UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO
DIPARTIMENTO DI INFORMATICA

TerMol

RELAZIONE HOMEWORK

Data

20-gen-2024

ESAME:

INTEGRAZIONE E TEST DI SISTEMI SOFTWARE

A.A. 2023-2024

Realizzato da:

Vito Angione 758172 v.angione2@studenti.uniba.it

Roberto Guastamacchia 754710 r.guastamacchia9@studenti.uniba.it

Sommario

HOMEWORK 1:

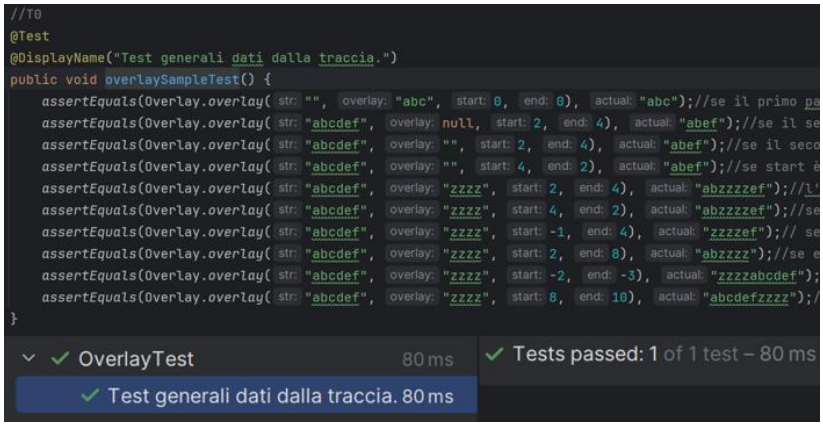
7 STEP:

1- Comprendere i requisiti (cosa deve fare il programma, input e output).

Il programma Java definisce una classe chiamata Overlay con un metodo overlay che prende una stringa di input, una stringa di overlay, e due indici start ed end. Il metodo sovrappone la parte della stringa di input tra gli indici specificati con la stringa di overlay e restituisce il risultato. Il codice gestisce casi speciali come stringhe nulle o vuote, indici negativi, pari a zero e indici che superano la lunghezza della stringa di input. L'obiettivo principale è manipolare e combinare le stringhe in base agli indici e alle regole specificate.

2- Esplora cosa fa il programma per vari input.

Per quanto riguarda il metodo overlay, abbiamo creato otto specification-based test per verificare, in base ai vari input forniti nel testo, che il metodo restituisca la giusta sovrapposizione della stringa, rispettando le regole definite.



➔ Come si puo notare, il test e andato a buon fine.

3- Esplorare input, output ed identificare le partizioni:

•Funzione Overlay:

-Il parametro "**str**", può essere nullo, vuoto o la sua lunghezza può essere maggiore di zero, ovvero contiene caratteri.
str = {null || "" || str.length > 0}

-Il parametro "**overlay**", può essere nullo, vuoto o la sua lunghezza può essere maggiore di zero, ovvero contiene caratteri.
overlay = {null || "" || str.overlay > 0}

-Il parametro "**start**", indica l'indice di inizio, può essere minore, pari o maggiore di zero.
Se maggiore di zero, può essere inferiore, pari o superiore alla lunghezza della stringa "str.length" ed inferiore, pari o superiore all'indice di fine "end".
start = {< 0 || == 0 || > 0 || start < str.length || start > str.length || start == str.length || start < end || start == end || start > end}

- Il parametro "**end**", indica l'indice di fine, può essere minore, pari o maggiore di zero.
Se maggiore di zero, può essere inferiore, pari o superiore alla lunghezza della stringa "str.length" ed inferiore, pari o superiore all'indice di inizio "start".

end = {< 0 || == 0 || > 0 || end < str.length || end > str.length || end == str.length|| end < start || end == start || end > start}

•Overlay (combinazioni):

#	str	overlay	start	end	Output attesi
C1	null	null	< 0	< 0	null
C2	null	null	< 0	> len	null
C3	null	null	> len	< 0	null
C4	null	null	> len	> len	null
C5	null	null	> end	< 0	null
C6	null	overlay.length > 0	< 0	< 0	null
C7	null	overlay.length > 0	> 0	> 0	null
C8	str.length = 0	Null	> 0	> 0	""
C9	str.length = 0	Null	< 0	< 0	""
C10	str.length = 0	Null	> len	> len	""
C11	str.length = 0	overlay.length > 0	< 0	< 0	overlay
C12	str.length = 0	overlay.length > 0	< 0	> 0	overlay
C13	str.length = 0	overlay.length > 0	> 0	> 0	overlay
C14	str.length > 0	null	< 0	< 0	str
C15	str.length > 0	null	0	0	str
C16	"lorenzo"	null	1	2	"lrenzo"
C17	str.length > 0	null || overlay.length >= 0	< 0 || = 0 || > 0 || > end || > len	< 0 || = 0 || > len || >0 && < len || > 0 && > len	Stringa modificata in base alle combinazioni

4- Casi limite:

▪Se la stringa non è nulla, (str != null), abbiamo che "null" è l'out-point e on-point, mentre "str" è l'in-point e off-point.
- "null" è l'out point e on point.
- "str" e un in point e off point.

5- Definire i casi di test:

- Casi eccezionali:

- T1: Se la stringa è nulla: (str == null);
- T2: Se l'overlay è nullo: (overlay == null);

- Test per i parametri vuoti:

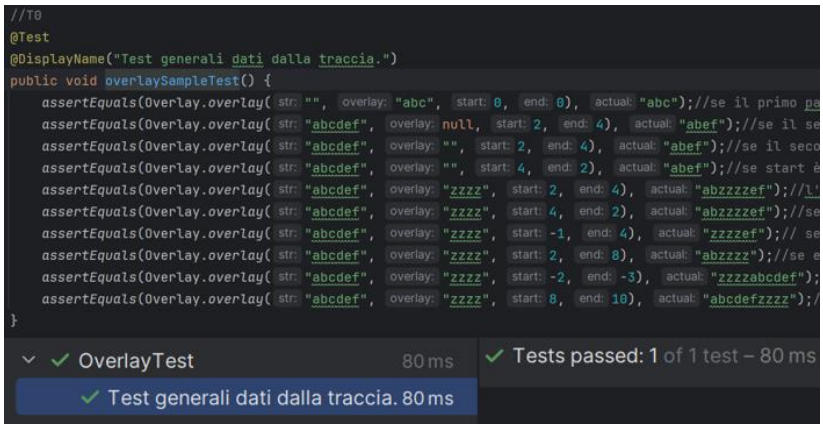
- T3: Se la stringa è vuota: (str == "");
- T4: Se l'overlay è vuoto: (overlay == "");

- Test sulle combinazioni:

- T5: Le stringhe sono vuote o contengono caratteri, l'overlay è vuoto o contiene caratteri, mentre gli indici start ed end sono entrambi negativi.
- T6: Sia la stringa che l'overlay contengono solo un carattere, mentre start ed end contengono numeri positivi, negativi e uguali.
- T7: Sia la stringa che l'overlay contengono caratteri speciali, spazi o punteggiature, mentre start ed end contengono numeri positivi, negativi e uguali.

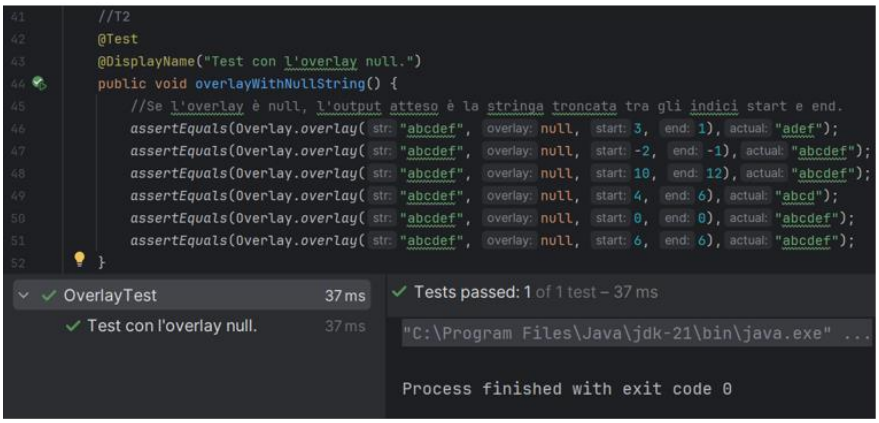
6- Automatizzare casi di test:

- Test generale dato dalla traccia - T0:

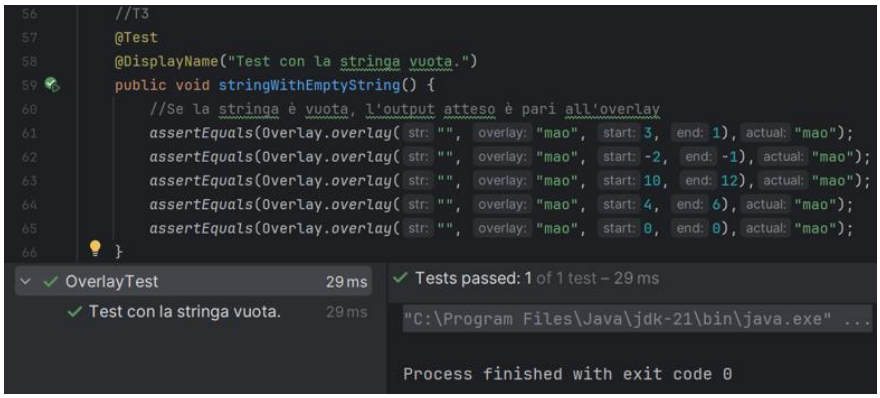


- T1 - Stringa nulla:

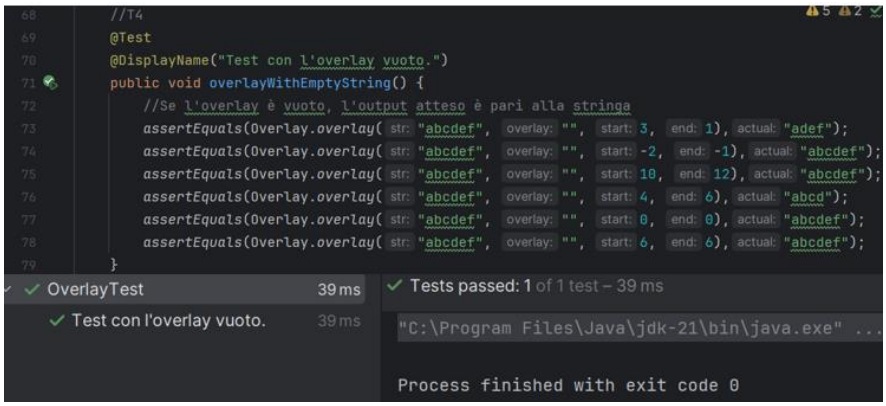
- T2 - Test con l'overlay null:



- T3 - Test con la stringa vuota:

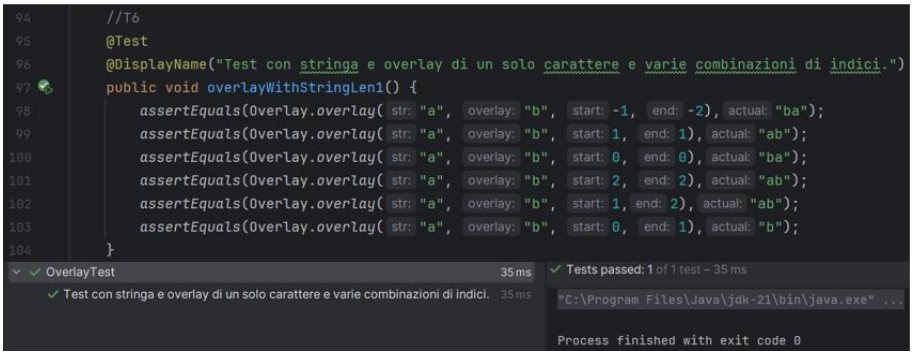


- T4 - Test con l'overlay vuoto:



- T5 - Test con gli indici start ed end negativi (< 0):

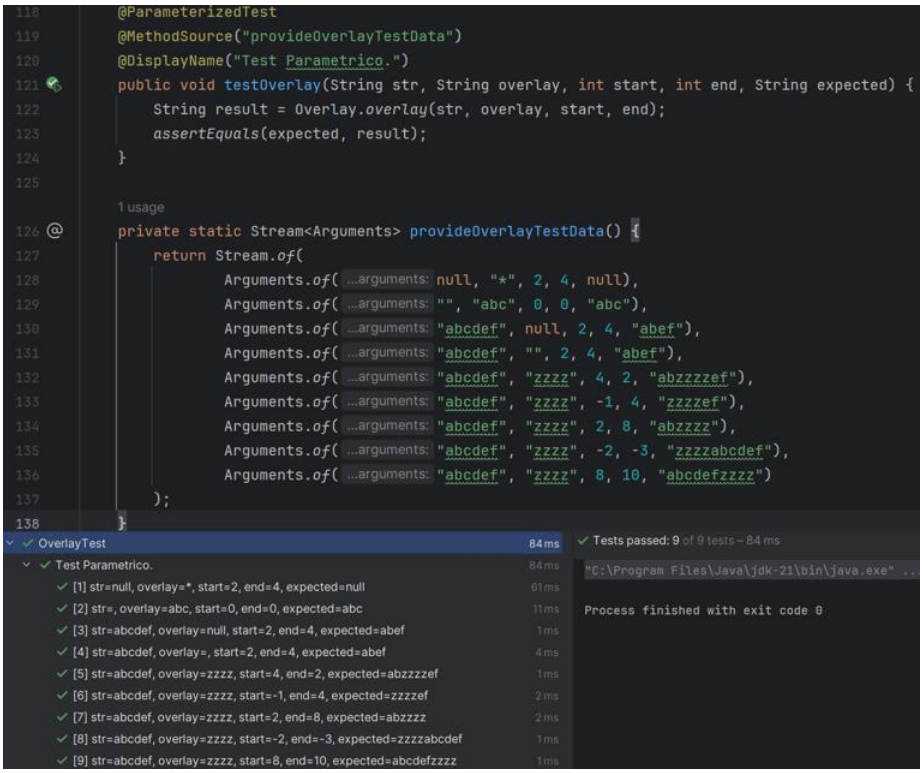
- T6 - Test con stringa e overlay di un solo carattere e varie combinazioni degli indici:



7- Aumentare la suite con creatività ed esperienza:

- **T7 - Test con stringa e overlay che contengono spazi, caratteri speciali e varie combinazioni degli indici:**
  Questo test è un caso di prova (test case) per il metodo overlay della classe Overlay.
  Questi test sono progettati per coprire diversi scenari di sovrapposizione di stringhe, inclusi casi con caratteri speciali, spazi e indici variabili. L'obiettivo è assicurarsi che il metodo overlay gestisca correttamente tali situazioni restituendo il risultato atteso.

- T8 - Test parametrico: provideOverlayTestData

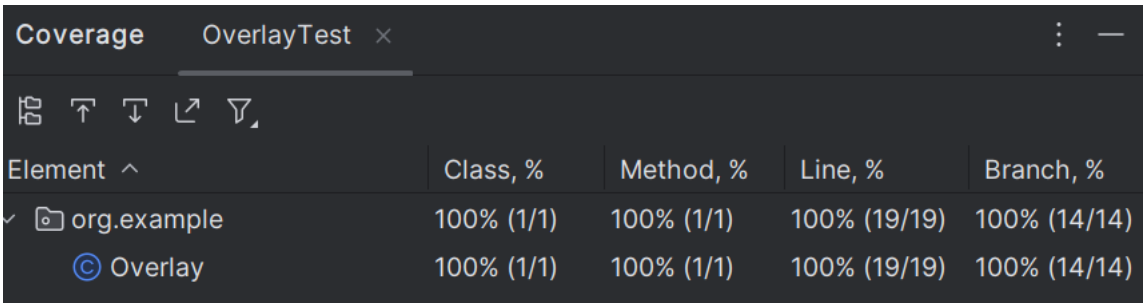


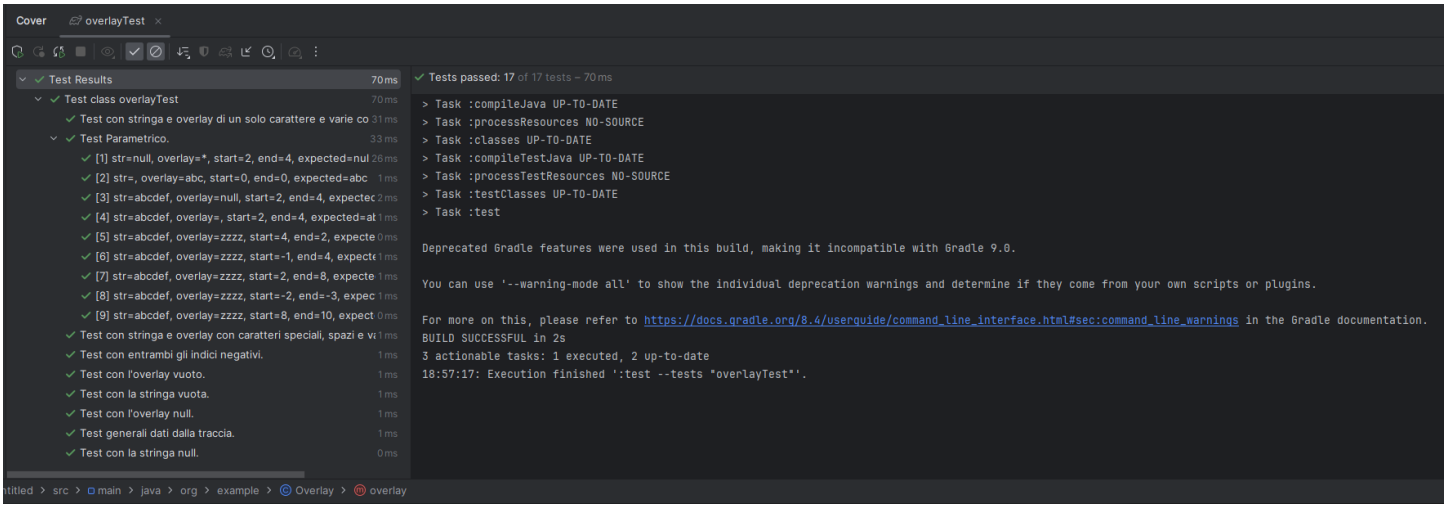
LEGGERE L'IMPLEMENTAZIONE:

```java
public static String overlay(final String str, String overlay, int start, int end) {
    //verifico se la stringa è nulla
    if (str == null) {
        return null;
    }
    //verifico se l'overlay è nullo
    if (overlay == null) {
        overlay = "";
    }
    //prendo la lunghezza della stringa
    final int len = str.length();
    //verifico se start è minore di zero, altrimento lo azzero
    if (start < 0) {
        start = 0;
    }
    //verifico se start è maggiore della lunghezza, altrimento lo pongo uguale alla lunghezza
    if (start > len) {
        start = len;
    }
    //verifico se end è minore di zero, altrimento lo azzero
    if (end < 0) {
        end = 0;
    }
     //verifico se end è maggiore della lunghezza, altrimento lo pongo uguale alla lunghezza
     if (end > len) {
         end = len;
     }
     //verifico se start è maggiore della fine, inverto start con end
     if (start > end) {
         final int temp = start;
         start = end;
         end = temp;
     }
     //restituisco l'overlay delle stringhe
     return str.substring(0, start) +
             overlay + str.substring(end);
 }
}
```

ESEGUIRE TEST CON TOOL DI CODE COVERAGE:

Abbiamo effettuato una prima analisi di code coverage con i test black-box effettuati nel primo homework e abbiamo notato che il codice risultava coperto al 100%.
Ciò non vuol dire che è stato sicuramente testato tutto al meglio, ma può darci un'indicazione sulle parti che sono state interessate dai nostri test.





overlayTest: 17 total, 17 passed — 70 ms

Collapse | Expand

Test class overlayTest		70 ms
Test con stringa e overlay di un solo carattere e varie combinazioni di indici.	passed	31 ms
Test Parametrico.		33 ms
[1] str=null, overlay=*, start=2, end=4, expected=null	passed	26 ms
[2] str=, overlay=abc, start=0, end=0, expected=abc	passed	1 ms
[3] str=abcdef, overlay=null, start=2, end=4, expected=abef	passed	2 ms
[4] str=abcdef, overlay=, start=2, end=4, expected=abef	passed	1 ms
[5] str=abcdef, overlay=zzzz, start=4, end=2, expected=abzzzzef	passed	0 ms
[6] str=abcdef, overlay=zzzz, start=-1, end=4, expected=zzzzef	passed	1 ms
[7] str=abcdef, overlay=zzzz, start=2, end=8, expected=abzzzz	passed	1 ms
[8] str=abcdef, overlay=zzzz, start=-2, end=-3, expected=zzzzabcdef	passed	1 ms
[9] str=abcdef, overlay=zzzz, start=8, end=10, expected=abcdefzzzz	passed	0 ms
Test con stringa e overlay con caratteri speciali, spazi e varie combinazioni di indici.	passed	1 ms
Test con entrambi gli indici negativi.	passed	1 ms
Test con l'overlay vuoto.	passed	1 ms
Test con la stringa vuota.	passed	1 ms
Test con l'overlay null.	passed	1 ms
Test generali dati dalla traccia.	passed	1 ms
Test con la stringa null.	passed	0 ms

Generated by IntelliJ IDEA on 18/01/2024, 19:16

HOMEWORK 2
PROPERTY BASED TESTING

SCELTA CODICE: Per il secondo Homework, abbiamo scelto di utilizzare la stessa classe `Overlay` poiché è idonea al property-based testing. La classe accetta vari tipi di input, tra cui stringhe principali e di sovrapposizione e interi per gli indici di inizio e fine. La presenza di condizioni e controlli, come gestire stringhe nulle o limitare gli indici, offre molte opportunità di test. Inoltre, la manipolazione delle stringhe mediante concatenazione e suddivisione rende la classe adatta a esplorare diversi scenari. La sua complessità e la varietà di input la rendono un eccellente candidato per il property-based testing.

```java
import net.jqwik.api.*;
import net.jqwik.api.constraints.IntRange;
import net.jqwik.api.statistics.Histogram;
import net.jqwik.api.statistics.Statistics;
import net.jqwik.api.statistics.StatisticsReport;
import org.example.Overlay;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class OverlayProprertyTest {
    @Property
    @Report(Reporting.GENERATED)
    @StatisticsReport(format = Histogram.class)
    void OverlayPrPTest(
            /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
            @ForAll ("arbitraryString") String firstString,
            @ForAll ("arbitraryString") String secondString,
            @ForAll @IntRange(min=-1, max=201) Integer start,
            @ForAll @IntRange(min=-1, max=201) Integer end
    ) {
        String s = Overlay.overlay(firstString,secondString,start,end);
        if(s==null){
            Statistics.collect( ...values: "Null Overlay");
        }else{
            if(secondString==null){
                Statistics.collect( ...values: "Added Empty Overlay");
            }
            else if(secondString.contains(" ") || firstString.contains(" ")){
                Statistics.collect( ...values: "Overlay WhiteSpaces");
            }
            else if(checkSpecialChars(secondString) || checkSpecialChars(firstString)){
                Statistics.collect( ...values: "Overlay With Special Chars");
            }
            else{
                Statistics.collect( ...values: "Standard Test Overlay");
            }
        }
    }

    @Property
    @Report(Reporting.GENERATED)
    @StatisticsReport(format = Histogram.class)
    void OverlayPrPTestStartIndex(
            /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
            @ForAll ("arbitraryString") String firstString,
            @ForAll ("arbitraryString") String secondString,
            @ForAll @IntRange(min=-1, max=201) Integer start,
            @ForAll @IntRange(min=-1, max=201) Integer end
```

```
    ) {
        Overlay.overlay(firstString, secondString, start, end);
        if(firstString==null){
            Statistics.collect( ...values: "Null String");
        }
        else if(start>end){
            Statistics.collect( ...values: "Start > End");
        }
        else if(start<0){
            Statistics.collect( ...values: "Start < 0");
        }
        else if(start>firstString.length()){
            Statistics.collect( ...values: "Start > Length");
        }
        else{
            Statistics.collect( ...values: "Valid Start Index");
        }
    }

    @Property
    @Report(Reporting.GENERATED)
    @StatisticsReport(format = Histogram.class)
    void OverlayPrPTestEndInex(
            /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
            @ForAll ("arbitraryString") String firstString,
            @ForAll ("arbitraryString") String secondString,
            @ForAll @IntRange(min=-1, max=201) Integer start,
            @ForAll @IntRange(min=-1, max=201) Integer end
    ) {
        Overlay.overlay(firstString, secondString, start, end);
        if(firstString==null){
            Statistics.collect( ...values: "Null String");
        }
        else if(start>end){
            Statistics.collect( ...values: "End < Start");
        }
        else if(end<0){
            Statistics.collect( ...values: "End < 0");
        }
        else if(end>firstString.length()){
            Statistics.collect( ...values: "End > Length");
        }
        else{
            Statistics.collect( ...values: "Valid End Index");
        }
    }
    no usages
    @Provide
    Arbitrary<String> arbitraryString() {
        return Arbitraries.strings().alpha().numeric().whitespace().ascii().ofMinLength( i: 0).ofMaxLength( i: 200)
                .injectNull( nullProbability: 0.1);
    }

    2 usages
    public static boolean checkSpecialChars(String word) {
        Pattern pattern = Pattern.compile( regex: "[a-zA-Z0-9 \\-_]+", Pattern.MULTILINE);
        Matcher matcher;
        matcher = pattern.matcher(word);
        boolean checker = matcher.matches();
        return !checker;
    }
}
```

1: Esprimere le proprietà da testare:

-Proprietà dell'input:

1. Proprietà sull'Indice di Inizio (start):

1.a) Controlliamo se la stringa principale (`firstString`) è nulla, se è nulla, collezioniamo la statistica "Null String".
1.b) Verifichiamo se l'indice di inizio (`start`) è maggiore dell'indice di fine (`end`), se lo è collezioniamo la statistica "Start > End".
1.c) Verifichiamo se l'indice di inizio (`start`) è minore di zero, se lo è collezioniamo la statistica "Start < 0".
1.d) Verifichiamo se l'indice di inizio (`start`) è maggiore della lunghezza della stringa principale (`firstString`), se lo è collezioniamo la statistica "Start > Length".
1.e) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid Start Index".

2. Proprietà sull'Indice di Fine (end):

2.a) Verifichiamo se l'indice di fine (`end`) è minore dell'indice di inizio (`start`), se lo è collezioniamo la statistica "End < Start".
2.b) Verifichiamo se l'indice di fine (`end`) è minore di zero, se lo è collezioniamo la statistica "End < 0".
2.c) Verifichiamo se l'indice di fine (`end`) è maggiore della lunghezza della stringa principale (`firstString`), se lo è collezioniamo la statistica "End > Length".
2.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid End Index".

Quindi stiamo testando la generazione di overlay nullo, la gestione di stringhe vuote o contenenti spazi bianchi e caratteri speciali e le condizioni sugli indici di inizio e fine in relazione alla lunghezza della stringa principale.

Proprietà dell'output:

3. Proprietà sull'Overlay generato:

3.a) Verifichiamo se l'overlay risultante dalla chiamata a `Overlay.overlay` è nullo, se è nullo, collezioniamo la statistica "Null Overlay".
3.b) Controlliamo se la stringa di sovrapposizione (`secondString`) è vuota o contiene spazi bianchi e se la stringa principale (`firstString`) è vuota o contiene spazi bianchi.
In base a queste condizioni, collezioniamo le statistiche "Added Empty Overlay" e "Overlay WhiteSpaces".
3.c) Verifichiamo se le stringhe di sovrapposizione (`secondString` e `firstString`) contengono caratteri speciali. Se contengono caratteri speciali, collezioniamo la statistica "Overlay With Special Chars".
3.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Standard Test Overlay".

Lasciare che sia il framework a scegliere i test:

```
void OverlayPrPTest(
        /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
        @ForAll ("arbitraryString") String firstString,
        @ForAll ("arbitraryString") String secondString,
        @ForAll @IntRange(min=-1, max=201) Integer start,
        @ForAll @IntRange(min=-1, max=201) Integer end
) {
```

Durante l'esecuzione del property-based testing, il framework genera e passa diversi valori casuali o arbitrari come input al metodo sotto test, come indicato dalle annotazioni @ForAll per i parametri del metodo.
L'annotazione @Property dichiara che il metodo è una proprietà da testare, e il framework esegue il metodo con diverse combinazioni di input per verificare che la proprietà specificata sia valida in una vasta gamma di scenari.
Nel caso specifico del metodo `Overlay.overlay`, che accetta quattro parametri di input, se uno di essi non è valido, la funzione gestisce l'input in modo sicuro senza lanciare eccezioni specifiche.
Ad esempio, se `firstString` è `null`, la funzione restituisce `null`, se gli indici `start` o `end` sono fuori dai limiti, la funzione adatta questi valori per garantire che siano accettabili.
La gestione degli input non validi è progettata per evitare errori senza ricorrere al sollevamento di eccezioni.

1.a) Controlliamo se la stringa principale (`firstString`) è nulla, se è nulla, collezioniamo la statistica "Null String".

```
void OverlayPrPTest(
        /* We generate two arbitrary strings and two integers within
        @ForAll ("arbitraryString") String firstString,
        @ForAll ("arbitraryString") String secondString,
        @ForAll @IntRange(min=-1, max=201) Integer start,
        @ForAll @IntRange(min=-1, max=201) Integer end
) {
    String s = Overlay.overlay(firstString,secondString,start,end);
    if(s==null){
        Statistics.collect( ...values: "Null String");
```

1.a) STATISTICHE: Null String
La categoria "Null String" indica che nel contesto del testing sono state esaminate situazioni in cui le stringhe erano nulle. Questo tipo di test è progettato per verificare come il sistema gestisce le stringhe nulle o se ci sono eventuali problematiche quando si affrontano tali dati.
Il test ha evidenziato 151 casi in cui l'operazione di overlay ha restituito una stringa nulla, indicando la corretta gestione delle stringhe nulle.

```
timestamp = 2024-01-20T19:53:04.373397500, [OverlayProprertyTest:OverlayPrPTest] (1000) statistics =
        # |                      label | count |
    -----|----------------------------|-------|----------------------------------------------------------
        0 |        Added Empty Overlay |    98 | ################
        1 |                Null String |   151 | ########################
        2 |        Overlay WhiteSpaces |   224 | ####################################
        3 | Overlay With Special Chars |   471 | ###############################################################
        4 |      Standard Test Overlay |    56 | #########
```

```
timestamp = 2024-01-20T19:53:04.380425100, OverlayProprertyTest:OverlayPrPTest =
                              |-----------------------jqwik-----------------------
tries = 1000                  | # of calls to property
checks = 1000                 | # of not rejected calls
generation = RANDOMIZED       | parameters are randomly generated
after-failure = SAMPLE_FIRST  | try previously failed sample, then previous seed
when-fixed-seed = ALLOW       | fixing the random seed is allowed
edge-cases#mode = MIXIN       | edge cases are mixed in
edge-cases#total = 1014       | # of all combined edge cases
edge-cases#tried = 177        | # of edge cases tried in current run
seed = 6238734954512023431    | random seed to reproduce generated values
```

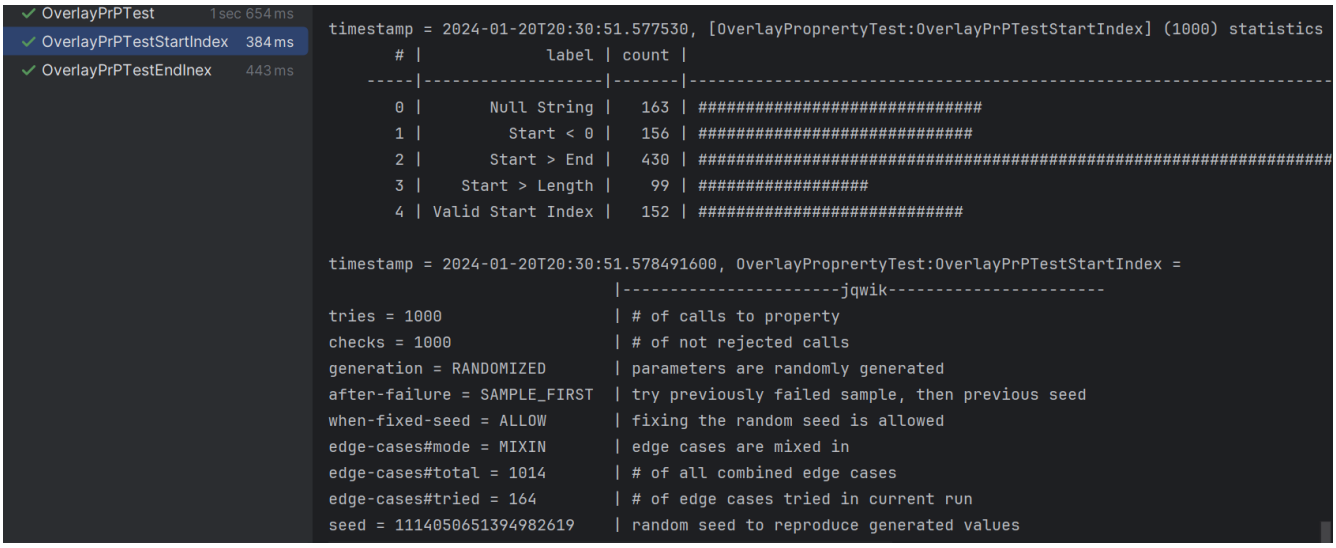
1.b) Verifichiamo se l'indice di inizio (`start`) è maggiore dell'indice di fine (`end`), se lo è collezioniamo la statistica "Start > End".

1.c) Verifichiamo se l'indice di inizio (`start`) è minore di zero, se lo è collezioniamo la statistica "Start < 0".

1.d) Verifichiamo se l'indice di inizio (`start`) è maggiore della lunghezza della stringa principale (`firstString`), se lo è collezioniamo la statistica "Start > Length".

1.e) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid Start Index".

```
43      void OverlayPrPTestStartIndex(
44              /* We generate two arbitrary strings and two integers
45              @ForAll ("arbitraryString") String firstString,
46              @ForAll ("arbitraryString") String secondString,
47              @ForAll @IntRange(min=-1, max=201) Integer start,
48              @ForAll @IntRange(min=-1, max=201) Integer end
49      ) {
50          Overlay.overlay(firstString, secondString, start, end);
51          if(firstString==null){
52              Statistics.collect( ...values: "Null String");
53          }
54          else if(start>end){
55              Statistics.collect( ...values: "Start > End");
56          }
57          else if(start<0){
58              Statistics.collect( ...values: "Start < 0");
59          }
60          else if(start>firstString.length()){
61              Statistics.collect( ...values: "Start > Length");
62          }
63          else{
64              Statistics.collect( ...values: "Valid Start Index");
65          }
66      }
```



1.b) STATISTICHE: Start > End

Verifica la corretta gestione degli indici di inizio e fine durante l'operazione di overlay.
Il test controlla se l'indice di inizio (start) è maggiore dell'indice di fine (end), identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test ha eseguito 1000 chiamate alla proprietà e come visibile dal test, la maggior parte delle chiamate (430) ha prodotto "Start > End", indicando una frequenza significativa di indici di inizio superiori agli indici di fine.

1.c) STATISTICHE: Start < 0

Il test controlla se l'indice di inizio (start) è minore di zero, identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di inizio (start) è inferiore a zero in 156 casi.

1.d) STATISTICHE: Start > Length

Il test controlla se l'indice di inizio (start) è superiore alla lunghezza della prima stringa, identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di inizio (start) è superiore alla lunghezza della prima stringa in 99 casi.

1.e) STATISTICHE: Valid Start Index

Il test controlla che, se non si verificano le condizioni precedenti, l'indice di inizio (start) è considerato valido.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di inizio (start) è stato considerato valido in 152 casi.

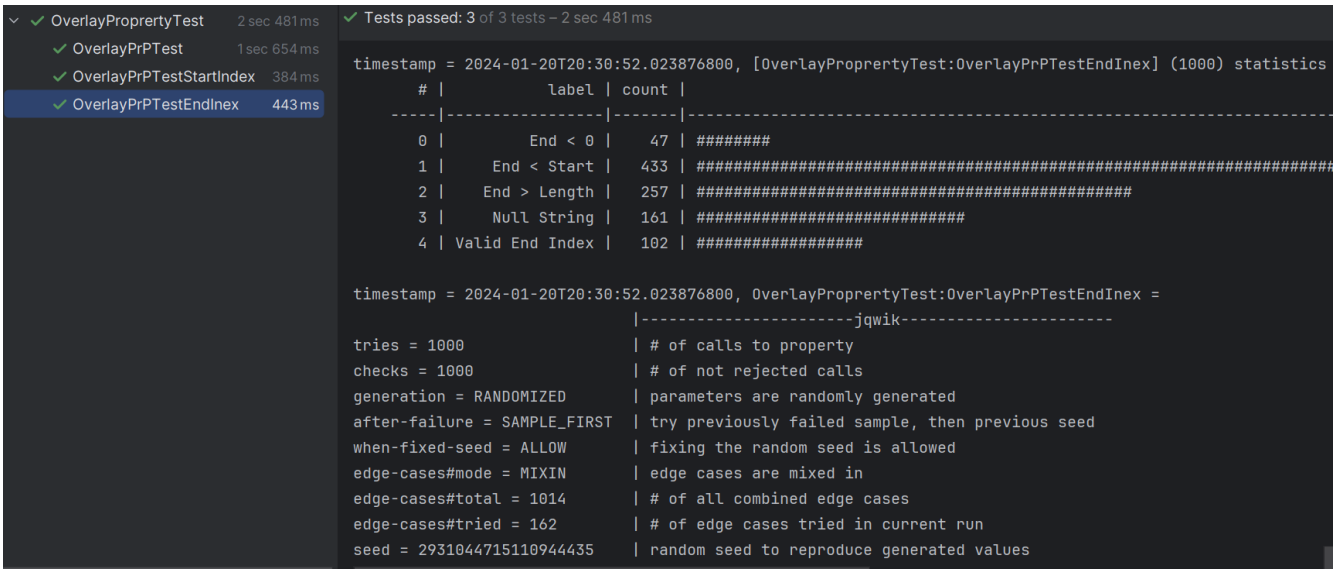
2.a) Verifichiamo se l'indice di fine (`end`) è minore dell'indice di inizio (`start`), se lo è collezioniamo la statistica "End < Start".

2.b) Verifichiamo se l'indice di fine (`end`) è minore di zero, se lo è collezioniamo la statistica "End < 0".

2.c) Verifichiamo se l'indice di fine (`end`) è maggiore della lunghezza della stringa principale (`firstString`), se lo è collezioniamo la statistica "End > Length".

2.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid End Index".

```
70      @StatisticsReport(format = Histogram.class)
71      void OverlayPrPTestEndInex(
72              /* We generate two arbitrary strings and two integers
73              @ForAll ("arbitraryString") String firstString,
74              @ForAll ("arbitraryString") String secondString,
75              @ForAll @IntRange(min=-1, max=201) Integer start,
76              @ForAll @IntRange(min=-1, max=201) Integer end
77      ) {
78          Overlay.overlay(firstString, secondString, start, end);
79          if(firstString==null){
80              Statistics.collect( ...values: "Null String");
81          }
82          else if(start>end){
83              Statistics.collect( ...values: "End < Start");
84          }
85          else if(end<0){
86              Statistics.collect( ...values: "End < 0");
87          }
88          else if(end>firstString.length()){
89              Statistics.collect( ...values: "End > Length");
90          }
91          else{
92              Statistics.collect( ...values: "Valid End Index");
93          }
```



2.a) STATISTICHE: End < Start

Verifica la corretta gestione degli indici di inizio e fine durante l'operazione di overlay.
Il test controlla se l'indice di fine (end) è minore dell'indice di inizio (start), identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test ha eseguito 1000 chiamate alla proprietà e come visibile dal test, la maggior parte delle chiamate (433) ha prodotto "End < Start", indicando una frequenza significativa di indici di fine inferiori agli indici di inizio.

2.b) STATISTICHE: End < 0

Il test controlla se l'indice di fine (end) è minore di zero, identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di fine (end) è inferiore a zero in 47 casi.

2.c) STATISTICHE: End > Length

Il test controlla se l'indice di fine (end) è superiore alla lunghezza della prima stringa, identificando potenziali situazioni in cui gli indici non sono impostati correttamente.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di fine (end) è superiore alla lunghezza della prima stringa in 257 casi.

2.d) STATISTICHE: Valid End Index

Il test controlla che, se non si verificano le condizioni precedenti, l'indice di fine (end) è considerato valido.
Il test è stato eseguito 1000 volte e come visibile dal test, l'indice di fine (end) è stato considerato valido in 102 casi.

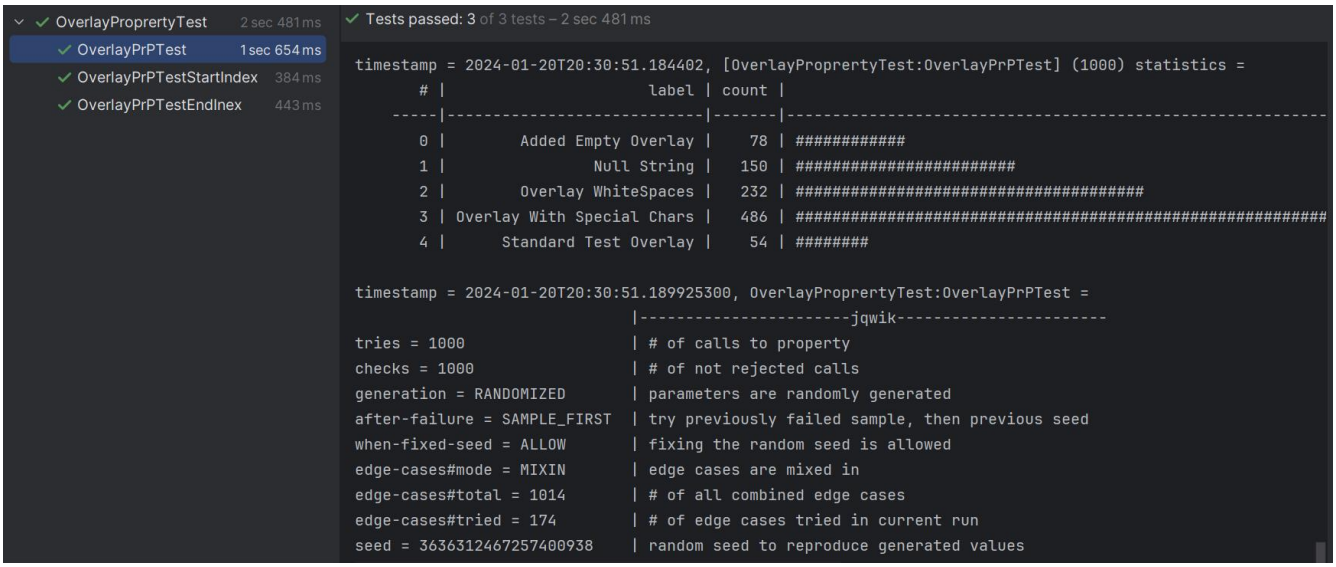
3.a) Verifichiamo se l'overlay risultante dalla chiamata a `Overlay.overlay` è nullo, se è nullo, collezioniamo la statistica "Null Overlay".

3.b) Controlliamo se la stringa di sovrapposizione (`secondString`) è vuota o contiene spazi bianchi e se la stringa principale (`firstString`) è vuota o contiene spazi bianchi.
In base a queste condizioni, collezioniamo le statistiche "Added Empty Overlay" e "Overlay WhiteSpaces".

3.c) Verifichiamo se le stringhe di sovrapposizione (`secondString` e `firstString`) contengono caratteri speciali. Se contengono caratteri speciali, collezioniamo la statistica "Overlay With Special Chars".

3.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Standard Test Overlay".

```
14    void OverlayPrPTest(
15            /* We generate two arbitrary strings and two integers within the range of -1
16            @ForAll ("arbitraryString") String firstString,
17            @ForAll ("arbitraryString") String secondString,
18            @ForAll @IntRange(min=-1, max=201) Integer start,
19            @ForAll @IntRange(min=-1, max=201) Integer end
20    ) {
21        String s = Overlay.overlay(firstString,secondString,start,end);
22        if(s==null){
23            Statistics.collect( ...values: "Null String");
24        }else{
25            if(secondString==null){
26                Statistics.collect( ...values: "Added Empty Overlay");
27            }
28            else if(secondString.contains(" ") || firstString.contains(" ")){
29                Statistics.collect( ...values: "Overlay WhiteSpaces");
30            }
31            else if(checkSpecialChars(secondString) || checkSpecialChars(firstString)){
32                Statistics.collect( ...values: "Overlay With Special Chars");
33            }
34            else{
35                Statistics.collect( ...values: "Standard Test Overlay");
36            }
37        }
```



3.a) STATISTICHE: Null Overlay

Il test controlla quante volte il risultato dell'overlay è una stringa nulla (null).
Il test è stato eseguito 1000 volte e come visibile dal test, In 150 casi, l'overlay ha generato una stringa nulla (s == null), indicando la corretta gestione delle situazioni in cui il risultato dell'overlay è nullo.

3.b) STATISTICHE: Added Empty Overlay" e "Overlay WhiteSpaces

Nel contesto di questo testing sono state esaminate situazioni in cui le stringhe erano vuote o contenente spazi vuoti.
Questo tipo di test è progettato per verificare come il sistema gestisce queste stringhe e se ci sono eventuali problematiche quando si affrontano tali dati.
Il test è stato eseguito 1000 volte e come visibile dal test, in 310 casi sono risultate stringhe vuote o contenente spazi al suo interno, in particolare osserviamo che per 78 volte sono state generate stringhe vuote, mentre per 232 volte sono state generate stringhe che al loro interno contenevano degli spazi.

3.c) STATISTICHE: Overlay With Special Chars

Questo tipo di test è progettato per verificare come il sistema gestisce le stringhe contenenti caratteri speciali e se ci sono eventuali problematiche quando si affrontano tali dati.
Il test è stato eseguito 1000 volte e come visibile dal test, per ben 486 casi sono risultate stringhe contenenti caratteri speciali.

3.d) STATISTICHE: Standard Test Overlay

La categoria "Standard Test Overlay" nel contesto del test di proprietà indica una situazione in cui l'operazione di overlay tra due stringhe con indici specifici produce un risultato standard, che non ricade nelle altre categorie specificate in precedenza.
In altre parole, "Standard Test Overlay" potrebbe indicare che, in questi casi specifici, l'operazione di overlay tra le due stringhe con gli indici forniti ha prodotto un risultato che non presenta particolari caratteristiche speciali. Il termine "standard" qui potrebbe essere interpretato come il comportamento atteso o normale dell'operazione di overlay in queste circostanze.
Osserviamo che solamente 54 volte abbiamo ottenuto queste tipologie di stringhe.

- STATISTICA – UTILIZZO DEL GRAFICO:

L'utilizzo di un'istogramma come strumento di supporto nei property test, è utile poiché il grafico visuale consente di esaminare la distribuzione dei dati generati durante i test, identificando pattern, individuando valori estremi e verificando il rispetto di proprietà specifiche. L'analisi dell'istogramma contribuisce all'ottimizzazione dei parametri di generazione dei dati, migliorando l'efficacia dei test nel rilevare eventuali problemi nel sistema. Inoltre, la chiara rappresentazione visiva semplifica la comunicazione dei risultati. In sintesi, l'istogramma costituisce uno strumento prezioso per una valutazione approfondita e una comunicazione efficace dei risultati dei property test.

- CODE COVERAGE:

Questa è la code coverage per quanto riguarda la classe associata all'homework2 che effettua i property test. Come si può notare dalle statistiche, la line e la branch coverage sono coperte al 100%.