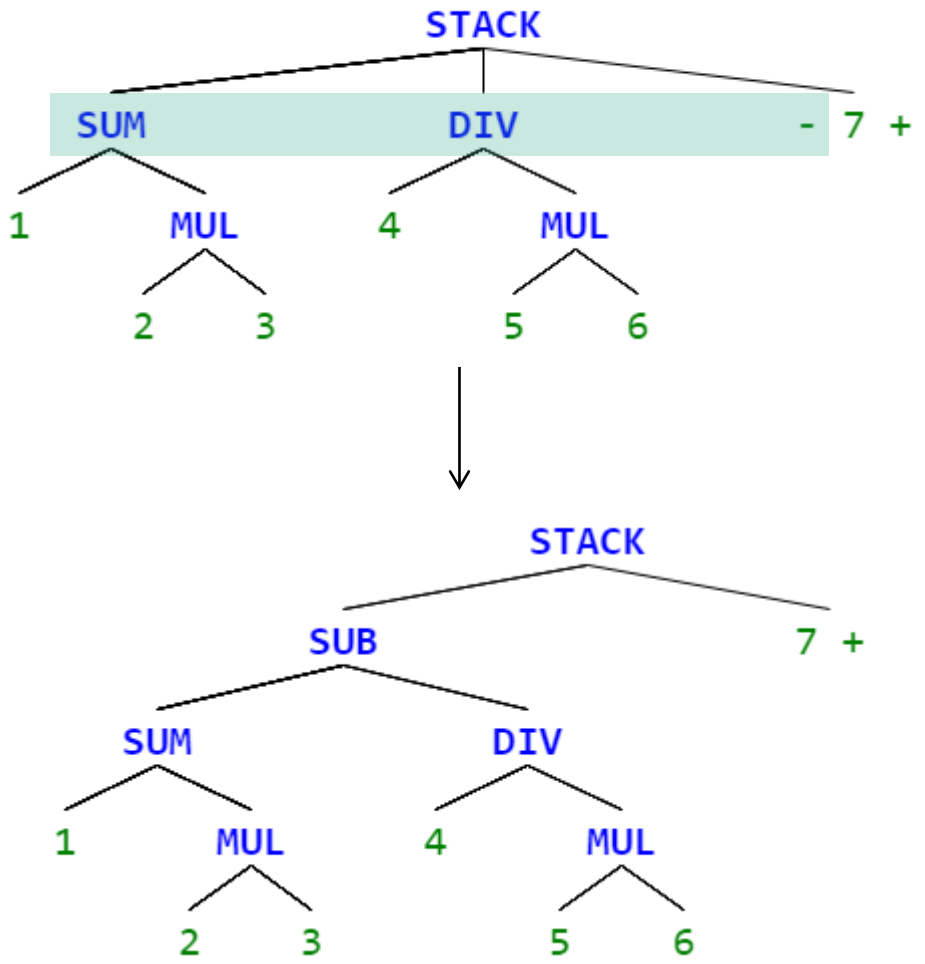
- Si Crea un oggetto [DIV] con gli oggetti [4] e [MUL]



Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

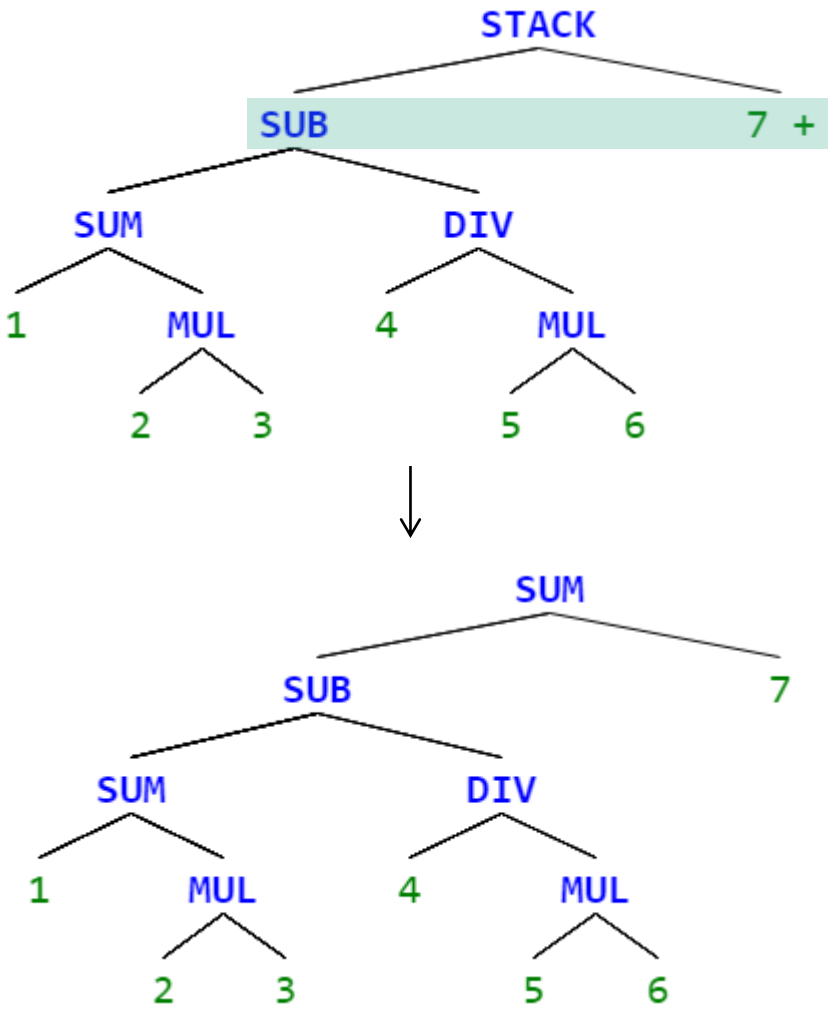
- Si trova un operatore quindi vengono tolti due elementi dallo stack.
- A questo punto lo stack è essenzialmente vuoto.
- Viene aggiunto un oggetto [SUB] con i due elementi come figli.



Sviluppo

ESEMPIO GENERAZIONE ESPRESSIONE NOTAZIONE RPN

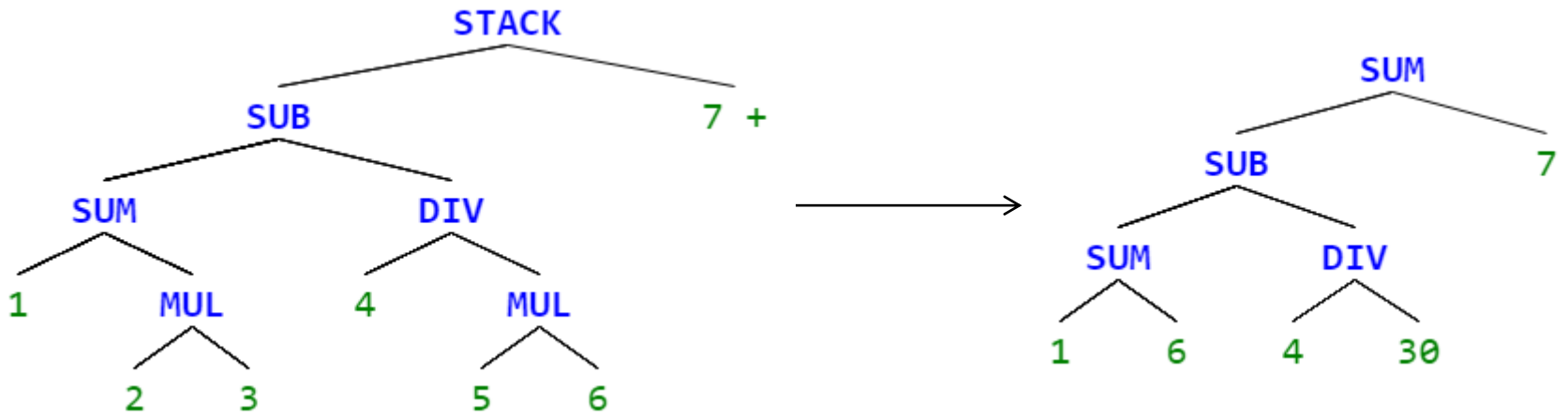
- Si trova un operatore quindi vengono tolti due elementi dallo stack.
- A questo punto lo stack è essenzialmente vuoto.
- Viene aggiunto un oggetto [SUB] con i due elementi come figli.
- Sullo stack è presente il singolo elemento [SUM] che è il risultato.
- **Espressione originale:**
  1 2 3 * + 4 5 6 * / - 7 +



Sviluppo

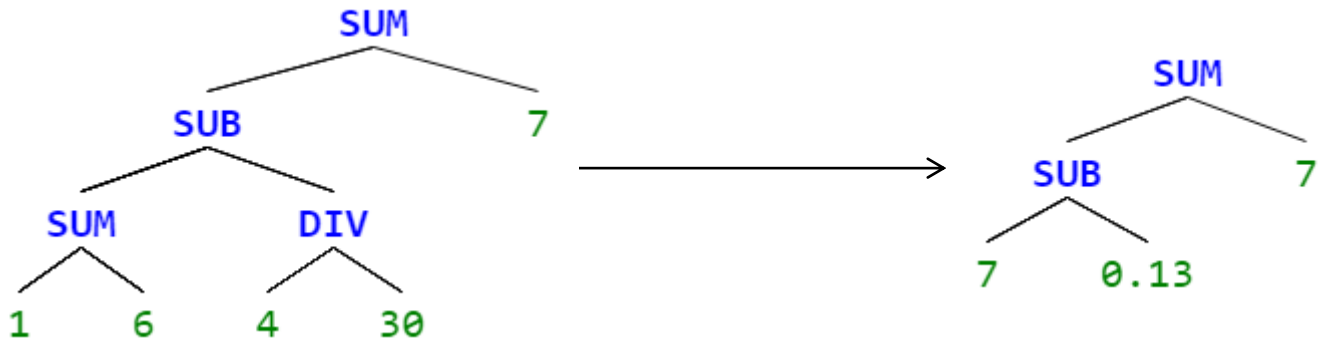
ESECUZIONE

- L'esecuzione dell'espressione per il calcolo del risultato è uguale a prescindere da quale notazione sia stata usata per la costruzione.



Sviluppo

ESECUZIONE

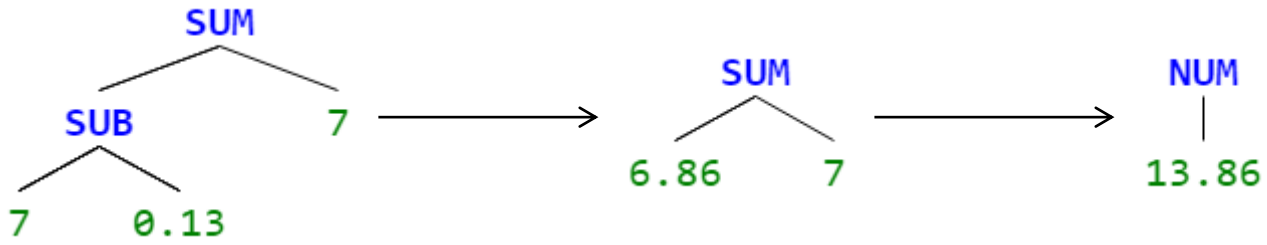


A cura di: Cristina Ochner, a.s. 2017-2018

Sviluppo

ESECUZIONE

- Il risultato delle due espressioni è equivalente.
  - **CLA:** `1 + 2 * 3 – 4 / 5 * 6 + 7`
  - **RPN:** `1 2 3 * + 4 5 6 * / - 7 +`



- L’esecuzione dell’espressione RPN potrebbe avvenire ‘al volo’ senza dover generare tutto l’albero dell’espressione.
- Tuttavia per riutilizzare la stessa interfaccia tra le varie classi si è preferito procedere in questo modo.

Testing

Unitest e Betatesting

- Sono stati creati degli Unit Test per verificare il funzionamento di alcuni moduli del software, in particolare:
- Nella classe Divisore sono stati testati I possibili input ed ouput della stessa.
- Nella classe Espressione è stato verificato che le espressioni vengano eseguite correttamente.
- L'applicazione, inoltre, è stata fatta provare ad alcuni betatester che hanno contribuito producendo dei report di problemi nell'Issue Tracker.

A cura di: Cristina Ochner, a.s. 2017-2018

Manutenzione

CORRETTIVA, ADATTATIVA ED EVOLUTIVA

- Modularità del Software:
  - Il software è stato scritto in modo modulare per poter essere riutilizzato in altri progetti java, in particolare la classe Expression e le sue sottoclassi sono pensate in tale modo.

- Licenza Open Source:
  - Il codice è rilasciato su GitHub con licenza GNU v3.0. La scelta è ricaduta su questa licenza in ottica di mantenimento dell'applicazione, perché forza la pubblicazione sotto stessa licenza di qualsiasi modifica effettuata da terzi garantendo così pubblico accesso alle correzioni e miglioramenti del codice.

A cura di: Cristina Ochner, a.s. 2017-2018

Manutenzione

CORRETTIVA, ADATTATIVA ED EVOLUTIVA

- Bug Tracking (GitHub)
  - Utilizzato il servizio pubblico di bug tracking fornito da GitHub che consente di tenere traccia di problemi dell'applicazione.

- Wiki Open Source
  - E' stata creata una Wiki Open in modo di poter fornire uno strumento di supporto a utenti e sviluppatori, dando la possibilità agli stessi di contribuire alla creazione di ulteriore documentazione.

Conclusioni

POSSIBILI SVILUPPI FUTURI

- Supporto delle parentesi nel calcolatore in notazione classica.
- Estensione calcolatrice a numerazione binaria ed esadecimale.
- Possibilità di trasformare un’espressione scritta in una data notazione in un’altra.

A cura di: Cristina Ochner, a.s. 2017-2018

Alternanza Scuola Lavoro

E STAGE VARI ☺

- Stage Terzo anno:
  4 settimane presso Istituto Comprensivo Pergine 1:
  - Sviluppo sito web d'Istituto
  - Esperienza con HTML e Javascript
  - Joomla

- Alternanza scuola lavoro Quinto anno:
  (100 ore) presso Vetri Speciali S.P.A. :
  - Gestione DB fornitori
  - Esperienza con Issue Tracker

A cura di: Cristina Ochner, a.s. 2017-2018

Alternanza Scuola Lavoro

E STAGE VARI ☺

Istituto ▾ Uffici ▾ Sedi ▾ Personale ▾ Genitori ▾ Alunni ▾ Didattica ▾ Documenti ▾ Pubblicità Legale Amm.Trasparente ▾

CHIUDI INFO

Docenti

- Comunicazioni e Circolari
- Corsi d'aggiornamento
- Modulistica
- Consigli di Classe
- Contratto Decentrato d'Istituto- Docenti (anno 2010 ancora in validità)
- Registro Elettronico
- Area Riservata

Benvenuto nella sezione

Docenti

dell'Istituto Comprensivo Pergine 1.

Qui puoi trovare informazioni, documenti, modulistica, informazioni e l'accesso alla tua area riservata

iCagenda - Calendar

Giugno 2018

Lun	Mar	Mer	Gio	Ven	Sab	Dom
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

BIBLIOGRAFIA

- "Ciclo di vita di un prodotto o servizio" G.Luchetti, 2017
- "Compact recursive-descent parsing of expressions" C.Keith, 1992
- "About Notations" http://www.cs.man.ac.uk/~pjj/cs212/fix.html Manchester University, 2013
- "Translation to and from Polish notation" C.L.Hamblin, 1962
- "Stack Machines: RPN calculator" A.Hayes, 2015

A cura di: Cristina Ochner, a.s. 2017-2018

GRAZIE
DELL'ATTENZIONE!

RINGRAZIAMENTI

- A mio fratello Francesco che come me sta sostenendo l'esame di maturità, che questo sia solo il primo dei grandi traguardi che otterrai nella vita con impegno e il tuo solito contagioso sorriso.

- Grazie a Federico, per i consigli e l'aiuto nello sviluppo di questo progetto, ma soprattutto per essermi sempre accanto, if dopo if, giorno dopo giorno. Grazie per essere quella parte di famiglia che mi sono scelta e che insieme scegliamo ogni giorno..
  <3

A cura di: Cristina Ochner, a.s. 2017-2018