Università degli Studi di Salerno
Esame di Ingegneria del Software

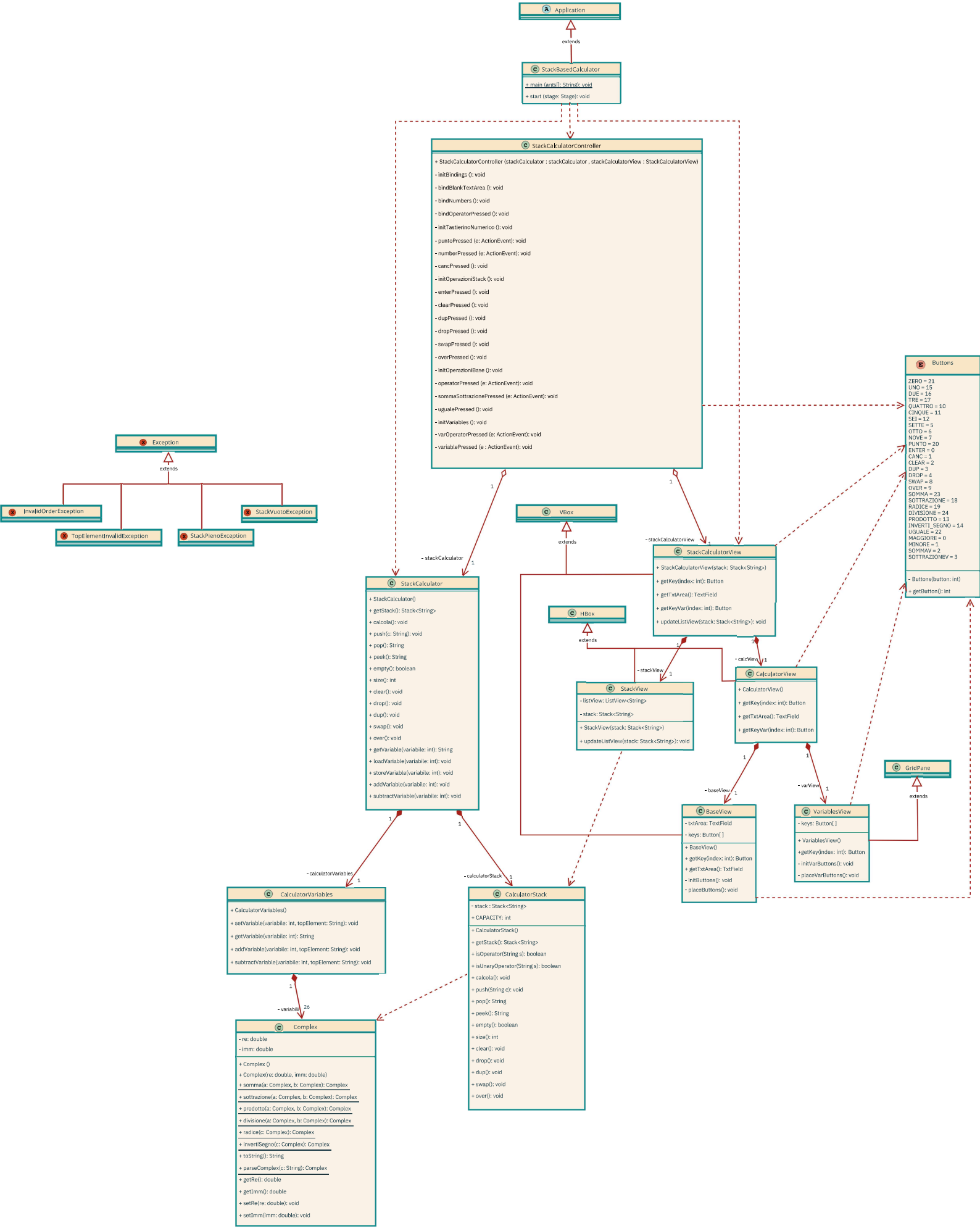
CLASS DIAGRAM

CALCOLATRICE PROGRAMMABILE BASATA SU STACK

2023/2024

Prodotto dal Gruppo 14

Apicella Antonio
Celano Benedetta Pia
Cirillo Francesco Pio
Fasolino Alessandra


Application
extends
StackBasedCalculator
+ main (args[]: String): void
+ start (stage: Stage): void
StackCalculatorController
+ StackCalculatorController (stackCalculator : stackCalculator , stackCalculatorView : StackCalculatorView)
- initBindings (): void
- bindBlankTextArea (): void
- bindNumbers (): void
- bindOperatorPressed (): void
- initTastierinoNumerico (): void
- puntoPressed (e: ActionEvent): void
- numberPressed (e: ActionEvent): void
- cancPressed (): void
- initOperazioniStack (): void
- enterPressed (): void
- clearPressed (): void
- dupPressed (): void
- dropPressed (): void
- swapPressed (): void
- overPressed (): void
- initOperazioniBase (): void
- operatorPressed (e: ActionEvent): void
- sommaSottrazionePressed (e: ActionEvent): void
- ugualePressed (): void
- initVariables (): void
- varOperatorPressed (e: ActionEvent): void
- variablePressed (e : ActionEvent): void
Buttons
ZERO = 21
UNO = 15
DUE = 16
TRE = 17
QUATTRO = 10
CINQUE = 11
SEI = 12
SETTE = 5
OTTO = 6
NOVE = 7
PUNTO = 20
ENTER = 0
CANC = 1
CLEAR = 2
DUP = 3
DROP = 4
SWAP = 8
OVER = 9
SOMMA = 23
SOTTRAZIONE = 18
RADICE = 19
DIVISIONE = 24
PRODOTTO = 13
INVERTI_SEGNO = 14
UGUALE = 22
MAGGIORE = 0
MINORE = 1
SOMMAV = 2
SOTTRAZIONEV = 3
- Buttons(button: int)
+ getButton(): int
Exception
extends
InvalidOrderException
TopElementInvalidException
StackPienoException
StackVuotoException
VBox
extends
- stackCalculator
- stackCalculatorView
StackCalculatorView
+ StackCalculatorView(stack: Stack<String>)
+ getKey(index: int): Button
+ getTxtArea(): TextField
+ getKeyVar(index: int): Button
+ updateListView(stack: Stack<String>): void
StackCalculator
+ StackCalculator()
+ getStack(): Stack<String>
+ calcola(): void
+ push(c: String): void
+ pop(): String
+ peek(): String
+ empty(): boolean
+ size(): int
+ clear(): void
+ drop(): void
+ dup(): void
+ swap(): void
+ over(): void
+ getVariable(variabile: int): String
+ loadVariable(variabile: int): void
+ storeVariable(variabile: int): void
+ addVariable(variabile: int): void
+ subtractVariable(variabile: int): void
HBox
extends
- stackView
- calcView
StackView
- listView: ListView<String>
- stack: Stack<String>
+ StackView(stack: Stack<String>)
+ updateListView(stack: Stack<String>): void
CalculatorView
+ CalculatorView()
+ getKey(index: int): Button
+ getTxtArea(): TextField
+ getKeyVar(index: int): Button
GridPane
extends
- baseView
- varView
BaseView
- txtArea: TextField
- keys: Button[ ]
+ BaseView()
+ getKey(index: int): Button
+ getTxtArea(): TxtField
- initButtons(): void
- placeButtons(): void
VariablesView
- keys: Button[ ]
+ VariablesView()
+getKey(index: int): Button
- initVarButtons(): void
- placeVarButtons(): void
- calculatorVariables
- calculatorStack
CalculatorVariables
+ CalculatorVariables()
+ setVariable(variabile: int, topElement: String): void
+ getVariable(variabile: int): String
+ addVariable(variabile: int, topElement: String): void
+ subtractVariable(variabile: int, topElement: String): void
CalculatorStack
- stack : Stack<String>
+ CAPACITY: int
+ CalculatorStack()
+ getStack(): Stack<String>
+ isOperator(String s): boolean
+ isUnaryOperator(String s): boolean
+ calcola(): void
+ push(String c): void
+ pop(): String
+ peek(): String
+ empty(): boolean
+ size(): int
+ clear(): void
+ drop(): void
+ dup(): void
+ swap(): void
+ over(): void
- variabili
Complex
- re: double
- imm: double
+ Complex ()
+ Complex(re: double, imm: double)
+ somma(a: Complex, b: Complex): Complex
+ sottrazione(a: Complex, b: Complex): Complex
+ prodotto(a: Complex, b: Complex): Complex
+ divisione(a: Complex, b: Complex): Complex
+ radice(c: Complex): Complex
+ invertiSegno(c: Complex): Complex
+ toString(): String
+ parseComplex(c: String): Complex
+ getRe(): double
+ getImm(): double
+ setRe(re: double): void
+ setImm(imm: double): void

Class Diagram Description

Main

Il Class Diagram in questione descrive la struttura dell'applicativo in grande dettaglio.

In cima al diagramma in conseguenza della scelta di utilizzare JavaFX vi è la classe StackBasedCalculator che ospita l'entry point dell'applicazione e che estende Application facendo di conseguenza l'Override del metodo start.

MVC

StackBasedCalculator dipende poi dalle tre classi più importanti dell'applicativo: StackCalculator, StackCalculatorView e StackCalculatorController che ricoprono il ruolo rispettivamente di Model, View e Controller, in accordo alle scelte progettuali fatte.

Il Controller

Il Controller è un Aggregato i cui Aggreganti sono un'istanza del Model e un'istanza della View; si nota che il Controller necessita dei sui aggreganti al fine di esplicitare la sua funzione: la coordinazione di Model e View.

Il Model

StackCalculator in quanto Model incapsula i dati, che nel caso dell'applicativo in questione sono un array di ventisei (26) variabili di tipo Complex e uno stack di stringhe, e fornisce metodi per l'interazione con i dati stessi. Si nota che al fine di aumentare la coesione delle classi e diminuire la quantità di responsabilità assegnate ad ognuna si è deciso di modellare l'array di variabili e lo stack in due classi diverse, questa scelta implementativa aumenta anche il grado di riutilizzabilità delle classi.

La divisione dello stack e dell'array ha come conseguenza il fatto che StackCalculator (il Model) è un Composto i cui Componenti sono CalculatorStack (che modella lo stack e i metodi associati) e CalculatorVariables (che modella l'array di variabili e i metodi associati). Il Composto deve quindi solo esporre i metodi per effettuare tutte le operazioni relative al Model ma per espletarle si affida poi a chiamate ai metodi dei suoi due Componenti che sono l'uno all'oscuro dell'esistenza dell'altro e che sono coordinati da StackCalculator per tutte le operazioni che richiedono sia lo stack sia l'array di variabili.

La Classe Complex

Interessante da considerare è il ruolo della classe Complex. CalculatorVariables contiene un array di Complex, questo è esplicitato nel diagramma da una relazione di composizione da 1 a 26 da CalculatorVariables a Complex. I metodi di Complex sono inoltre fondamentali per lo stack, che deve effettuare le operazioni sui numeri complessi, da cui la relazione di dipendenza che lega CalculatorStack a Complex.

La View

Per quanto riguarda la View per semplificare l'implementazione si è deciso di renderla composta da delle sotto-View, nello specifico StackCalculatorView è la View vera e propria ed è una VBox i cui componenti sono le due HBox StackView (che mostra le ultime posizioni dello stack visibili) e CalculatorView; CalculatorView è a sua volta composto dalla GridPane VariablesView (che mostra gli operatori per le variabili e le variabili stesse) e dalla VBox BaseView (che mostra la text area e un GridPane che contiene il tastierino numerico, gli operatori di base e le operazioni sullo stack).

L'Enumerazione Buttons

Si nota la presenza di una Enumerazione chiamata Buttons dalla quale sono dipendenti il Controller e tutte le classi della View eccetto StackView. Buttons associa ai nomi dei tasti il loro indice nell'array di tipo Button che li contiene, in questo modo la scrittura di codice riferito ad un certo tasto è agevolata e il prodotto risultante sarà più leggibile.

Le Eccezioni

A sinistra del Class Diagram è possibile vedere le Eccezioni create in merito al codice scritto, tutte estendono Exception in quanto sono state ritenute tutte da gestire opportunamente. La scelta di non collegare con rapporti di dipendenza le classi che contengono metodi che lanciano le eccezioni alle eccezioni stesse (opzionale secondo lo standard UML) è motivata dalla salvaguardia della leggibilità del diagramma. Si può comunque fare affidamento sugli Use Case per sapere in quali situazioni bisogna lanciare quali eccezioni.