



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:	4
7 STEP:	4
1- Comprendere i requisiti (cosa deve fare il programma, input e output).	4
2- Esplora cosa fa il programma per vari input.	4
3- Esplorare input, output ed identificare le partizioni:	4
4- Casi limite:	5
5- Definire i casi di test:	6
6- Automatizzare casi di test:	6
7- Aumentare la suite con creatività ed esperienza:	9
LEGGERE L'IMPLEMENTAZIONE:	10
ESEGUIRE TEST CON TOOL DI CODE COVERAGE:	11
HOMEWORK 2:	13
PROPERTY BASED TESTING	13
1: Esprimere le proprietà da testare:	15
-Proprietà dell'input:	15
1. Proprietà sull'Indice di Inizio (start):	15
2. Proprietà sull'Indice di Fine (end):	15
Proprietà dell'output:	15
3. Proprietà sull'Overlay generato:	15
Lasciare che sia il framework a scegliere i test:	16
1.a) Controlliamo se la stringa principale ('firstString') è nulla, se è nulla, collezioniamo la statistica "Null String".	16
1.b) Verifichiamo se l'indice di inizio ('start') è maggiore dell'indice di fine ('end'), se lo è collezioniamo la statistica "Start > End".	17
1.c) Verifichiamo se l'indice di inizio ('start') è minore di zero, se lo è collezioniamo la statistica "Start < 0".	17
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".	17
1.e) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid Start Index".	17
1.b) STATISTICHE: Start > End	18
1.c) STATISTICHE: Start < 0	18
1.d) STATISTICHE: Start > Length	18
1.e) STATISTICHE: Valid Start Index	18
2.a) Verifichiamo se l'indice di fine ('end') è minore dell'indice di inizio ('start'), se lo è collezioniamo la statistica "End < Start".	19
2.b) Verifichiamo se l'indice di fine ('end') è minore di zero, se lo è collezioniamo la statistica "End < 0".	19
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".	19
2.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Valid End Index".	19
2.a) STATISTICHE: End < Start	20
2.b) STATISTICHE: End < 0	20
2.c) STATISTICHE: End > Length	20



2.d) STATISTICHE: Valid End Index	20
3.a) Verifichiamo se l'overlay risultante dalla chiamata a <code>Overlay.overlay</code> è nullo, se è nullo, collezioniamo la statistica "Null Overlay"	21
3.b) Controlliamo se la stringa di sovrapposizione (<code>secondString</code>) è vuota o contiene spazi bianchi e se la stringa principale (<code>firstString</code>) è vuota o contiene spazi bianchi.	21
3.c) Verifichiamo se le stringhe di sovrapposizione (<code>secondString</code> e <code>firstString</code>) contengono caratteri speciali.	21
3.d) Se tutte le condizioni precedenti non sono soddisfatte, collezioniamo la statistica "Standard Test Overlay".	21
3.a) STATISTICHE: Null Overlay	22
3.b) STATISTICHE: Added Empty Overlay" e "Overlay WhiteSpaces	22
3.c) STATISTICHE: Overlay With Special Chars	22
3.d) STATISTICHE: Standard Test Overlay	22



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.

```
//10
@Test
@DisplayName("Test generali dati dalla traccia.")
public void overlaySampleTest() {
    assertEquals(Overlay.overlay( str: "", overlay: "abc", start: 0, end: 0), actual: "abc");//se il primo p
    assertEquals(Overlay.overlay( str: "abcdef", overlay: null, start: 2, end: 4), actual: "abef");//se il se
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "", start: 2, end: 4), actual: "abef");//se il seco
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "", start: 4, end: 2), actual: "abef");//se start è
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: 2, end: 4), actual: "abzzzzef");//1
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: 4, end: 2), actual: "abzzzzef");//se
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: -1, end: 4), actual: "zzzzabef");// se
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: 2, end: 8), actual: "abzzzz");//se a
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: -2, end: -3), actual: "zzzzabcdef");//
    assertEquals(Overlay.overlay( str: "abcdef", overlay: "zzzz", start: 8, end: 10), actual: "abcdefzzzz");//
}
```

✓ OverlayTest 80 ms ✓ Tests passed: 1 of 1 test – 80 ms
✓ Test generali dati dalla traccia. 80 ms

➔ Come si può notare, il test è 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” è 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:

```
//T0
@Test
@DisplayName("Test generali dati dalla traccia.")
public void overlaySampleTest() {
    assertEquals(Overlay.overlay(str: "", overlay: "abc", start: 0, end: 0), actual: "abc");//se il primo pa
    assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 2, end: 4), actual: "abef");//se il se
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 2, end: 4), actual: "abef");//se il seco
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 4, end: 2), actual: "abef");//se start è
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: 2, end: 4), actual: "abzzzzef");//\
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: 4, end: 2), actual: "abzzzzef");//se
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: -1, end: 4), actual: "zzzzef");// se
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: 2, end: 8), actual: "abzzzz");//se e
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: -2, end: -3), actual: "zzzzabcdef");//
    assertEquals(Overlay.overlay(str: "abcdef", overlay: "zzzz", start: 8, end: 10), actual: "abcdefzzzz");//
}
```

✓ OverlayTest 80 ms ✓ Tests passed: 1 of 1 test – 80 ms

✓ Test generali dati dalla traccia. 80 ms

- T1 - Stringa nulla:

```
29 //T1
30 @Test
31 @DisplayName("Test con la stringa null.")
32 public void stringWithNullString() {
33     //Se la stringa è null, l'output atteso è null.
34     assertNull(Overlay.overlay(str: null, overlay: "mao", start: 3, end: 1));
35     assertNull(Overlay.overlay(str: null, overlay: "mao", start: -2, end: -1));
36     assertNull(Overlay.overlay(str: null, overlay: "mao", start: 10, end: 12));
37     assertNull(Overlay.overlay(str: null, overlay: "mao", start: 4, end: 6));
38     assertNull(Overlay.overlay(str: null, overlay: "mao", start: 0, end: 0));
39 }
```

✓ OverlayTest 57 ms ✓ Tests passed: 1 of 1 test – 57 ms

✓ Test con la stringa null. 57 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

Process finished with exit code 0



- T2 - Test con l'overlay null:

```
41 //T2
42 @Test
43 @DisplayName("Test con l'overlay null.")
44 public void overlayWithNullString() {
45     //Se l'overlay è null, l'output atteso è la stringa troncata tra gli indici start e end.
46     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 3, end: 1), actual: "adef");
47     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: -2, end: -1), actual: "abcdef");
48     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 10, end: 12), actual: "abcdef");
49     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 4, end: 6), actual: "abcd");
50     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 0, end: 0), actual: "abcdef");
51     assertEquals(Overlay.overlay(str: "abcdef", overlay: null, start: 6, end: 6), actual: "abcdef");
52 }
```

✓ OverlayTest 37 ms Tests passed: 1 of 1 test - 37 ms

✓ Test con l'overlay null. 37 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

Process finished with exit code 0

- T3 - Test con la stringa vuota:

```
56 //T3
57 @Test
58 @DisplayName("Test con la stringa vuota.")
59 public void stringWithEmptyString() {
60     //Se la stringa è vuota, l'output atteso è pari all'overlay
61     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: 3, end: 1), actual: "mao");
62     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: -2, end: -1), actual: "mao");
63     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: 10, end: 12), actual: "mao");
64     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: 4, end: 6), actual: "mao");
65     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: 0, end: 0), actual: "mao");
66 }
```

✓ OverlayTest 29 ms Tests passed: 1 of 1 test - 29 ms

✓ Test con la stringa vuota. 29 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

Process finished with exit code 0

- T4 - Test con l'overlay vuoto:

```
68 //T4
69 @Test
70 @DisplayName("Test con l'overlay vuoto.")
71 public void overlayWithEmptyString() {
72     //Se l'overlay è vuoto, l'output atteso è pari alla stringa
73     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 3, end: 1), actual: "adef");
74     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: -2, end: -1), actual: "abcdef");
75     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 10, end: 12), actual: "abcdef");
76     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 4, end: 6), actual: "abcd");
77     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 0, end: 0), actual: "abcdef");
78     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: 6, end: 6), actual: "abcdef");
79 }
```

✓ OverlayTest 39 ms Tests passed: 1 of 1 test - 39 ms

✓ Test con l'overlay vuoto. 39 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

Process finished with exit code 0



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

```
90 //T5
91 @Test
92 @DisplayName("Test con entrambi gli indici negativi.")
93 public void overlayWithNegativeNumber() {
94     //Start e end se sono negativi, vengono considerati come zero.
95     assertEquals(Overlay.overlay(str: "abcdef", overlay: "mao", start: -1, end: -2), actual: "maoabcdef");
96     assertEquals(Overlay.overlay(str: "abcdef", overlay: "mao", start: -7, end: -8), actual: "maoabcdef");
97     assertEquals(Overlay.overlay(str: "", overlay: "", start: -1, end: -2), actual: "");
98     assertEquals(Overlay.overlay(str: "abcdef", overlay: "", start: -1, end: -2), actual: "abcdef");
99     assertEquals(Overlay.overlay(str: "", overlay: "mao", start: -1, end: -2), actual: "mao");
100     assertEquals(Overlay.overlay(str: "abcdef", overlay: "mao", start: -8, end: -2), actual: "maoabcdef");
101     assertEquals(Overlay.overlay(str: "abcdef", overlay: "mao", start: -1, end: -8), actual: "maoabcdef");
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
```

✓ OverlayTest 33 ms ✓ Tests passed: 1 of 1 test - 33 ms

✓ Test con entrambi gli indici negativi. 33 ms

C:\Program Files\Java\jdk-21\bin\java.exe ...

Process finished with exit code 0

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

```
94 //T6
95 @Test
96 @DisplayName("Test con stringa e overlay di un solo carattere e varie combinazioni di indici.")
97 public void overlayWithStringLen1() {
98     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: -1, end: -2), actual: "ba");
99     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: 1, end: 1), actual: "ab");
100     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: 0, end: 0), actual: "ba");
101     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: 2, end: 2), actual: "ab");
102     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: 1, end: 2), actual: "ab");
103     assertEquals(Overlay.overlay(str: "a", overlay: "b", start: 0, end: 1), actual: "b");
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
```

✓ OverlayTest 35 ms ✓ Tests passed: 1 of 1 test - 35 ms

✓ Test con stringa e overlay di un solo carattere e varie combinazioni di indici. 35 ms

C:\Program Files\Java\jdk-21\bin\java.exe ...

Process finished with exit code 0



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.

```
106 //T7
107 @Test
108 @DisplayName("Test con stringa e overlay con caratteri speciali, spazi e varie combinazioni di indici.")
109 public void overlayWithSpecialChar() {
110     assertEquals(Overlay.overlay(str: "à sda", overlay: "vf", start: -1, end: -2), actual: "vfà sda");
111     assertEquals(Overlay.overlay(str: "asdds", overlay: "èasd", start: 5, end: 5), actual: "asddsèasd");
112     assertEquals(Overlay.overlay(str: "asd -.", overlay: "asd", start: 0, end: 0), actual: "asdasd -.");
113     assertEquals(Overlay.overlay(str: "asddere", overlay: "--.sad", start: 8, end: 8), actual: "asddere--.sad");
114     assertEquals(Overlay.overlay(str: "@#sdasde", overlay: "asd", start: 1, end: 2), actual: "@asdsdasde");
115     assertEquals(Overlay.overlay(str: "#eeras", overlay: "####", start: 0, end: 1), actual: "####eeras");
116 }
117
118 OverlayTest 36ms ✓ Tests passed: 1 of 1 test - 36 ms
119 ✓ Test con stringa e overlay con caratteri speciali, spazi e varie combinazioni di indici. 36ms
120 "C:\Program Files\Java\jdk-21\bin\java.exe" ...
121 Process finished with exit code 0
```

- T8 - Test parametrico: provideOverlayTestData

```
118 @ParameterizedTest
119 @MethodSource("provideOverlayTestData")
120 @DisplayName("Test Parametrico.")
121 public void testOverlay(String str, String overlay, int start, int end, String expected) {
122     String result = Overlay.overlay(str, overlay, start, end);
123     assertEquals(expected, result);
124 }
125
126 @
127 private static Stream<Arguments> provideOverlayTestData() {
128     return Stream.of(
129         Arguments.of(...arguments: null, "*", 2, 4, null),
130         Arguments.of(...arguments: "", "abc", 0, 0, "abc"),
131         Arguments.of(...arguments: "abcdef", null, 2, 4, "abef"),
132         Arguments.of(...arguments: "abcdef", "", 2, 4, "abef"),
133         Arguments.of(...arguments: "abcdef", "zzzz", 4, 2, "abzzzzef"),
134         Arguments.of(...arguments: "abcdef", "zzzz", -1, 4, "zzzzef"),
135         Arguments.of(...arguments: "abcdef", "zzzz", 2, 8, "abzzzz"),
136         Arguments.of(...arguments: "abcdef", "zzzz", -2, -3, "zzzzabcdef"),
137         Arguments.of(...arguments: "abcdef", "zzzz", 8, 10, "abcdefzzzz")
138     );
139 }
140
141 OverlayTest 84ms ✓ Tests passed: 9 of 9 tests - 84 ms
142 ✓ Test Parametrico. 84ms
143 ✓ [1] str=null, overlay=*, start=2, end=4, expected=null 61ms
144 ✓ [2] str=, overlay=abc, start=0, end=0, expected=abc 11ms
145 ✓ [3] str=abcdef, overlay=null, start=2, end=4, expected=abef 1ms
146 ✓ [4] str=abcdef, overlay=, start=2, end=4, expected=abef 4ms
147 ✓ [5] str=abcdef, overlay=zzzz, start=4, end=2, expected=abzzzzef 1ms
148 ✓ [6] str=abcdef, overlay=zzzz, start=-1, end=4, expected=zzzzef 2ms
149 ✓ [7] str=abcdef, overlay=zzzz, start=2, end=8, expected=abzzzz 2ms
150 ✓ [8] str=abcdef, overlay=zzzz, start=-2, end=-3, expected=zzzzabcdef 1ms
151 ✓ [9] str=abcdef, overlay=zzzz, start=8, end=10, expected=abcdefzzzz 1ms
152 "C:\Program Files\Java\jdk-21\bin\java.exe" ...
153 Process finished with exit code 0
```



LEGGERE L'IMPLEMENTAZIONE:

```
28 public static String overlay(final String str, String overlay, int start, int end) {
29     //verifico se la stringa è nulla
30     if (str == null) {
31         return null;
32     }
33     //verifico se l'overlay è nullo
34     if (overlay == null) {
35         overlay = "";
36     }
37     //prendo la lunghezza della stringa
38     final int len = str.length();
39     //verifico se start è minore di zero, altrimenti lo azzero
40     if (start < 0) {
41         start = 0;
42     }
43     //verifico se start è maggiore della lunghezza, altrimenti lo pongo uguale alla lunghezza
44     if (start > len) {
45         start = len;
46     }
47     //verifico se end è minore di zero, altrimenti lo azzero
48     if (end < 0) {
49         end = 0;
50     }
51     //verifico se end è maggiore della lunghezza, altrimenti lo pongo uguale alla lunghezza
52     if (end > len) {
53         end = len;
54     }
55     //verifico se start è maggiore della fine, inverto start con end
56     if (start > end) {
57         final int temp = start;
58         start = end;
59         end = temp;
60     }
61     //restituisco l'overlay delle stringhe
62     return str.substring(0, start) +
63         overlay + str.substring(end);
64     }
65 }
```



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.

Coverage OverlayTest x				
Element ^				
	Class, %	Method, %	Line, %	Branch, %
✓ org.example	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
Ⓢ Overlay	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)

Cover overlayTest x

Test Results 70ms

- ✓ Test class overlayTest 70ms
- ✓ Test con stringa e overlay di un solo carattere e varie co 31ms
- ✓ Test Parametrico. 33ms
 - ✓ [1] str=null, overlay=*, start=2, end=4, expected=nul 28ms
 - ✓ [2] str=, overlay=abc, start=0, end=0, expected=abc 1ms
 - ✓ [3] str=abcdef, overlay=null, start=2, end=4, expectec 2ms
 - ✓ [4] str=abcdef, overlay=, start=2, end=4, expected=at 1ms
 - ✓ [5] str=abcdef, overlay=zzzz, start=4, end=2, expecte 0ms
 - ✓ [6] str=abcdef, overlay=zzzz, start=-1, end=4, expecte 1ms
 - ✓ [7] str=abcdef, overlay=zzzz, start=-2, end=8, expecte 1ms
 - ✓ [8] str=abcdef, overlay=zzzz, start=-2, end=-3, expect 1ms
 - ✓ [9] str=abcdef, overlay=zzzz, start=8, end=10, expect 0ms
- ✓ Test con stringa e overlay con caratteri speciali, spazi e vi 1ms
- ✓ Test con entrambi gli indici negativi. 1ms
- ✓ Test con l'overlay vuoto. 1ms
- ✓ Test con la stringa vuota. 1ms
- ✓ Test con l'overlay null. 1ms
- ✓ Test generali dati dalla traccia. 1ms
- ✓ Test con la stringa null. 0ms

Tests passed: 17 of 17 tests - 70ms

- > Task :compileJava UP-TO-DATE
- > Task :processResources NO-SOURCE
- > Task :classes UP-TO-DATE
- > Task :compileTestJava UP-TO-DATE
- > Task :processTestResources NO-SOURCE
- > Task :testClasses UP-TO-DATE
- > Task :test

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.4/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 2s

3 actionable tasks: 1 executed, 2 up-to-date

18:57:17: Execution finished ':test --tests "overlayTest"'.

hitted > src > main > java > org > example > Overlay > overlay



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



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.

```
1  import net.jqwik.api.*;
2  import net.jqwik.api.constraints.IntRange;
3  import net.jqwik.api.statistics.Histogram;
4  import net.jqwik.api.statistics.Statistics;
5  import net.jqwik.api.statistics.StatisticsReport;
6  import org.example.Overlay;
7  import java.util.regex.Matcher;
8  import java.util.regex.Pattern;
9
10 public class OverlayPropertyTest {
11     @Property
12     @Report(Reporting.GENERATED)
13     @StatisticsReport(format = Histogram.class)
14     void OverlayPrPTest(
15         /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
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 Overlay");
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         }
38     }
39
40     @Property
41     @Report(Reporting.GENERATED)
42     @StatisticsReport(format = Histogram.class)
43     void OverlayPrPTestStartIndex(
44         /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
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     }
67
68     @Property
69     @Report(Reporting.GENERATED)
70     @StatisticsReport(format = Histogram.class)
71     void OverlayPrPTestEndIndex(
72         /* We generate two arbitrary strings and two integers within the range of -1 to 200, inclusively. */
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         }
94     }
95
96     no usages
97     @Provide
98     Arbitrary<String> arbitraryString() {
99         return Arbitraries.strings().alpha().numeric().whitespace().ascii().ofMinLength(1).ofMaxLength(200)
100             .injectNull( nullProbability: 0.1);
101     }
102
103     2 usages
104     public static boolean checkSpecialChars(String word) {
105         Pattern pattern = Pattern.compile( regex: "[a-zA-Z0-9 \\- _]", Pattern.MULTILINE);
106         Matcher matcher;
107         matcher = pattern.matcher(word);
108         boolean checker = matcher.matches();
109         return !checker;
110     }
```



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, [OverlayPropertyTest: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, OverlayPropertyTest: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 }
```



```
✓ OverlayPrPTest 1sec 654 ms
✓ OverlayPrPTestStartIndex 384 ms
✓ OverlayPrPTestEndIndex 443 ms

timestamp = 2024-01-20T20:30:51.577530, [OverlayPropertyTest:OverlayPrPTestStartIndex] (1000) statistics
# | label | count |
---|-----|-----|
0 | Null String | 163 | #####
1 | Start < 0 | 156 | #####
2 | Start > End | 430 | #####
3 | Start > Length | 99 | #####
4 | Valid Start Index | 152 | #####

timestamp = 2024-01-20T20:30:51.578491600, OverlayPropertyTest:OverlayPrPTestStartIndex =
|-----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 = 164 | # of edge cases tried in current run
seed = 1114050651394982619 | random seed to reproduce generated values
```

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     }
```



```
✓ OverlayPropertyTest 2 sec 481 ms
✓ OverlayPrPTest 1 sec 654 ms
✓ OverlayPrPTestStartIndex 384 ms
✓ OverlayPrPTestEndInex 443 ms

Tests passed: 3 of 3 tests - 2 sec 481 ms

timestamp = 2024-01-20T20:30:52.023876800, [OverlayPropertyTest:OverlayPrPTestEndInex] (1000) statistics
# | label | count |
---|-----|-----|
0 | End < 0 | 47 | #####
1 | End < Start | 433 | #####
2 | End > Length | 257 | #####
3 | Null String | 161 | #####
4 | Valid End Index | 102 | #####

timestamp = 2024-01-20T20:30:52.023876800, OverlayPropertyTest:OverlayPrPTestEndInex =
|-----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 = 162 | # of edge cases tried in current run
seed = 2931044715110944435 | random seed to reproduce generated values
```

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     }
```



```
✓ OverlayPropertyTest 2 sec 481 ms
✓ OverlayPrPTest 1 sec 654 ms
✓ OverlayPrPTestStartIndex 384 ms
✓ OverlayPrPTestEndIndex 443 ms

Tests passed: 3 of 3 tests - 2 sec 481 ms

timestamp = 2024-01-20T20:30:51.184402, [OverlayPropertyTest:OverlayPrPTest] (1000) statistics =
# | label | count |
---|-----|-----|
0 | Added Empty Overlay | 78 | #####
1 | Null String | 150 | #####
2 | Overlay WhiteSpaces | 232 | #####
3 | Overlay With Special Chars | 486 | #####
4 | Standard Test Overlay | 54 | #####

timestamp = 2024-01-20T20:30:51.189925300, OverlayPropertyTest: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 = 174 | # of edge cases tried in current run
seed = 3636312467257400938 | random seed to reproduce generated values
```

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%.

Element	Class, %	Method, %	Line, %	Branch, %
org.example	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
Overlay	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)

Element	Class, %	Method, %	Line, %	Branch, %
OverlayPropertyTest	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
OverlayPrPTest	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
OverlayPrPTestStartIndex	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
OverlayPrPTestEndIndex	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)

7 sec 921ms
edge-cases#tried = 169
seed = 801197196638928921
Process finished with exit code 0

TestPropertyBased: 3 total, 3 passed			7.92 s
OverlayPrPTest	passed	4.35 s	
OverlayPrPTestStartIndex	passed	2.06 s	
OverlayPrPTestEndIndex	passed	1.49 s	

Generated by IntelliJ IDEA on 21/01/24, 15:14