PROGETTO DI SOFTWARE ENGINEERING

Master’s degree in Computer Engineering

A.A. 2021-22

S.GRIMALDI29@STUDENTI.UNISA.IT
E.DIMAURO5@STUDENTI.UNISA.IT
A.CUZZOCREA2@STUDENTI.UNISA.IT
A.DEGRUTTOLA@STUDENTI.UNISA.IT

Indice

Project Assignment

Problem statement

✦ Develop an application implementing a scientific programmable calculator supporting complex numbers
✦ Starting point: the User Epics given at the end of this presentation

The task

✦ Groups of 4 students
✦ The groups define their user stories
✦ Based on the provided User Epics
✦ Teachers may add or change the user stories during the project
✦ The project will be performed using the Scrum process
✦ **TIME BOXING**: the program **does not need to be complete** at the end of the project!

Tools

✦ The groups must set up a Git repository for the project on GitHub
The repository must contain
✦ Source code (obviously), including unit tests (based on JUnit)
✦ A (short) document describing the software architecture
✦ The product backlog
✦ The sprint backlogs for each sprint
✦ (At the end) A presentation of the project describing both the product and the process

Tools

✦ The team will use Trello (http://trello.com) as a platform for tracking the project activities (Sprint Task Board) and their completion
• For each sprint, the tasks assigned to each member of the team and the time spent on them must be tracked
✦ Links:
- https://blog.trello.com/how-to-scrum-and-trello-for-teams-at-work

The deadlines

✦ First delivery: Sun. November 21
✦ Pre-game: Initial product backlog + 1st sprint planning and backlog + Description of software architecture (5 min. presentation per group)
✦ Second delivery: Sun. November 28
✦ 1st Sprint Release + Sprint backlog + Updated prod. backlog + Project Burndown chart + Sprint review report + Sprint retrospective report
✦ Third delivery: Sun. December 5
✦ 2nd Sprint Release + Sprint backlog + Updated prod. backlog + Project Burndown chart + Sprint review report + Sprint retrospective report
✦ Fourth delivery: Sun. December 12
✦ 3rd Sprint Release + Sprint backlog + Updated prod. backlog + Project Burndown chart + Sprint review report + Sprint retrospective report + Final presentation (8 min. per group)

Classroom activity

✦ Two of the three weekly lectures will be used for short demonstrations of the software released on the previous week (during these demos you may be asked to show also the product/sprint backlogs, review/retrospective reports, burndown chart)
✦ A final demonstration/presentation will be given by each group in the week between Dec 13 and Dec 17

Artifacts: backlogs

✦ The product backlog and sprint backlogs should be kept or copied in a Word or Excel document for ease of inspection by the teachers

• You may use a Google Docs/Sheets shared document, and have in your GitHub repo a file/document containing the links

✦ Don't delete completed items from the backlog; just move them to a different part of the doc

Artifacts: burndown chart

✦ You should make the burndown chart (or the "extended" burndown chart) for tracking the whole project (not the sprint burndown chart)

Artifacts: review report

✦ Keep it short (max 1 page)

✦ Essential information:

- Story Points planned vs completed, Project Velocity
- Short discussion of issues discovered implementing/examining the release (e.g. bugs, technical problems, technical debts)
- Short discussion of issues with the Product Backlog (e.g. need to add/refine/modify some user stories)

Artifacts: retrospective report

✦ Keep it short (max 1 page)

✦ Follow the "starfish diagram"

- Start / More of / Keep doing / Less of / Stop

✦ Keep it about the process (not the product)

Time organization 1

✦ Each member of the group is expected to work on the project for **10 hours** per week

✦ Better to have these hours divided between two or three days (agreed among the members of the team)

Time organization 2

✦ Pre-Game: the members of the team will work mostly together

• Initial product backlog: the members of the team will fill the role of the Product Owner
  ▸ Try to keep a shared vision of the product

• Tools setup: GitHub repo, Trello, coding conventions, development platforms, external libraries (if needed)

• Definition of Done

• Design of the architecture of the program

• First sprint planning (initial estimate of velocity, selected user stories)

• If needed, get acquainted with the architecture and technology chosen (e.g. develop a simple "Hello world"-like program to check that everything is in place for compiling and running code following your architecture/technology/libraries etc.)

Time organization 3

✦ At the beginning of each sprint

• If necessary, Product Backlog grooming: less than 1 hour, together
  ▸ Try to address the issues discovered during the previous sprint review

• Sprint planning: 1.5 hours, together
  ▸ Choose user stories for current sprint
  ▸ Define and assign tasks (requires some design)
  ▸ When dividing a user story into tasks, ensure that there are "intermediate checkpoints" where, even if the story is not completed, the code compiles and passes its tests

Time organization 4

✦ At the beginning of each workday within the sprint:

• Daily meeting: 5 minutes, together

- ▸ What you did last day, what you are going to do today, problems/obstacles you have encountered
- Refine and discuss design decisions: 5-30 minutes, subsets of the team
  - ▸ Discuss decisions with other members whose work is going to be impacted
  - ▸ Agree on the interfaces between the different parts

Time organization 5

✦ At the end of each workday within the sprint:
- Take the time needed to integrate the code that is ready with the rest of the release (you can do this more often than daily)

Time organization 6

✦ At the end of each each sprint:
- Code review: 30 minutes, individually
  - ▸ Examine the code produced by other team members, to spot problems in design or coding (see the checklists later in this presentation)
  - ▸ Better to agree on who reviews what, so as to avoid duplicate work
  - ▸ Not necessary to have a 100% coverage, but try to have a fair sampling (for instance, ensure that work by a team member is reviewed by a different colleague at each sprint)
  - ▸ Problems found can be discussed in the Sprint Review

Time organization 7

✦ At the end of each each sprint:
- Sprint Review: 40 minutes, together
- Sprint Retrospective: 20 minutes, together

Evaluation criteria 1

✦ The rating of each student will be based on the individual tasks performed by the student (as documented in each iteration) and on the overall evaluation of the project
  - Individual tasks: 50%
  - Overal project evaluation: 50%

✦ Each member of the group will be asked at the end to provide (confidentially) an estimate of the relative effort contributed by each other member
  - No "free riders"!

Evaluation criteria 2

✦ Appearance and usability: 10%
✦ Quality of the design: 30%
✦ Quality of the coding: 15%
✦ Quality of the tests: 15%
✦ Compliance with the Scrum process: 30%
✦ Effort: a multiplicative coefficient that scales all of the above
- You are expected to work **10 hours/week**, and divide the tasks among the team members!

Design checklist

✦ Are the packages/classes designed to provide a set of highly related services (high cohesion)?
✦ Are unnecessary dependencies avoided (low coupling)?
✦ Are abstraction mechanisms used to improve reusability and/or break dependency chains?
✦ Are design patterns used when appropriate?
✦ Are duplications/repetitions in the design avoided? (Don't Repeat Yourself principle)

Coding checklist

✦ Are naming conventions followed consistently? Are names chosen so as to be easily understandable?
✦ Is code formatting (e.g. indentation) consistent and readable?
✦ Are "hard coded" constants/"magic" numbers in the code avoided?

✦ Is the code adequately commented? (Comments should describe the interfaces, and why you are doing things a certain way – not be redundant)
✦ Are duplications/repetitions in the code avoided? (Don't Repeat Yourself principle)
✦ Is the code structure readable? (e.g. split too complex operations into several methods)

Testing checklist
✦ Are the unit tests automated?
✦ Do the tests cover the public part of each class? (at least for business-logic classes)
✦ Do the test cases cover adequately the input space (e.g. including limit values, special conditions etc.)

User Epics
WARNING!
✦ The following User Epics are in order of priority (most important first)
✦ You are not expected to complete ALL the User Epics within the project duration
• You must complete as much as you can WITHOUT compromising on the quality

Basic operation of the calculator
The operation of the calculator revolves around a stack data structure. The stack contains complex numbers (of course real numbers are supported as a special case of complex numbers). The user can either input a number (using the Cartesian notation, e.g. "7.2+4.9j"; the imaginary part can be omitted, writing "42" instead of "42+0j"), that is pushed onto the stack, or the name of an operation, that is executed taking its operands from the stack and pushing its result onto the stack. The user interface should show the top locations of the stack (at least 12 elements) and should have a text area for entering the user input. The calculator should support at least the following operations: "+" (addition), "-" (subtraction), "*" (multiplication), "/" (division), "sqrt" (square root), "+-" (invert sign).
For instance, if the user enters the numbers "5" and "9" and then the operations "-" and "sqrt", the stack will contain just the number "0+2j" ("-" takes "5" and "9" from the stack and pushes their difference "-4", then "sqrt" takes "-4" from the stack and pushes its square root "0+2j").

Stack manipulation commands
The calculator includes the following operations for manipulating the stack: "clear" that removes all the elements; "drop" that removes the last element (i.e. the top); "dup" that pushes a copy of the last element; "swap" that exchanges the last two elements; "over" that pushes a copy of the second last element.
Examples:
STACK BEFORE (bottom to top) OPERATION STACK AFTER (bottom to top)
Z1 Z2 Z3 … Zk-2 Zk-1 Zk clear (empty)
Z1 Z2 Z3 … Zk-2 Zk-1 Zk drop Z1 Z2 Z3 … Zk-2 Zk-1
Z1 Z2 Z3 … Zk-2 Zk-1 Zk dup Z1 Z2 Z3 … Zk-2 Zk-1 Zk Zk
Z1 Z2 Z3 … Zk-2 Zk-1 Zk swap Z1 Z2 Z3 … Zk-2 Zk Zk-1
Z1 Z2 Z3 … Zk-2 Zk-1 Zk over Z1 Z2 Z3 … Zk-2 Zk-1 Zk Zk-1

Variables
The calculator supports 26 variables, named with the letters from "a" to "z". For each variable "x", the operation ">x" takes the top element from the stack and saves it into the variable "x". The operation "<x" pushes the value of the variable "x" onto the stack. The operation "+x" takes the top element from the stack and adds it to the value of the variable "x" (storing the result of the addition into "x"). The operation "-x" takes the top element from the stack and subtracts it from the value of the

variable "x" (storing the result of the subtraction into "x").

User-defined operations

The user can define a new operation by specifying a name and a sequence of operations (including the push of numbers). When the userdefined operation is invoked, all the operations in the sequence are executed in order. The definition of a user-defined operation may contain other user-defined operations. The user can delete a user-defined operation. The user can modify the definition of a user-defined operation. The user can save to a file the existing user-defined operations, and reload them from a file, even in a different usage session.
Example: the user can define an operation "hypotenuse", that takes from the stack the values of the two catheti and pushes the value of the hypothenuse. The definition of this operation could be the sequence:
dup * swap dup * + sqrt

Save/Restore variables

The calculator includes the following operations that operate on the variables: "save", which saves a copy of all the 26 variables on a "variable stack" (distinct from the stack used for the operands by the operations); "restore", which restores for all variables the last values that were saved on the "variable stack" (removing them from that stack). It is possible to perform several times the "save" operation, preserving several sets of variable values. This mechanism can be used to temporarily modify a variable within a userdefined operation, without making this modification visible outside of the execution of the user-defined operation.
For example, the "hypothenuse" operation (that takes the two catheti from the stack) could be defined with the following sequence :
save >b >a <a <a * <b <b * + sqrt restore
Another example: the following sequence can be used to define a "solve2degree" operation for solving second degree equations; at the beginning the stack must contain the 3 coefficients of the equation a, b and c (pushed in this order); at the end the stack will contain the two solutions:
save >c >b >a <b <b * 4 <a <c * * - sqrt >d
<b +- <d – 2 <a * / <b +- <d + 2 <a * / restore

Transcendental functions

The calculator includes the following operations, taking complex numbers as operands (from the stack) and producing (possibly) complex results (pushed onto the stack): "mod" (modulus/magnitude); "arg" (argument/phase); "pow" (power); "exp" (exponential); "log" (natural logarithm); "sin" (sine); "cos" (cosine); "tan" (tangent); "asin" (arc sine); "acos" (arc cosine); "atan" (arc tangent).University of Salerno
2021/2022

Convenzioni

Piattaforme utilizzate:

- Netbeans 8.2 IDE
- SceneBuilder
- GitHub
- Trello
- Microsoft Teams
- Telegram

Librerie utilizzate:

- Complex

Convenzioni sul codice:

- Vengono seguite le convenzioni di Java: ad esempio i nomi delle classi iniziano con lettera maiuscola, i metodi iniziano con lettera minuscola, viene adoperata la notazione Camel case, ecc.
- I nomi di classi, package e metodi sono in inglese
- La documentazione al codice (generata attraverso Javadoc) ed eventuali commenti sono in italiano

Definition of done:

- L'implementazione della user story deve essere in accordo con tutti i criteri di accettazione della stessa
- Tutti i test associati alla user story devono essere eseguiti con successo
- Il codice associato alla user story deve essere integrato a quello già scritto
- In seguito all'integrazione tutti i test (anche quelli associati a user stories già completate) devono essere eseguiti con successo
- Il codice associato alla user story deve essere sottoposto a revisione: almeno un'altra persona del team deve revisionarlo

Backlog

Product Backlog

Debt 15. È necessario attuare un accorgimento che impedisca ad un utente di definire delle operazioni che generino un loop infinito

Story Points: 5

Priorità: Should

SaveRes 1. Come utente intermedio
Voglio che la calcolatrice disponga di un'ulteriore struttura dati Stack chiamata "variable stack"
Così da avere una struttura d'appoggio per le variabili

Acc. Criteria:

Story Points: 2

Priorità: Could

SaveRes 2. Come utente intermedio
Voglio che la calcolatrice supporti l'operazione "save"
Così da salvare una copia delle 26 variabili nel variable stack

Acc. Criteria:

Story Points: 3

Priorità: Could

SaveRes 3. Come utente intermedio
Voglio che la calcolatrice supporti l'operazione "restore"
Così da recuperare, per tutte le variabili, gli ultimi valori salvati nel variable stack (rimuovendoli dallo stack)

Acc. Criteria:

Story Points: 3

Priorità: Could

Debt 16. Si evidenzia la mancanza di un meccanismo che renda possibile visualizzare le operazioni-utente a disposizione in un certo istante

Story Points: 3

Priorità: Could

Trascend 1. Come utente
Voglio poter inserire l'operazione "mod"
Così da calcolare il modulo del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 2. Come utente
Voglio poter inserire l'operazione "arg"
Così da calcolare la fase del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 3. Come utente
Voglio poter inserire l'operazione "pow"
Così da calcolare il risultato di X^Y (dove X e Y sono rispettivamente il secondo e il primo elemento nello stack) e inserirlo nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 4. Come utente
Voglio poter inserire l'operazione "exp"
Così da calcolare l'esponenziale del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 5. Come utente
Voglio poter inserire l'operazione "log"
Così da calcolare il logaritmo naturale del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 6. Come utente
Voglio poter inserire l'operazione "sin"
Così da calcolare il seno del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 7. Come utente
Voglio poter inserire l'operazione "cos"
Così da calcolare il coseno del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 8. Come utente
Voglio poter inserire l'operazione "tan"
Così da calcolare la tangente del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 9. Come utente
Voglio poter inserire l'operazione "asin"
Così da calcolare l'arcoseno del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 10. Come utente
Voglio poter inserire l'operazione "acos"
Così da calcolare l'arcocoseno del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Trascend 11. Come utente
Voglio poter inserire l'operazione "atan"
Così da calcolare l'arcotangente del numero presente in cima allo stack ed inserire il risultato nello stack

Acc. Criteria:

Story Points: 5

Priorità: Could

Primo Sprint Backlog

☑ Basic 1. Come utente
Voglio che l'applicazione sia desktop
Cosicché possa accedervi da pc

Acc. Criteria:
Dato che ho il computer in funzione
Quando clicco sull'applicazione
Allora quest'ultima si avvierà

Story Points: 1

Priorità: Must

☑ Basic 2. Come utente
Voglio che l'applicazione visualizzi una finestra a schermo
Cosicché possa interagirvi

Acc. Criteria:
Dato che ho cliccato sull'applicazione
Quando quest'ultima si avvia
Allora vedrò la sua finestra

Story Points: 1

Priorità: Must

☑ Basic 3. Come utente
Voglio che i dati vengano inseriti in una struttura dati Stack
Così da immagazzinarli secondo una logica LIFO

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando un elemento viene inserito nello stack
Allora quest'ultimo verrà visualizzato in cima allo Stack stesso

Story Points: 3

Priorità: Must

☑ Basic 4. Come utente
Voglio che l'interfaccia utente mostri almeno le prime 12 posizioni dello stack
Così da visionare gli ultimi numeri caricati

Acc. Criteria:
Dato che l'applicazione è in funzione
Quando visualizzo la sua interfaccia grafica
Allora vedrò i numeri collocati nelle prime 12 posizioni dello stack (se presenti)

Story Points: 5

Priorità: Must

☑ Basic 5. Come utente
Voglio che l'interfaccia utente abbia una text field
Cosicché possa scrivervi

Acc. Criteria:
Dato che l'applicazione è in funzione
Quando visualizzo la sua interfaccia grafica
Allora vedrò una text field in cui potrò inserire del testo

Story Points: 2

Priorità: Must

☑ Basic 6. Come utente
Voglio inserire un numero complesso in notazione Cartesiana (a+bj), contemplando tutte le possibili forme di scrittura (jb+a, …)
Cosicché esso possa essere caricato nello Stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito nella text field un numero complesso in notazione Cartesiana (a+bj) e clicco invio
Allora questo verrà inserito in cima allo Stack

Story Points: 13

Priorità: Must

☑ Basic 7. Come utente
Voglio poter inserire un numero complesso senza parte immaginaria, omettendo quest'ultima (a~~+bj~~)
Così da rendere più semplice la scrittura di numeri reali

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito nella text field un numero complesso senza parte immaginaria, omettendo quest'ultima (a~~+bj~~), e clicco invio
Allora questo verrà inserito in cima allo Stack

Story Points: 3

Priorità: Must

☑ Basic 8. Come utente
Voglio inserire il simbolo "+"
Così da effettuare la somma tra i primi 2 numeri prelevati dallo stack ed inserire in cima a quest'ultimo il risultato ottenuto

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "+" nella text field e clicco invio
Allora verranno prelevati i primi 2 numeri dallo Stack: quello in cima a quest'ultimo corrisponderà al secondo operando, quello immediatamente al di sotto corrisponderà al primo operando, verrà eseguita la somma ed il risultato verrà inserito nello Stack

Story Points: 8

Priorità: Must

☑ Basic 9. Come utente
Voglio inserire il simbolo "-"
Così da effettuare la differenza tra i primi 2 numeri prelevati dallo stack ed inserire in cima a quest'ultimo il risultato ottenuto

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "-" nella text field e clicco invio
Allora verranno prelevati i primi 2 numeri dallo Stack: quello in cima a quest'ultimo corrisponderà al secondo operando, quello immediatamente al di sotto corrisponderà al primo operando, verrà eseguita la differenza ed il risultato verrà inserito nello Stack

Story Points: 5

Priorità: Must

☑ Basic 10. Come utente
Voglio inserire il simbolo “*”
Così da effettuare il prodotto tra i primi 2 numeri prelevati dallo stack ed inserire in cima a quest’ultimo il risultato ottenuto

Acc. Criteria:
Dato che l’applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo “*” nella text field e clicco invio
Allora verranno prelevati i primi 2 numeri dallo Stack: quello in cima a quest’ultimo corrisponderà al secondo operando, quello immediatamente al di sotto corrisponderà al primo operando, verrà eseguita il prodotto ed il risultato verrà inserito nello Stack

Story Points: 5

Priorità: Must

☑ Basic 11. Come utente
Voglio inserire il simbolo “/”
Così da effettuare il rapporto tra i primi 2 numeri prelevati dallo stack ed inserire in cima a quest’ultimo il risultato ottenuto

Acc. Criteria:
Dato che l’applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo “/” nella text field e clicco invio
Allora verranno prelevati i primi 2 numeri dallo Stack: quello in cima a quest’ultimo corrisponderà al secondo operando, quello immediatamente al di sotto corrisponderà al primo operando, verrà eseguita il rapporto ed il risultato verrà inserito nello Stack

Story Points: 5

Priorità: Must

☑ Basic 12. Come utente
Voglio inserire il simbolo “sqrt”
Così da estrarre la radice quadrata del primo numero prelevato dallo stack ed inserire in cima a quest’ultimo il risultato ottenuto

Acc. Criteria:
Dato che l’applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo “sqrt” nella text field e clicco invio
Allora verrà prelevato il primo numero dallo Stack, ossia quello in cima a quest’ultimo, verrà estratta la sua radice quadrata ed il risultato verrà inserito nello Stack

Story Points: 5

Priorità: Must

☑ Basic 13. Come utente

Voglio inserire il simbolo “+-”
Così da modificare il segno del primo numero prelevato dallo stack ed inserire in cima a quest’ultimo il risultato ottenuto

Acc. Criteria:
Dato che l’applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo “+-” nella text field e clicco invio
Allora verrà prelevato il primo numero dallo Stack, ossia quello in cima a quest’ultimo, verrà eseguita la modifica del segno ed il risultato verrà inserito nello Stack

Story Points: 5

Priorità: Must

Secondo Sprint Backlog

☑ Debt 1. Non è stata ancora prevista una visualizzazione adeguata delle cifre decimali e un separatore delle migliaia. Si intende limitare la visibilità alle prime 8 cifre decimali e individuare un opportuno separatore delle migliaia

Story Points: 3

Priorità: Must

☑ Debt 2. Deve essere aumentata la dimensione finale del font della calcolatrice

Story Points: 1

Priorità: Must

☑ Debt 3. Modificare la visualizzazione dei numeri nello stack in modo che i numeri puramente reali e puramente immaginari vengano visualizzati rispettivamente senza la parte reale e senza la parte immaginaria

Story Points: 2

Priorità: Must

☑ Debt 4. Modificare l'interfaccia grafica al fine di rendere più piacevole la user experience

Story Points: 3

Priorità: Must

☑ Debt 5. Non è stato ancora previsto l'inserimento da parte dell'utente di un'operazione preceduta o seguita da spazi

Story Points: 1

Priorità: Must

☑ Debt 6. Per rimanere coerenti con l'immissione delle operazioni è necessario modificare anche l'immissione dei numeri, facendo in modo che quando un numero viene digitato con degli spazi al suo interno venga mostrato un messaggio di errore "Syntax Error" al momento della pressione del tasto invio. Deve essere comunque consentito che un numero sia preceduto o seguito da spazi

Story Points: 2

Priorità: Must

☑ Stack 1. Come utente
Voglio che la calcolatrice supporti l'operazione "clear"
Così da svuotare tutto il contenuto dello stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "clear" nella text field e clicco invio
Allora verranno eliminati tutti i numeri presenti all'interno dello stack

Story Points: 3

Priorità: Must

☑ Stack 2. Come utente
Voglio che la calcolatrice supporti l'operazione "drop"
Così da rimuovere l'elemento in cima allo stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "drop" nella text field e clicco invio
Allora verrà eliminato il primo numero dallo Stack, ossia quello in cima a quest'ultimo

Story Points: 3

Priorità: Must

☑ Stack 3. Come utente
Voglio che la calcolatrice supporti l'operazione "dup"
Così da inserire nello stack una copia dell'elemento in cima allo stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "dup" nella text field e clicco invio
Allora verrà inserito in cima allo Stack una copia del primo numero dello Stack, ossia quello in cima a quest'ultimo

Story Points: 3

Priorità: Must

☑ Stack 4. Come utente
Voglio che la calcolatrice supporti l'operazione "swap"
Così da scambiare tra loro i primi 2 elementi in cima allo stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "swap" nella text field e clicco invio
Allora verranno invertite le posizioni dei primi 2 numeri in cima allo Stack

Story Points: 3

Priorità: Must

☑ Stack 5. Come utente
Voglio che la calcolatrice supporti l'operazione "over"
Così da inserire nello stack una copia del suo secondo elemento

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il simbolo "over" nella text field e clicco invio
Allora verrà inserito in cima allo Stack una copia del secondo numero dello Stack

Story Points: 3

Priorità: Must

☑ Bug 1. Quando viene eseguita un'operazione che prevede 2 operandi e nello Stack ne è presente uno solo, quest'ultimo viene cancellato al successivo inserimento di un numero da parte dell'utente

Story Points: 3

Priorità: Must

☑ Debt 7. Non è stato ancora introdotto un meccanismo adeguato alla segnalazione di errori all'utente. Si prevede di notificare all'interno della text field sia l'impossibilità di eseguire un'operazione (per mancanza di elementi all'interno dello stack) sia un errore di sintassi

Story Points: 5

Priorità: Must

☑ Debt 8. I test delle operazioni non sono stati progettati tenendo conto di tutti i casi possibili: eccezioni e stato dello Stack. Si intende modificarli opportunamente prevedendo anche di utilizzare il metodo setUp() in modo da ridurre le ripetizioni di codice

Story Points: 5

Priorità: Must

☑ Debt 9. I metodi isImaginary(), isComplex(), pushComplex() e recognizer() della classe Calculator devono essere modificati in modo da ridurre la ripetizione del metodo trim()

Story Points: 1

Priorità: Must

☑ Var 1. Come utente intermedio
Voglio che la calcolatrice abbia a disposizione 26 variabili, indicate con lettere che vanno da "a" a "z"
Così da salvare dei numeri in memoria

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "show" seguito dal nome di una variabile e clicco invio
Allora verrà visualizzato un messaggio contenente il valore della variabile

Story Points: 3

Priorità: Should

☑ Var 2. Come utente intermedio
Voglio poter inserire l'operazione ">x"
Così da rimuovere il numero in cima allo stack e salvarlo nella variabile "x"

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito ">x" nella text field e clicco invio
Allora il numero in cima allo stack verrà rimosso dallo stesso

Story Points: 5

Priorità: Should

☑ Var 3. Come utente intermedio
Voglio poter inserire l'operazione "<x"
Così da inserire il contenuto della variabile "x" nello stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "<x" nella text field e clicco invio
Allora il numero contenuto nella variabile x verrà inserito in cima allo stack

Story Points: 5

Priorità: Should

☑ Var 4. Come utente intermedio
Voglio poter inserire l'operazione "+x"
Così da sommare il numero in cima allo stack e il contenuto della variabile "x" per poi immettere il risultato in cima allo Stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "+x" nella text field e clicco invio
Allora al contenuto della variabile x verrà sommato il numero presente in cima allo stack, tale numero verrà rimosso dallo stack, e nello stesso verrà inserito in cima il risultato ottenuto

Story Points: 5

Priorità: Should

☑ Var 5. Come utente intermedio
Voglio poter inserire l'operazione "-x"
Così da sottrarre il numero in cima allo stack dal contenuto della variabile "x" per poi immettere il risultato in cima allo Stack

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "-x" nella text field e clicco invio
Allora al contenuto della variabile x verrà sottratto il numero presente in cima allo stack, tale numero verrà rimosso dallo stack, e nello stesso verrà inserito in cima il risultato ottenuto

Story Points: 5

Priorità: Should

☑ Bug 2. L’operazione negate() della classe Calculator che viene applicata ad un numero puramente reale oppure ad un numero puramente immaginario dà problemi

Story Points: 1

Priorità: Must

✗ UserOp 1. Come utente avanzato
Voglio poter definire una nuova operazione-utente, specificando un nome ed una sequenza di operazioni (incluso l'inserimento di numeri nello stack)
Così da memorizzare operazioni utili in futuro

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il nome di una nuova operazione-utente seguita dall'operazione stessa e clicco invio
Allora verrà visualizzato un messaggio di corretto salvataggio in memoria

Story Points: 8

Priorità: Should

Terzo Sprint Backlog

☑ Var 4 rev. Come utente intermedio
Voglio poter inserire l'operazione "+x"
Così da sommare il numero in cima allo stack e il contenuto della variabile "x" per poi salvare il risultato in quest'ultima

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "+x" nella text field e clicco invio
Allora al contenuto della variabile x verrà sommato il numero presente in cima allo stack, tale numero verrà rimosso dallo stack ed il risultato verrà inserito nella variabile, sovrascrivendo il valore precedente

Story Points: 2

Priorità: Must

☑ Var 5 rev. Come utente intermedio
Voglio poter inserire l'operazione "-x"
Così da sottrarre dal contenuto della variabile "x" il numero in cima allo stack per poi salvare il risultato in quest'ultima

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito "-x" nella text field e clicco invio
Allora al contenuto della variabile x verrà sottratto il numero presente in cima allo stack, tale numero verrà rimosso dallo stack ed il risultato verrà inserito nella variabile, sovrascrivendo il valore precedente

Story Points: 2

Priorità: Must

☑ Debt 10. I test delle eccezioni lanciate dai metodi di tutte le classi non sono stati progettati correttamente: è necessaria una loro progettazione ed implementazione

Story Points: 2

Priorità: Must

☑ UserOp 1. Come utente avanzato
Voglio poter definire una nuova operazione-utente, specificando un nome ed una sequenza di operazioni (incluso l'inserimento di numeri nello stack)
Così da memorizzare operazioni utili in futuro

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il nome di una nuova operazione-utente seguita dall'operazione stessa e clicco invio
Allora verrà visualizzato un messaggio di corretto salvataggio in memoria

Story Points: 8

Priorità: Should

☑ UserOp 2. Come utente avanzato
Voglio poter definire una nuova operazione-utente che contiene un'operazione-utente precedentemente definita
Così da realizzare delle espressioni ancora più complesse

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il nome di un'operazione-utente seguita dall'operazione stessa che contiene un'operazione-utente precedentemente definita e clicco invio
Allora verrà visualizzato un messaggio di corretto salvataggio in memoria

Story Points: 8

Priorità: Should

☑ UserOp 3. Come utente avanzato
Voglio poter invocare un'operazione precedentemente definita
Così da determinarne l'esecuzione

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il nome di un'operazione-utente e clicco invio
Allora verrà visualizzato un messaggio di corretta esecuzione dell'operazione-utente

Story Points: 8

Priorità: Should

☑ UserOp 4. Come utente avanzato
Voglio poter cancellare un'operazione-utente
Così da eliminare un’operazione-utente non più utile

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito “del” seguito dal nome di un’operazione utente e clicco invio
Allora verrà visualizzato un messaggio di avvenuta eliminazione dell’operazione-utente

Story Points: 5

Priorità: Should

☑ UserOp 5. Come utente avanzato
Voglio poter modificare la definizione di un'operazione-utente
Così da sovrascrivere un'operazione-utente che non ritengo più utile

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando digito il nome di un’operazione-utente già definita seguita da una nuova operazione e clicco invio
Allora verrà visualizzato un messaggio di avvenuta modifica in memoria

Story Points: 3

Priorità: Should

☑ UserOp 6. Come utente avanzato
Voglio poter salvare le operazioni-utente definite in una sessione di utilizzo
Così da poterle riutilizzare anche in sessioni di utilizzo successive

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando clicco sulla voce “Save User-Operations” del menu
Allora verrà visualizzata una finestra in cui è possibile scegliere o creare il file su cui salvare le operazioni-utente

Story points: 8

Priorità: Should

☑ UserOp 7. Come utente avanzato
Voglio poter ricaricare le operazioni-utente salvate in una sessione di utilizzo precedente
Così da poterle riutilizzare senza doverle definire nuovamente

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando clicco sulla voce “Restore User-Operations” del menu
Allora verrà visualizzata una finestra in cui è possibile scegliere il file da cui importare le operazioni-utente

Story Points: 8

Priorità: Should

☑ Instr 1. Come utente
Voglio poter consultare una lista di istruzioni
Così da conoscere i comandi utilizzabili sulla calcolatrice

Acc. Criteria:
Dato che l'applicazione è in funzione e visualizzo la sua interfaccia grafica
Quando clicco sulla voce “Instructions” del menu
Allora verrà visualizzata una finestra in cui è possibile consultare la lista delle istruzioni della calcolatrice

Story Points: 5

Priorità: Should

☑ Debt 11. Modificare ulteriormente l’interfaccia grafica al fine di rendere più piacevole la user experience: concentrarsi in particolare su colori e messaggi di errore

Story Points: 3

Priorità: Should

☑ Debt 12. In seguito al comando show “x” i numeri non vengono visualizzati allo stesso modo rispetto a come vengono visualizzati all’interno dello stack. L’obiettivo è uniformare le visualizzazioni

Story Points: 3

Priorità: Should

☑ Debt 13. La classe FXMLDocumentController deve essere modificata in modo da ottimizzare la gestione del metodo trim()

Story Points: 1

Priorità: Should

☑ Debt 14. I file nel progetto devono essere opportunamente organizzati in package

Story Points: 1

Priorità: Should

Architettura

Progettazione

Lo stile architetturale a cui ci ispiriamo per la progettazione della calcolatrice a Stack è il Model-View-Controller. Quest'ultimo è molto comune nelle applicazioni web e nelle applicazioni desktop e si basa sulla separazione tra la parte del programma che gestisce i dati del problema in esame e la parte del programma che si occupa della visualizzazione dei dati medesimi e dell'interazione con l'utente.

La parte del programma che si occupa di gestire i dati prende il nome di Model; View è la parte che si occupa della visualizzazione; Controller è la parte che si occupa dell'interazione con l'utente. Tipicamente View e Controller sono accoppiati poiché l'interazione con l'utente tende a dipendere da come sono visualizzati i dati.

In figura 1 vi è un prototipo a bassa fedeltà della GUI (Graphic User Interface) che si intende realizzare. Gran parte dell'interfaccia è occupata dalla rappresentazione dello Stack: in particolare si rendono visibili i primi 12 elementi dello stesso. Immediatamente al di sopra è collocata una Text Field per mezzo della quale l'utente può impartire comandi alla calcolatrice.

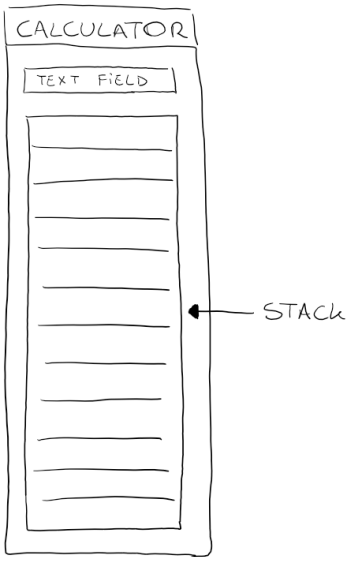


Figura 1

Il programma si basa sulle seguenti classi principali:

- FXMLDocumentController: è il controller dell'applicazione
- ProjectSE: è la classe che estende la classe Application, ovvero l'entry-point dell'applicazione JavaFX
- Calculator: è la classe principale per il funzionamento della calcolatrice
- Complex: è la classe che consente l'utilizzo dei numeri complessi. Essa è situata all'interno della libreria importata da questo repository GitHub
- ArrayDeque<E>: è una classe appartenente al Collection Frame di Java che consente di implementare una logica LIFO.

Le classi Calculator ed ArrayDeque<E>, in particolare ArrayDeque<Complex>, sono in Composizione tra loro: un oggetto della classe ArrayDeque<Complex> è una variabile di istanza per la classe Calculator.

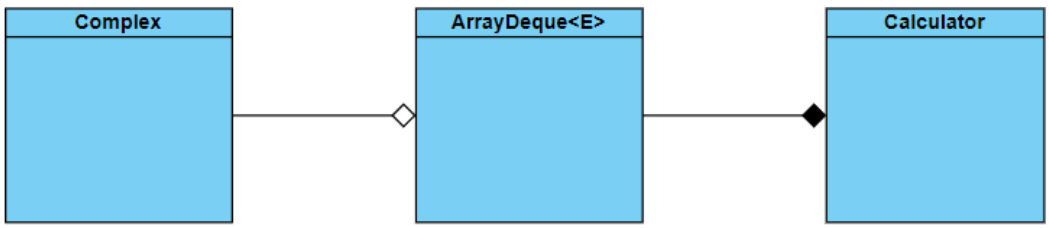


Figura 2

Risultato finale

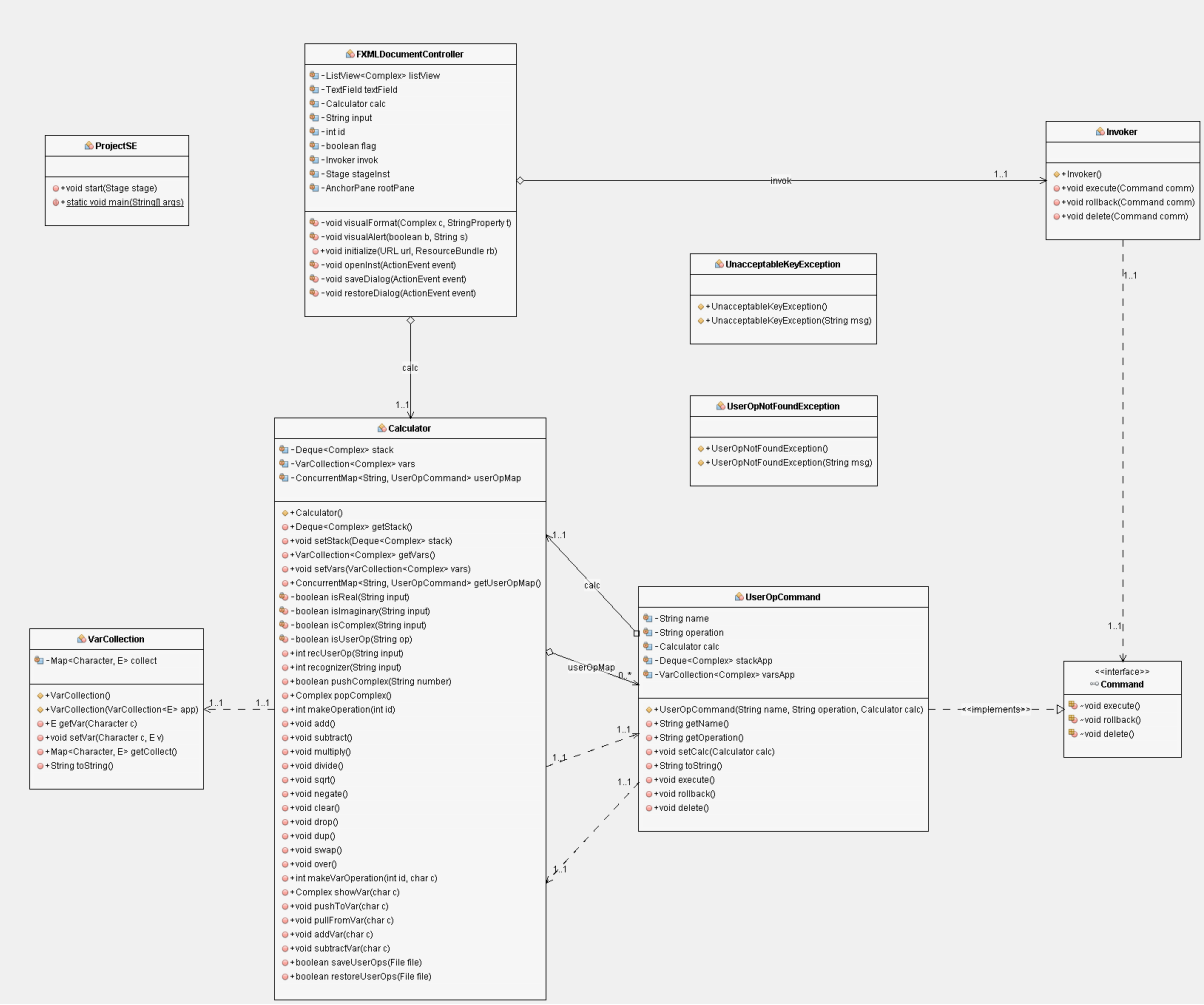


Figura 3

- ProjectSE: è la classe che estende la classe Application, ovvero l'entry-point dell'applicazione JavaFX.
- FXMLDocumentController: è il controller dell'applicazione e, pertanto, gestisce l'interazione tra l'utente e la calcolatrice. Si occupa, tra le altre cose, della corretta visualizzazione dei numeri complessi all'interno della listview, della generazione di messaggi all'interno della textfield, della cattura di possibili eccezioni lanciate dai diversi metodi invocati, della visualizzazione di apposite finestre, come file choosers e alert dialogs.
- Calculator: è la classe principale dell'intero progetto e fornisce metodi e variabili d'istanza che consentono, tra le altre cose, di riconoscere l'input immesso dall'utente, operare sui numeri complessi, agire sullo stack contenente gli stessi, sfruttare 26 variabili [a,…, z], memorizzare operazioni-utente, salvare e ripristinare all'occorrenza queste ultime.
- VarCollection: è la classe che implementa la collezione contenente le 26 variabili [a,…, z] e stabilisce come interagire con esse.
- Command: è l'interfaccia alla base del pattern omonimo.
- UserOpCommand: classe che implementa l'interfaccia Command. Fornisce variabili d'istanza e metodi che consentono di eseguire e cancellare le operazioni-utente. Si

osservi, inoltre, l'importanza del metodo rollback(), che interviene qualora un'operazione-utente sollevi un'eccezione, annullando tutte le modifiche da essa apportate su stack e variabili.
- Invoker: classe prevista dal pattern Command.
- UnacceptableKeyException: eccezione sollevata quando si prova ad accedere (o a modificare) ad una delle 26 variabili in modo erroneo, ovvero mediante chiave inesistente.
- UserOpNotFoundException: eccezione sollevata dal metodo execute() della classe UserOpCommand quando l'operazione-utente che si prova ad eseguire richiama a sua volta un'operazione non più esistente.

Review e Retrospective

Primo sprint

Review

Story Points pianificati

56

Per ipotesi si è stimato di impiegare 10 min per ogni Story Point, il che implica che in uno sprint (10 ore) vengano realizzate circa 60 story points ed in 3 sprint circa 180 story points

Story Points completati

61

Velocità

61 $\frac{story\ points}{sprint}$

Criticità implementative

- Il file .java della libreria utilizzata per la gestione dei numeri complessi presentava incongruenze con la visualizzazione desiderata. Per tale motivo si è scelto di apportare una modifica al metodo toString() del sorgente, per poi ricavarne un file .jar da includere nel progetto.
- La gestione di tutte le forme possibili di immissione da parte dell'utente di un numero complesso ha richiesto l'introduzione di molteplici controlli all'interno dei metodi isImaginary(), isComplex(), pushComplex(). Il debug ed il test sono stati, pertanto, impegnativi ed hanno richiesto buona parte dello sprint per essere completati.
- La volontà di ottenere una migliore visualizzazione degli elementi all'interno dello Stack, centrandoli nelle celle costituenti la ListView (widget scelto per la rappresentazione dello Stack), ha richiesto un'approfondita consultazione della documentazione. Si è deciso di utilizzare il metodo setCellFactory() della classe ListView.
- La libreria Complex presenta una criticità nella gestione del caso particolare in cui *a* e/o *b* sono uguali a "-0" o "-0.000...". Per risolvere tale problema è stato utilizzato il metodo matches() della classe String che, grazie ad un'opportuna espressione regolare, è in grado di individuare il numero 0 in tutte le sue forme. In questo modo è stato possibile modificare qualsiasi occorrenza di quest'ultimo sostituendolo con la forma "0".

Debiti tecnici

- Non è stato ancora previsto l'inserimento da parte dell'utente di un'operazione preceduta o seguita da spazi.
- Non è stata ancora prevista una visualizzazione adeguata delle cifre decimali. Si intende limitare la visibilità alle prime 8 cifre decimali e individuare un opportuno separatore delle migliaia.
- Deve essere aumentata la dimensione finale del font della calcolatrice.
- Modificare la visualizzazione dei numeri nello stack in modo che i numeri puramente reali e puramente immaginari vengano visualizzati rispettivamente senza la parte reale e senza la parte immaginaria.
- Modificare l'interfaccia grafica al fine di rendere più piacevole la user experience.
- Non è stato ancora introdotto un meccanismo adeguato alla segnalazione di errori all'utente. Si prevede di notificare sia l'impossibilità di eseguire un'operazione (per mancanza di elementi all'interno dello stack) sia un errore di sintassi all'interno della textfield.

Criticità su Product e Sprint Backlog

- Non sono state riscontrate particolari criticità sul Product Backlog
- Verso la fine dello sprint, dato che le user stories previste erano state portate a termine, è stata aggiunta un'ulteriore user story: Basic 13

Retrospective

Less of (cose da fare di meno)

Keep doing (cose da continuare a fare)

- Effettuare il daily scrum attraverso videochiamate.

More of (cose da fare di più)

- Maggiore comunicazione durante le giornate lavorative attraverso uno scambio più fitto di messaggi sul gruppo telegram.

Start (cose da cominciare a fare)

- Iniziare le sessioni di lavoro di mattina presto o di primo pomeriggio.

Stop (cose da non fare più)

- Evitare di effettuare sessioni di lavoro serali.

Secondo Sprint

Review

Story Points pianificati

61

Story Points completati

65

Velocità

63 $\frac{story\ points}{sprint}$ (media tra le velocità del primo e secondo sprint)

Criticità implementative

- Inizialmente era stato previsto che +i e -i fossero numeri complessi puramente immaginari da inserire all'interno dello Stack, ma successivamente è nata un'incongruenza con quanto richiesto dal cliente: +i e -i devono essere comandi per eseguire una somma ed una differenza con la variabile i. Si è scelto di impedire l'immissione dell'unità immaginaria digitando +i o -i: qualora l'utente intenda inserire all'interno dello Stack l'unità immaginaria positiva, può digitare *i* o *1i* o *+1i*, mentre per l'unità immaginaria negativa può utilizzare la forma *-1i*. Il problema è stato risolto invertendo l'ordine delle verifiche all'interno del metodo recognizer() della classe Calculator.
- L'implementazione dei metodi riguardanti le variabili ha richiesto particolare attenzione per via della possibilità che venissero lanciate differenti eccezioni (NoSuchElementException quando non vi sono abbastanza elementi nello Stack per eseguire un'operazione, NullPointerException quando viene richiesta un'operazione su una variabile che contiene un valore nullo). Si è ravvisata, pertanto, la necessità di gestirle opportunamente, introducendone anche una ulteriore: UnacceptableKeyException (lanciata quando viene chiamata un'operazione su una variabile, ma quest'ultima non rientra in quelle disponibili, ossia le lettere dell'alfabeto [a-z]). Affinché venissero visualizzati messaggi di errore coerenti, è stato necessario agire anche sul controller catturando opportunamente tali eccezioni quando lanciate.
- L'implementazione della classe di test VarCollectionTest.java non è stata banale per via della scelta di rendere la classe VarCollection generica (così da poterla utilizzare eventualmente anche per la gestione di altri oggetti oltre a quelli istanze della classe Complex). Al fine di rendere quanto più possibile generale il test, all'interno della classe VarCollectionTest è stata creata la variabile di istanza VarCollection<Object>.

Debiti Tecnici

- In seguito al comando show "x" i numeri non vengono visualizzati allo stesso modo rispetto a come vengono visualizzati all'interno dello stack. L'obiettivo è uniformare le visualizzazioni

Criticità sul Product Backlog e Sprint Backlog

- Per la prima volta sono stati incontrati Debiti tecnici e Bug. La scelta della collocazione di questi ultimi nel product backlog non è stata banale, in quanto ha richiesto un confronto tra la loro priorità e quella delle user stories già presenti all'interno del Product Backlog
- Le user stories Var 4 e Var 5 sono condizionate da un'incomprensione dell'epica Variables. Di fatti, il risultato dell'operazione +x e dell'operazione -x non va collocato in cima allo stack bensì va a sovrascrivere il contenuto della variabile x. Si è ravvisata, pertanto, la necessità di introdurre in cima al Product Backlog (per via della loro elevata

priorità) due nuove user stories (Var 4 rev e Var 5 rev) che rispecchiano la corretta interpretazione dell'epica Variables.

Retrospective

Less of (cose da fare di meno)

Keep doing (cose da continuare a fare)

- Effettuare il daily scrum attraverso videochiamate.

More of (cose da fare di più)

Start (cose da cominciare a fare)

Stop (cose da non fare più)

Terzo Sprint

Review

Story Points pianificati

63

Story Points completati

67

Velocità

64 $\frac{story\ points}{sprint}$ (media tra le velocità del primo, secondo e terzo sprint)

Criticità implementative

- L'implementazione dei metodi save() e restore() della classe Calculator ha richiesto una riflessione circa il tipo di file da utilizzare: binario o testuale. Data la natura dell'oggetto da salvare si è scelto di utilizzare un file binario.
- Dato che un'operazione-utente può includere al suo interno altre operazioni-utente, è necessario implementare un meccanismo che renda indisponibili operazioni-utente che richiamano altre operazioni-utente precedentemente eliminate. Si è scelto, pertanto, di adottare una politica di tipo CASCADE per l'implementazione del metodo delete() della classe UserOpCommand.

Debiti tecnici

- Si evidenzia la mancanza di un meccanismo che renda possibile visualizzare le operazioni-utente a disposizione in un certo istante.
- È necessario attuare un accorgimento che impedisca ad un utente di definire delle operazioni che generino un loop infinito. Ad esempio: op1 richiama op2 ed op2 richiama op1.

Criticità sul Product Backlog e Sprint Backlog

- Durante lo sprint sono stati evidenziati debiti prontamente collocati nel product backlog a seconda del loro grado di priorità.

Retrospective

Less of (cose da fare di meno)

Keep doing (cose da continuare a fare)

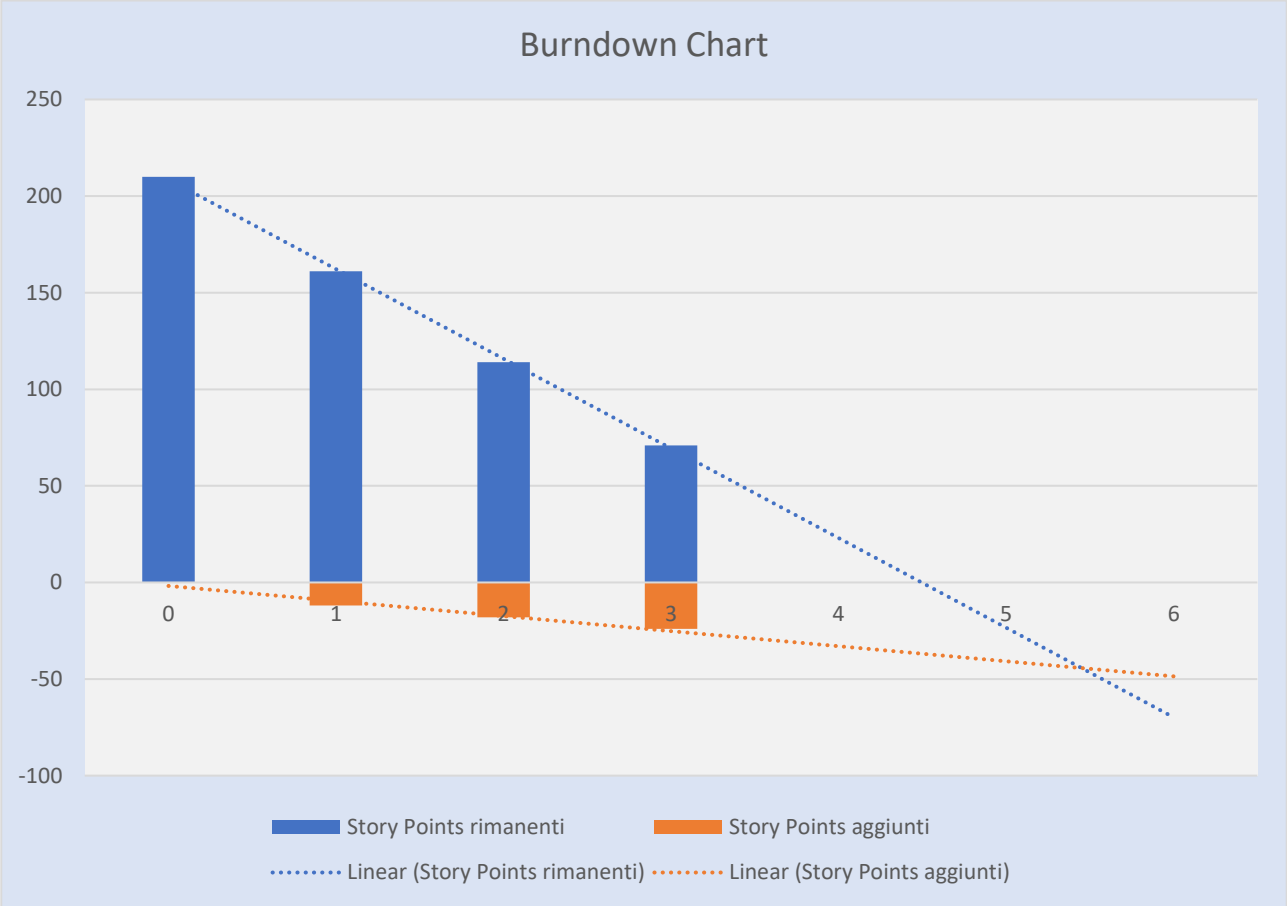
- Effettuare il daily scrum attraverso videochiamate.

More of (cose da fare di più)

Start (cose da cominciare a fare)

Stop (cose da non fare più)

Burndown Chart



Sprints	SP rimanenti	SP aggiunti
0	210	0
1	161	-12
2	114	-18
3	71	-24