LF

RELAZIONE HOMEWORK

Data

01-giu-2023

ESAME:

INTEGRAZIONE E TEST DI SISTEMI SOFTWARE

A.A. 2022-2023

**Realizzato da**

Cafeuri Lorenzo 744403~ITPS~l.cafueri@studenti.uniba.it

Iordache Flavius Gabriel 735748[~ITPS~f.iordache@studenti.uniba.it](mailto:~ITPS~f.iordache@studenti.uniba.it)

Sommario

[HOMEWORK 1 3](#_bookmark0)

1. [Understanding the requirements 3](#_bookmark1)

[Obiettivo della classe 3](#_bookmark2)

[Input 3](#_bookmark3)

[Output 3](#_bookmark4)

1. [Explore what the program does for various inputs 4](#_bookmark5)
2. [Explore inputs, outputs and identify partitions 5](#_bookmark6)

[Input individuali 5](#_bookmark7)

[Combinazioni di input 5](#_bookmark8)

[Output attesi 5](#_bookmark9)

1. [Boundary case 6](#_bookmark10)
2. [Device test cases 6](#_bookmark11)
3. [Automate test case 7](#_bookmark12)

[Casi di Test 8](#_bookmark13)

1. [Augment the test suite with creativity and experience 11](#_bookmark14)

[HOMEWORK 2 11](#_bookmark15)

[Parte 2 13](#_bookmark16)

[Completamento suite di test 14](#_bookmark17)

* 1. [Understanding the requirements 14](#_bookmark18)
  2. [Explore what the program does for various inputs 15](#_bookmark19)
  3. [Explore inputs, output and identify partition 15](#_bookmark20)
  4. [Boundary case 16](#_bookmark21)
  5. [Device test cases 16](#_bookmark22)

[HOMEWORK 3 16](#_bookmark23)

[Codice scelto 16](#_bookmark24)

[Property based testing 17](#_bookmark25)

# HOMEWORK 1

Immagine che contiene testo, schermata

Descrizione generata automaticamenteCODICE SCELTO:

## Understanding the requirements

Obiettivo del metodo

### Il metodo wrapIfMissing serve per assicurarsi che una stringa (str) sia "avvolta" con un'altra stringa (wrapWith), solo se la stringa da avvolgere non inizia o termina già con la stringa specificata.

### Input

* String str: testo di partenza in cui attuare le sostituzioni, può essere anche null;
* String wrapWith: la stringa da sostituire, può essere anche null;

### Output

* Il metodo restituirà la stringa risultante ottenuta dopo aver applicato le operazioni di avvolgimento. Il risultato sarà una stringa avvolta con wrapWith solo se necessario.

## Explore what the program does for various inputs

In questa fase sono state effettuate delle esplorazioni con diversi input per capire al meglio come il programma si comporta in diversi casi.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

## Explore inputs, outputs and identify partitions

### Input individuali

|  |  |
| --- | --- |
| **str** | **wrapWith** |
| Null | Null |
| Empty | Empty |
| str.lenght() > 0 | wrapWith.lenght() > 0 |

### Combinazioni di input

|  |  |  |
| --- | --- | --- |
| **#** | **str** | **wrapWith** |
| C1 | Null | Null |
| C2 | Null | Empty |
| C3 | Empty | Null |
| C4 | Empty | Empty |
| C5 | - | Null |
| C6 | - | Empty |
| C7 | Null | - |
| C8 | Empty | - |
| C9 | - | - |
| C10 | Starts with wrapWith | - |
| C11 | Ends with wrapWith | - |
| C12 | Starts and ends with wrapWith | - |
| C13 | Equal to wrapWith | - |

Quelle rappresentate nella tabella sono alcune delle combinazioni possibili. Queste, ovviamente, non verranno tutte testate ma possono essere ulteriormente scremate fino ad ottenere solo quelle utili per testare al meglio tutte le possibilità.

### Output attesi

|  |
| --- |
| **Output:** |
| Null |
| Stringa vuota |
| Stringa di partenza |
| Stringa con aggiunta del wrap |
| Stringa con aggiunta del wrap solo all’inizio |
| Stringa con aggiunta del wrap solo alla fine |

## Boundary case

Il caso limite trovato riguarda il parametro **wrapWith**: il limite in questo caso si ha quando wrapWith è uguale a **str**, in questo caso il metodo restituirà comunque solamente str in quanto è come se str iniziasse e finisse comunque con wrapWith, di conseguenza il metodo non va a wrappare str

## Device test cases

|  |  |
| --- | --- |
| **T1** | str is null |
| **T2** | str empty |
| **T3** | wrapWith is empty |
| **T4** | wrapWith is null |
| **T5** | All the strings are empty |
| **T6** | All the strings are null |
| **T7** | str starts with wrapWith |
| **T8** | str ends with wrapWith |
| **T9** | str starts and ends with wrapWith |
| **T10** | str is equal to wrapWith |

## Automate test case

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST SCENARIO ID** | | nullEmptyCases | **Test Case Description** | In questa suite di test andiamo a testare tutti i casi in cui sono presenti parametri null o empty |
| Test: | | | | |
| **#** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** |
| T1 | str = null  wrapWith = “a” | null | null | pass |
| T2 | str = empty  wrapWith = “a” | empty | empty | pass |
| T3 | str = “a”  wrapWith = empty | “a” | “a” | pass |
| T4 | str = “a”  wrapWith = empty | “a” | “a” | pass |
| T5 | str = empty  wrapWith = empty | empty | empty | pass |
| T6 | str = null  wrapWith = null | null | null | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST SCENARIO ID** | | strIniziaEOFinisceConWrapWith | **Test Case Description** | In questa suite di test andiamo a testare tutti i casi in cui sono str inizia e/o termina con wrapWith |
| Test: | | | | |
| **#** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** |
| T7 | str = “xab”  wrapWith = “x” | “xabx” | “xabx” | pass |
| T8 | str = “abx”  wrapWith = “x” | “xabx” | “xabx” | pass |
| T9 | str = “xabx”  wrapWith = “x” | “xabx” | “xabx” | pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST SCENARIO ID** | | strUgualeAWrapWith | **Test Case Description** | In questa suite di test andiamo a testare boundary case, dove str è uguale a wrapWith |
| Test: | | | | |
| **#** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** |
| T7 | str = “x”  wrapWith = “x” | “x” | “x” | pass |

## Augment the test suite with creativity and experience

Dopo aver stilato i primi test, abbiamo individuato un ulteriore classe di test andando ad analizzare delle particolari casistiche di wrap. Da queste analisi, sono stati creati altri quattro test:

|  |  |
| --- | --- |
| T11 | str starts with a character contained in wrapWith |
| T12 | str ends with a character contained in wrapwith |
| T13 | str is cointaned in wrapWith |
| T14 | str and wrapWith contain spaces |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST SCENARIO ID** | | testParticolari | **TEST CASE DESCRIPTION** | Test inerenti casi particolari |
| Test: | | | | |
| **#** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** |
| T11 | str = “xab”  wrapWith = “xx” | “xxxabxx” | “xxxabxx” | pass |
| T12 | str = “abx”  wrapWith = “xx” | "xxabxxx" | "xxabxxx" | pass |
| T13 | str = “x”  wrapWith = “xx” | "xxxxx" | "xxxxx" | pass |
| T14 | str = “a b”  wrapWith = “1 1” | "1 1a b1 1" | "1 1a b1 1" | pass |

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

# 

# HOMEWORK 2

# TASK 1

# Test strutturali (White Box)

La prima fase dell’homework 2 è stata quella di esplorare il codice per vedere in dettaglio le operazioni effettuate.

In seguito, abbiamo effettuato una prima analisi di code coverage con i test black-box effettuati nel primo homework e abbiamo notato che il codice del metodo preso in considerazione risultava coperto al 100%.

Dopo un’attenta analisi del codice, non è stato necessario proseguire con il testing strutturale. Non ci è risultato neanche possibile attuare criteri di copertura del codice quali MC/DC o Condition + Branches Coverage, data la semplicità delle condizioni presenti all’interno del codice.

Copertura del codice

Dopo aver avviato il test, è stata riscontrata una copertura totale del codice tramite i test di specifica scritti durante il primo Homework.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

## Parte 2

Il codice scelto per la seconda parte dell’homework 2 è il seguente: identicalColelctions() della classe CollectionOperation

Il numero minimo di test individuato per ricoprire completamente la code coverage è 4, e i test sono i seguenti

Immagine che contiene schermata, testo

Descrizione generata automaticamente

Nel **primo** test, eseguiamo una verifica per identificare se le dimensioni delle due Collection differiscono, al fine di determinare il false del primo **if** nel codice.

Nel **secondo** test, analizziamo la situazione in cui le collection abbiano dimensioni uguali ma contengano stringhe diverse, conducendo ad un false nel secondo **if.**

Nel **terzo** test, esaminiamo il caso in cui le stringhe siano le stesse, ma la loro cardinalità differisce, portando ad un false nel terzo return.

Infine, effettuiamo il **quarto** test in cui le raccolte contengono le stesse stringhe con la stessa cardinalità, che conduce al caso true.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

## Completamento suite di test

### Understanding the requirements

Obiettivo del metodo

Il metodo verifica se le collezioni hanno la stessa dimensione, poi calcola la frequenza di ciascun elemento all'interno delle collezioni e verifica se le frequenze sono uguali per tutti gli elementi. Se le dimensioni sono diverse o se le frequenze degli elementi sono diverse, il metodo restituirà false, altrimenti, restituirà true, indicando se le collezioni sono identiche o no.

*Input*

* Final Collection<?> a
* Final Collection<?> b

In input si prende le due collection da confrontare

*Output*

* Boolean

Il metodo restituisce true se le collection hanno gli stessi elementi con le stesse cardinalità, altrimenti false.

### Explore what the program does for various inputs

In questa fase sono state effettuate delle esplorazioni con diversi input per capire al meglio come il programma si comporta in diversi casi.



Immagine che contiene testo, schermata

Descrizione generata automaticamente

### Explore inputs, output and identify partition

*Input individuali*

|  |  |
| --- | --- |
| **Collection <?> a** | **Collection <?> b** |
| Empty | Empty |
| a.size() > 0 | b.size() > 0 |

*Combinazioni di input*

C1: a = b = Empty

C2: a.size() < b.size()

C3: a.size() > b.size()

C4: a.size() = b.size()

C5: a.size() = b.size(), ma caratteri diversi

C6: a.size() = b.size(), caratteri identici, stessa cardinalità

C7: a.size() = b.size(), caratteri identici, diversa cardinalità

*Output attesi*

|  |
| --- |
| **Output**: |
| True (identiche) |
| False |

### Boundary case

Per quanto riguarda i boundary case abbiamo individuato i casi 'On Point' si verificano quando due collezioni soddisfano contemporaneamente tre condizioni: dimensioni uguali, stesso numero di elementi distinti e frequenze degli elementi uguali. L'unico caso 'In Point' è quando le due collezioni hanno le stesse dimensioni, gli stessi elementi distinti e le stesse frequenze, incluso il caso in cui sono vuote. Tutti gli altri casi sono 'Off Point'.

### Device test cases

|  |  |
| --- | --- |
| **T1** | Collection are empty |
| **T2** | a.size() < b.size() |
| **T3** | a.size() > b.size() |
| **T4** | a.size() = b.size() |
| **T5** | a.size() = b.size() e i caratteri sono differenti |
| **T6** | a.size() = b.size() con stessi caratteri e cardinalità diverse |
| **T7** | a.size() = b.size() con stessi caratteri e uguali cardinalità |

# HOMEWORK 3

## Codice scelto:

Immagine che contiene testo, schermata

Descrizione generata automaticamente

*Obiettivo del metodo*

Il metodo eseguiGioco rappresenta un gioco in cui il giocatore ha tre tentativi per indovinare un numero pensato dal computer (numPensato). Il metodo legge l'input dell'utente, controlla se il numero indovinato dall'utente corrisponde a numPensato, fornisce feedback sul successo o il fallimento del giocatore e tiene traccia delle partite vinte. Alla fine del gioco, il giocatore riceve un messaggio che indica se ha vinto o perso.

Il metodo riceve in input tre valori interi che corrispondono a giorno, mese ed anno di una data e restituisce vero se è una data valida (quindi i tre valori combinati creano una data valida), altrimenti restituisce falso.

Purtoppo per questioni di tempo non siamo riusciti a concludere e a perfezionare questo ultimo homework, nonostante siamo riusciti anche nei property based test, ci scusi anche per il ritardo.

*Lorenzo Cafueri, Flavius Gabriel Iordache*