**Relazione Test**

**Desperate Colleagues 2.5** – **GRUPPO 1**

**Data**

15/06/2023

**Esame**

Integrazione e Test di Sistemi Software

a.a. 2022/2023

REALIZZATO DA:

Detomaso Gabriele – 736743 – g.detomaso6@studenti.uniba.it

Pipoli Alessandra – 744391 – a.pipoli13@studenti.uniba.it

Sommario

[**Homework1 - Black Box Test** 4](#_Toc136427949)

[assignMarkToStudent() 4](#_Toc136427950)

[Comprensione dei requisiti 4](#_Toc136427951)

[Identificazione delle partizioni 4](#_Toc136427952)

[Identificazione dei boundary cases 4](#_Toc136427953)

[Ideare casi di test 5](#_Toc136427954)

[Test case template 5](#_Toc136427955)

[Test su J-Unit 6](#_Toc136427956)

[Modifiche al metodo post testing 8](#_Toc136427957)

[addNewCourseAttender() 9](#_Toc136427958)

[Comprensione dei requisiti 9](#_Toc136427959)

[Identificazione delle partizioni 9](#_Toc136427960)

[Identificazione dei boundary cases 9](#_Toc136427961)

[Ideare casi di test 10](#_Toc136427962)

[Test case template 10](#_Toc136427963)

[Test su J-Unit 10](#_Toc136427964)

[Modifiche al metodo post testing 12](#_Toc136427965)

[getSpecifSubscription() 13](#_Toc136427966)

[Comprensione dei requisiti 13](#_Toc136427967)

[Identificazioni delle partizioni 14](#_Toc136427968)

[Identificazione dei boundary cases 14](#_Toc136427969)

[Ideare casi di test 14](#_Toc136427970)

[Test case template 15](#_Toc136427971)

[Test su J-Unit 15](#_Toc136427972)

[**Homework2 – Task 1** 16](#_Toc136427973)

[Report 16](#_Toc136427974)

[Report dopo le modifiche 18](#_Toc136427975)

[**Homework2 – Task 2** 19](#_Toc136427976)

[countMarksInInclusiveRange() 19](#_Toc136427977)

[Comprensione dei requisiti 19](#_Toc136427978)

[Ideare casi di test per la full code coverage 20](#_Toc136427979)

[Test case template 20](#_Toc136427980)

[Report 21](#_Toc136427981)

[Test su J-Unit 21](#_Toc136427982)

[Individuare le partizioni seguendo una metodologia basata sulle specifiche 22](#_Toc136427983)

[Ideare casi di test utilizzando una metodologia basata sulle specifiche 23](#_Toc136427984)

[**Homework 3** 24](#_Toc136427985)

[getStudentsWithHigherMark() 24](#_Toc136427986)

[Comprensione dei requisiti 24](#_Toc136427987)

[Identificare le proprietà basate sui requisiti 25](#_Toc136427988)

[Test case template 25](#_Toc136427989)

[Property Based Test 25](#_Toc136427990)

# **Homework1 - Black Box Test**

## assignMarkToStudent()

Immagine che contiene testo

Descrizione generata automaticamente

### Comprensione dei requisiti

Il seguente metodo viene utilizzato per assegnare un voto ad uno specifico Studente identificato da una matricola univoca.

Lo studente deve essere già iscritto ad un corso.

Il metodo riceve due parametri in input:

* mark, di tipo Integer, rappresenta il voto assegnato ad uno studente.

Range ipotizzato: da 18 a 30 e lode inclusi.

* mat, di tipo Stringa, rappresenta la matricola assegnata allo studente.

Range ipotizzato: la lunghezza della matricola deve essere uguale a 6 e deve contenere solo caratteri numerici.

Il metodo non restituisce alcun valore (in quanto void).

### Identificazione delle partizioni

|  |  |
| --- | --- |
| Parametro **mark** | Parametro **mat** |
| 1. Interi positivi 2. Interi negativi 3. Valore 0 | 1. Stringa nulla 2. Stringa vuota 3. Stringa di lunghezza > 0 4. Stringa alfanumerica 5. Stringa numerica |

### Identificazione dei boundary cases

Parametro **mark:**

* **On point**: 18 e 31
* **Off point**: 17 e 32
* **In point**: 18 <= mark <= 31
* **Out point**: mark <= 17 OR mark >= 32

Parametro **mat**:

* **On point:** lunghezza = 6 AND numerica
* **Off point**: (lunghezza = 5 OR =7) OR alfanumerica
* **In point**: lunghezza = 6 AND numerica
* **Out point**: (lunghezza <= 5 OR >= 7) OR alfanumerica

### Ideare casi di test

**T1.** mark è un intero positivo valorizzato con 18

**T2.** mark è un intero positivo con valore minore di 18

**T3.** mark è un intero positivo valorizzato con 31

**T4.** mark è un intero positivo con valore maggiore di 31

**T5.** mark è un intero negativo

**T6.** mat è nulla

**T7.** mat ha valore vuoto

**T8.** La lunghezza di mat è minore di 6 e contiene solo caratteri numerici

**T9.** La lunghezza di mat è uguale a 6 e contiene caratteri alfanumerici

**T10.**La lunghezza di mat è maggiore di 6 e contiene solo caratteri numerici

**T11.** mat deve avere come valore la matricola di uno studente non iscritto ad alcun corso.

**T12.** mat deve avere come valore la matricola di uno studente non inserito nel sistema.

### [Test case template](Test%20suites%20Desperate%20Colleagues%20Gruppo1.xlsx)

### Test su J-Unit

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Il metodo **setup()** viene richiamato una sola volta prima dell’esecuzione di tutti i test e inizializza i vari corsi.

Il metodo **init()** viene richiamato prima dell’esecuzione di ogni test e iscrive studenti nel ‘courseManager1’.

Il metodo **tearDown()** viene richiamato dopo l’esecuzione di ogni test e disiscrive gli studenti da tutti i corsi.

Il metodo **clear()** viene richiamato una sola volta terminata l’esecuzione di tutti i test e assegna valore null a tutti i corsi.

I metodi descritti qui sopra fanno parte dei metodi del ciclo di esecuzione dei test su J-Unit quindi sono richiamati per tutti i metodi testati nell’Homework1 e nell’Homework2.

Per non ripetere la descrizione su ogni metodo tstato sono stati riportati solo qui.

#### Test T2, T4 e T5

I tre test sono stati uniti con un assertThrows perché simili, in quanto tutti danno voti non accettabili dal sistema. Di conseguenza devono essere lanciate delle eccezioni.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T1 e T3

I due test sono stati uniti con un assertDoesNotThrow perché simili, in quanto entrambi danno voti accettabili. Di conseguenza ci si aspetta l’assegnazione del voto senza lancio di eccezioni.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T6, T7, T8, T9, T10, T11 e T12

I sette test sono stati uniti con un assertThrows perché simili, in quanto vengono passati al metodo chiamato dei valori per la matricola dello studente che sono in un formato errato o non sono accettabili dal sistema (esempio passaggio di una matricola di uno studente non iscritto al corso).

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

### Modifiche al metodo post testing

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Dopo il fallimento del test T3, causato dal rifiuto del voto 31, è stato incluso il suddetto valore modificando il range dei voti accettati. Questo range è specificato nel metodo setMark() richiamabile su un oggetto di tipo CourseSubstruction.

## addNewCourseAttender()

Immagine che contiene testo

Descrizione generata automaticamente

### Comprensione dei requisiti

Il seguente metodo permette di aggiungere un nuovo studente al corso.

Lo studente non deve già essere iscritto e la data di iscrizione deve precedere la data di scadenza delle iscrizioni.

Il metodo riceve due parametri in input:

* student, oggetto di tipo Studente, rappresenta lo studente che si vuole aggiungere al corso
* subDate, oggetto di tipo LocalDate, rappresenta la data dell’iscrizione

Il metodo restituisce un boolean con valore:

* Vero se l’inserimento va a buon fine
* Falso altrimenti

### Identificazione delle partizioni

|  |  |
| --- | --- |
| Parametro **student** | Parametro **subDate** |
| 1. Valore nullo 2. Valorizzato | 1. Valore nullo 2. Valorizzato |

|  |
| --- |
| Output **Boolean** |
| 1. Valore true 2. Valore false |

### Identificazione dei boundary cases

Parametro **subDate**:

* **On point**: subdate = alla data di chiusura iscrizione
* **Off point**: subDate = della data di chiusura iscrizione + un giorno
* **In point**: subDate <= della data di chiusura iscrizione
* **Out point**: subDate > della data di chiusura iscrizione

### Ideare casi di test

**T1.** student è nullo

**T2.** student è già stato iscritto al corso

**T3.** Student non è già iscritto al corso ma solo inserito nel sistema

**T4.** subDate è nulla

**T5.** subDate è valorizzata con la data antecedente, di un giorno, rispetto alla data di chiusura iscrizione

**T6.** subDate è valorizzata con la data uguale alla data di chiusura iscrizione

**T7.** subDate è valorizzata con la data maggiore, di un giorno, rispetto alla data chiusura iscrizione.

### [Test case template](file:///C:\Users\39331\Progetti\ITSS\CasoStudioITSS\Documentazione%20Gruppo1\Test%20suites%20Desperate%20Colleagues%20Gruppo1.xlsx)

### Test su J-Unit

#### Test T1

Questo test controlla che venga lanciata un’eccezione nel caso si tenti di passare il valore null al posto di un oggetto Student.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T2 e T3

I due test sono stati uniti con un assertEquals perché simili, in quanto entrambi cercano di iscrivere uno studente esistente ad un corso e restituiscono un valore booleano. Questi valori booleani restituiti vengono confrontati con dei valori booleani attesi.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T4

Questo test verifica che venga lanciata un’eccezione nel caso si passi una data nulla in input al metodo chiamato.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T5 e T6

I due test sono stati uniti con un assertTrue perché simili, in quanto le date delle iscrizioni sono ritenute valide. Quindi si verifica che il metodo richiamato restituisca un valore true una volta eseguito.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T7

Questo test verifica, tramite un assertThrows, che non si possa aggiungere un’iscrizione una volta superata la data di chiusura iscrizioni. Il test ha successo se viene lanciata un’eccezione.

Immagine che contiene testo

Descrizione generata automaticamente

### Modifiche al metodo post testing

Immagine che contiene testo

Descrizione generata automaticamente

Per ovviare al fallimento del test T1 si è aggiunto un controllo nel metodo, tale che venga lanciata un’eccezione specifica nel caso si tenti di richiamare il metodo passando in input il valore null al posto dello studente da iscrivere.

## getSpecifSubscription()

Immagine che contiene testo

Descrizione generata automaticamente

### Comprensione dei requisiti

Il seguente metodo permette di cercare tramite la matricola l’iscrizione di uno specifico studente al corso, nel caso lo studente è iscritto viene restituita l’iscrizione.

Lo studente deve essere stato già inserito nel sistema.

Il metodo riceve un parametro in input:

* mat, tipo String, rappresenta la matricola univoca di uno studente da ricercare

Range ipotizzato: la lunghezza della matricola deve essere uguale a 6 e deve contenere solo caratteri numerici.

Il metodo restituisce:

* Un oggetto di tipo CourseSubscription nel caso lo studente sia già iscritto al corso
* Null altrimenti

### Identificazioni delle partizioni

|  |
| --- |
| Parametro **mat** |
| 1. Stringa nulla 2. Stringa vuota 3. Stringa di lunghezza > 0 4. Stringa alfanumerica 5. Stringa numerica |

|  |
| --- |
| Output **courseSubstruction** |
| 1. Valore nullo 2. Valorizzato |

### Identificazione dei boundary cases

Parametro **mat**:

* **On point:** lunghezza = 6 AND numerica
* **Off point**: (lunghezza = 5 OR =7) OR alfanumerica
* **In point**: lunghezza = 6 AND numerica
* **Out point**: (lunghezza <= 5 OR >= 7) OR alfanumerica

### Ideare casi di test

**T1.** mat è nulla

**T2.** mat ha valore vuoto

**T3.** La lunghezza di mat è minore di 6 e contiene solo caratteri numerici

**T4.** La lunghezza di mat è uguale a 6 e contiene caratteri alfanumerici

**T5.** La lunghezza di mat è uguale a 6, contiene solo caratteri numerici e appartiene ad uno studente iscritto al corso

**T6.** La lunghezza di mat è maggiore di 6 e contiene solo caratteri numerici

**T7.** mat deve avere come valore la matricola di uno studente inserito nel sistema ma non iscritto ad un corso

**T8.** mat deve avere come valore la matricola di uno studente non inserito nel sistema

### [Test case template](file:///C:\Users\39331\Progetti\ITSS\CasoStudioITSS\Documentazione%20Gruppo1\Test%20suites%20Desperate%20Colleagues%20Gruppo1.xlsx)

### Test su J-Unit

#### Test T1

Questo test verifica, tramite un assertThrows, che venga lanciata un’eccezione qualora si tenti di passare il valore null al posto della matricola dello studente del quale si vuole cercare l’iscrizione.

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T2, T3, T4, T6, T7 e T8

Questi 6 test sono stati uniti con un assertNull perché simili, in quanto vengono passati al metodo chiamato dei valori per la matricola dello studente che sono in un formato errato o non sono accettabili dal sistema (esempio, passaggio di una matricola di uno studente non iscritto al corso). Ovviamente il valore atteso dall’esecuzione del metodo è null.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

#### Test T5

Questo test permette di verificare, tramite un assertIstanceOf, se dando in input un parametro corretto viene effettivamente restituita l’iscrizione al corso dello studente cercato.

Immagine che contiene testo

Descrizione generata automaticamente

# **Homework2 – Task 1**

### Report

Dopo aver testato i metodi con una metodologia black box, utilizzando il tool JaCoCo è stato generato il report di code coverage, il criterio utilizzato è: Condition + Branch Coverage.

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

Da questo emerge che il metodo getSpecificSubscription() risulta non totalmente coperto.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Precisamente la riga 123 è evidenziata in giallo, questo equivale a direche viene valutato solo uno dei 2 rami dell’else if. Effettivamente eseguendo il codice con il debug si è confermato che durante l’esecuzione del metodo, in caso di condizione (dell’else if a riga 123) vera l’esecuzione passa dal *ramo* vero (viene eseguita la riga 124), nel caso di condizione falsa l’esecuzione si ferma prima (nello specifico al primo if o al secondo else if).

Due motivazioni:

1. **Mutua esclusione dell’if – else if**. Il motivo per cui si usa la struttura if – else if è proprio l’ottimizzazione delle espressioni logiche da calcolare. Appena una condizione risulta vera le successive vengono ignorate.
2. **Presenza del return**. Fa’ sì che, durante l’esecuzione, una volta entrati nell’if di riga 120 il metodo termini restituendo il risultato, saltando così l’esecuzione delle righe successive.

Per coprire totalmente il metodo si è deciso di modificare gli else if in semplici if, in modo da garantire la valutazione di tutte le condizioni al momento di esecuzione del metodo. Inoltre sono state create 2 nuove variabili, una conterrà il risultato da restituire mentre l’altra è un Boolean che funge da flag di segnalazione. L’introduzione di queste 2 nuove variabili serve a posticipare il return dopo l’esecuzione di tutte le condizioni, infatti solo quando viene trovato l’elemento ricercato si setta il flag a ‘true’ così che possa essere eseguito il break a riga 133 che permette di terminare il ciclo e ritornare il valore.

Immagine che contiene testo

Descrizione generata automaticamente

### Report dopo le eventuali modifiche

Immagine che contiene grafico

Descrizione generata automaticamente

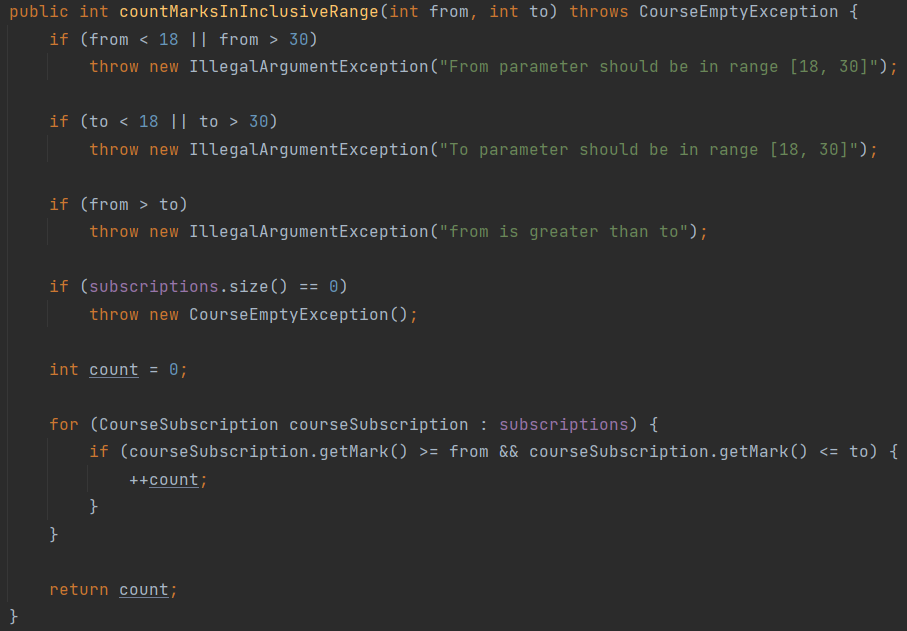
### Conclusioni

Questa ovviamente è una soluzione adottata per raggiungere il 100% di code coverage, ma va a discapito dell’ottimizzazione del codice. Infatti prima delle modifiche se la condizione del primo if risultava vera le altre 2 condizioni presenti nei successivi else if non venivano calcolate, condizioni che ora vengono calcolate a prescindere dal risultato della prima, appesantendo così il carico di lavoro del metodo e aggiungendo di fatto calcoli inutili.

Per tali motivi si è deciso di fare roll back delle modifiche, per preservare l’ottimizzazione del metodo, in quanto il codice non coperto non genera fault

# **Homework2 – Task 2**

## countMarksInInclusiveRange()



### Comprensione dei requisiti

Il seguente metodo permette di conteggiare il numero di studenti, iscritti in un determinato corso, con voto incluso in un range passato in input.

Il metodo riceve due parametri in input:

* **from**, tipo int, rappresenta il limite inferiore del range dei voti da prendere in considerazione per il conteggio degli studenti.
* **to**, tipo int, rappresenta il limite superiore del range dei voti da prendere in considerazione per il conteggio degli studenti.

Sia il parametro from, che il parametro to, devono avere come valore un numero che vada da 18 a 30 inclusi. Inoltre, from deve essere minore di to. Ovviamente è richiesto che ci debba essere almeno uno studente iscritto al corso.

La mancanza di una di queste precondizioni comporterà il lancio di una eccezione da parte del metodo durante la sua esecuzione.

Il metodo restituisce:

* Un valore int che rappresenta il numero di studenti iscritti al corso con un voto presente nel **range** di voti accettabili -> [18 , 30].

### Ideare casi di test per la full code coverage

Il criterio utilizzato è: Condition + Branch Coverage.

**T1.** from è un intero positivo minore di 18

**T2.** from è un intero positivo maggiore di 30

**T3.**  to è un intero positivo minore di 18

**T4.** to è un intero positivo maggiore di 30

**T5.** from e to sono valori contenuti nel range e from è maggiore di to

**T6.** from e to sono valori contenuti nel range e non ci sono iscritti nel corso

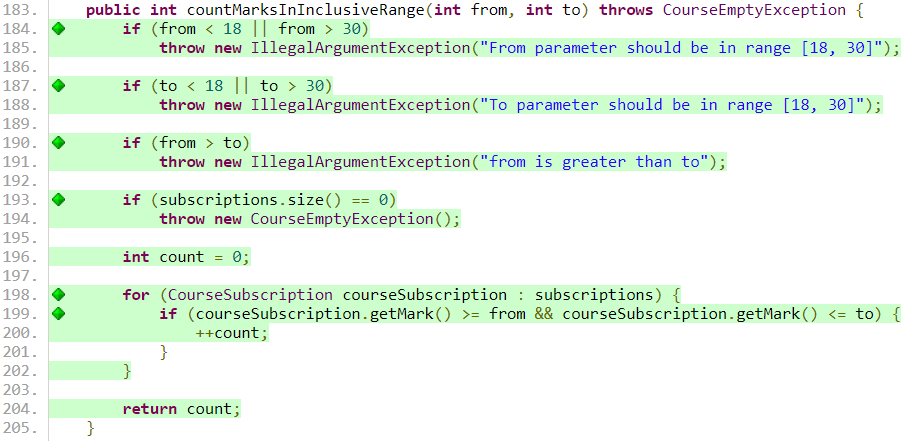
**T7.** from e to sono valori contenuti nel range e ci deve essere un solo studente iscritto al corso che abbia il voto compreso nel range

**T8.** from e to sono valori contenuti nel range e ci devono essere solo studenti che abbiano il voto minore al limite inferiore del range

**T9.** from e to sono valori accettabili nel range e ci devono essere solo studenti che abbiano il voto maggiore al limite superiore del range

### [Test case template](file:///C:\Users\39331\Progetti\ITSS\CasoStudioITSS\Documentazione%20Gruppo1\Test%20suites%20Desperate%20Colleagues%20Gruppo1.xlsx)

### Report

Una volta implementati i test per raggiungere il 100% di code coverage è stato generato il report: 

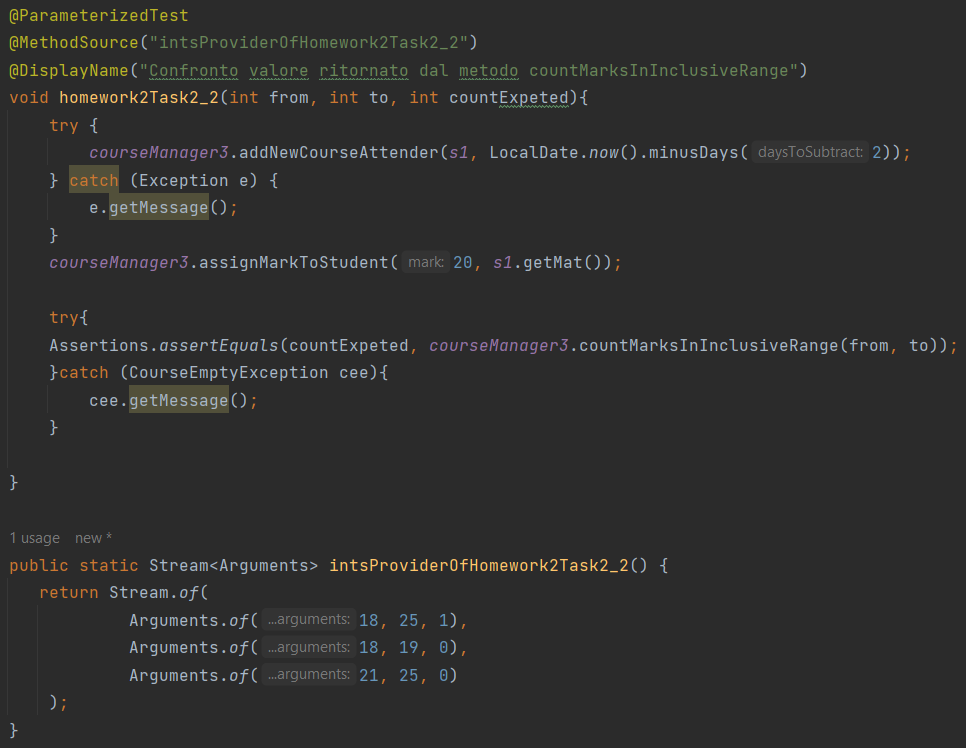
### Test su J-Unit

Di seguito i 2 test parametrici scritti:

Il test **homeworkTask2\_1** verifica i casi in cui il metodo richiamato lancia un’eccezione.



Il test **homeworkTask2\_2** verifica i casi in cui il metodo termina l’esecuzione restituendo un valore.



Nonostante la copertura totale del codice del metodo, risulta evidente la presenza di un bug.

Per la logica con cui è stata progettata la classe il range di voti accettabili dovrebbe essere [18, 31] essendo possibile assegnare 31 come voto ad uno studente iscritto. Se si inserisce 31 come valore di from o to il metodo lancia un’eccezione.

### Individuare le partizioni seguendo una metodologia basata sulle specifiche

|  |  |
| --- | --- |
| Parametro **from** | Parametro **to** |
| 1. Valore negativo 2. Valore positivo | 1. Valore negativo 2. Valore positivo |

|  |
| --- |
| Output **count** |
| 1. Valore 0 2. Valore > 0 |

### Ideare casi di test utilizzando una metodologia basata sulle specifiche

Per il ragionamento fatto in precedenza è importante includere altri test che potrebbero individuare possibili comportamenti logicamente sbagliati.

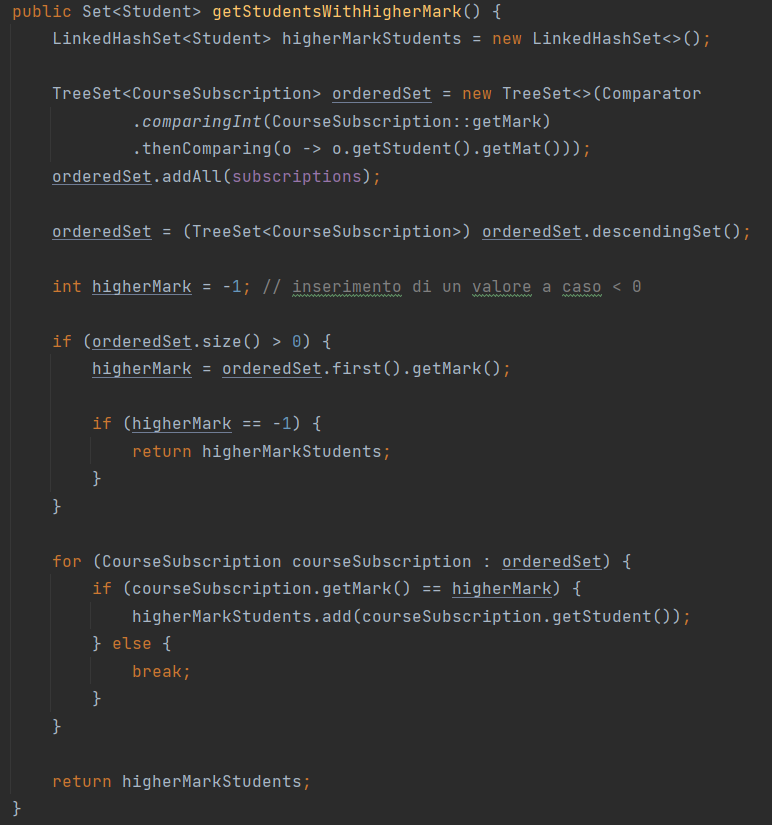
**T1** From ha valore 31

**T2** To ha valore 31

**T3** From e To devono avere uno stesso valore e devono esserci almeno uno studente iscritto con lo stesso voto

# **Homework 3**

## getStudentsWithHigherMark()



### Comprensione dei requisiti

Il seguente metodo permette di conteggiare il numero di studenti con il voto più alto, iscritti in un determinato corso.

Il metodo non riceve nulla in input, ma utilizza l’insieme delle iscrizioni contenute nello specifico corso per ricercare gli studenti con voto più alto.

Il metodo restituisce:

* Un Set non vuoto di studenti, se nel corso è presente almeno un’iscrizione e almeno uno studente ha un voto assegnato.
* Un Set vuoto di studenti, se nel corso non ci sono iscrizioni o se nessuno studente iscritto ha un voto assegnato.

### Identificare le proprietà basate sui requisiti

**Restituzione insieme pieno** di studenti con voto maggiore - Il corso ha almeno uno studente iscritto e almeno uno studente ha il voto assegnato.

**Restituzione insieme** di studenti **vuoto** - Il corso non ha studenti iscritti o tutti gli iscritti non hanno un voto assegnato.

### [Test case template](file:///C:\Users\39331\Progetti\ITSS\CasoStudioITSS\Documentazione%20Gruppo1\Test%20suites%20Desperate%20Colleagues%20Gruppo1.xlsx)

### Property Based Test

#### Restituzione insieme pieno

METODO PROVIDE

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Il metodo ‘studentsProvider’ restituisce al metodo property chiamante un oggetto di tipo ListArbitrary (contenente oggetti Arbitrary) che servirà per generare la lista di studenti casuali.

Sono stati inserite varie costrizioni come la lunghezza del nome, cognome e matricola dello studente, i vari valori con cui generare le stringhe e il vincolo che la matricola sia diversa per ogni studente.

METODO PROPERTY

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

il metodo property ‘studentWithMarkAndWithout’ genera 4 parametri:

* Una lista di studenti di dimensione variabile [20, 35]
* Una lista di 35 date con intervallo di date generabili [2023-05-01, 2023-05-31]
* Una lista di 35 voti con intervallo di voti generabili[18, 31]
* Una lista di 5 interi con intervallo di interi generabili [0, 35], segnala 5 posizioni in cui lo studente non avrà voto.

Una volta istanziato il nuovo corso chiamato ‘courseManager4’ si procede a salvare in ‘higherMark’ il voto più alto generato facendo attenzione ad ignorare i voti generati per le posizioni degli studenti senza voto.

Successivamente viene riempito il corso con gli studenti (e il relativo voto) generati, tenendo sempre conto di lasciare senza voto gli studenti con posizione contenuta nella lista ‘pos’. Contemporaneamente gli studenti con voto più alto vengono inseriti anche in un Set utile poi per confrontare il risultato restituito dal metodo testato.

Il test ha successo se il Set di studenti restituito è uguale al Set ‘higherMarkStudents’.

STATISTICHE

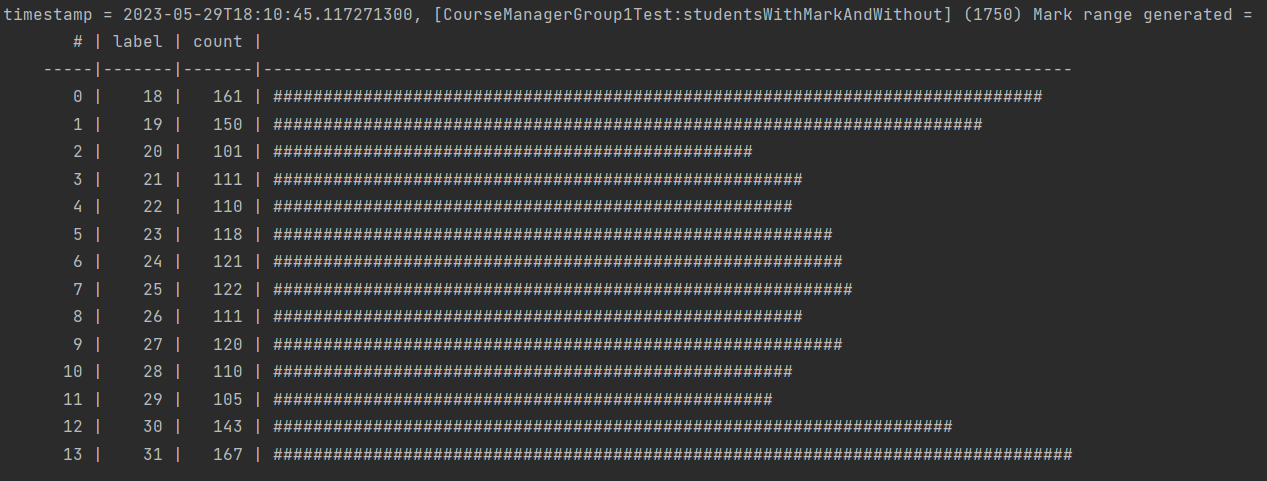
**Higher mark range assigned**

Immagine che contiene schermata, testo

Descrizione generata automaticamente

Questa statistica riporta quante volte ogni voto è stato il più alto del corso su 50 prove.

**Mark range generated**



Questa seconda statistica riporta tutti i voti generati in 50 prove, indipendentemente se siano stati assegnati o meno. Come si può notare i valori più generati sono gli estremi dell’intervallo, dove è più probabile che ci siano dei bug.

Questa osservazione non è inaspettata in quanto il framework Jqwik si concentra maggiormente sul generare i valori per testare i boundary case.

#### Restituzione insieme vuoto

METODO PROVIDE

Il provide ‘studentsProvider’ per generare la lista degli studenti è lo stesso descritto in precedenza.

METODO PROPERTY

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Il metodo property ‘allStudentsWithoutMark’ genera 2 parametri:

* Una lista di studenti di dimensione variabile [0, 35]
* Una lista di 35 date con intervallo di date generabili [2023-05-01, 2023-05-31]

Il corso ‘courseManager4’ una volta istanziato viene riempito con gli studenti presenti nella lista ‘students’, ma a nessuno viene assegnato il voto. Nel caso la lista abbia 0 elementi il corso rimane vuoto.

Il test ha successo se il metodo restituisce un Set vuoto di studenti.