Integrazione e Test di Sistemi Software 2024



Cristian Pazienza ITPS 775695

Sommario

Specification test	3
Step 1: Understanding the requirements	3
Step 2: Explore what the program does for various inputs	4
Step 3: Explore inputs, outputs and identify partitions	5
Step 4: Identify boundary cases	6
Step 5: Devise test cases	7
Step 6: Automate test cases	8
Step 7: Augment the test suite with creativity and experience	10
Structural testing	11
Code coverage	11
Mutation test	11

Specification test

Step 1: Understanding the requirements

Il metodo maskAndFillToLength() modifica una stringa applicando una maschera su determinati caratteri a intervalli definiti e ripetendo la stringa fino a raggiungere una lunghezza massima.

Il metodo riceve quattro parametri in input:

- 1. str: la stringa di input per le modifiche
- 2. maskChar: un carattere per effettuare la mascherazione
- 3. interval: l'intervallo dei caratteri
- 4. maxLength: la lunghezza finale della stringa

Il metodo restituisce:

- 1. Una stringa modificata
- 2. Null se non vengono sodisfatti le precondizioni

Step 2: Explore what the program does for various inputs

Il metodo andrà ad eseguire queste azioni

- 1. Maschera i caratteri a intervalli definiti.
- 2. Ripete la stringa modificata fino a raggiungere maxLength.
- 3. Restituisce null in caso di:
 - o strè null.
 - o maxLength <= 0.</p>
 - o interval <= 0.</p>

Eseguo dei semplici test per verificare se il metodo funzioni correttamente

```
import Esame.MaskAndRepeatUtils;
 import org.junit.jupiter.api.Assertions;
 import org.junit.jupiter.api.Test;
public class SimpleExamTest {
                            @Test
                            public void simpleTest() {
                                                           Assertions.assertEquals(MaskAndRepeatUtils.maskAndFillToLength("abc", '*', 2, 6),
  "a*ca*c");
                            }
                            @Test
                            public void simpleTest2() {
                                                           Assertions.assertEquals(MaskAndRepeatUtils.maskAndFillToLength("abc", '*', 2, 4),
   "a*ca");
                              }
                  SimpleExamTest,simpleTest and 1 more: 2 total, 2 passed
                      "C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java exe" -ea -Didea test.cyclic.buffer.size=1048576 "-Javaagent:C:\Program Files\LetBrains\IntelliJ IDEA 2024.2.4\libidea_rt_jar=55401:C:\Program Files\LetBrains\IntelliJ IDEA 2024.2.4\libidea_rt_jar=55401:C:\
                    1.3.0 jar.C:\Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\mathcal{Eq:Users\Cristan\\Cristan\Cristan\\Cristan\\Cristan\\Cristan\Cristan\\Cristan\Cristan\\Cristan\\Cristan\Cristan\\Cristan\Cristan\Cristan\C
                               SimpleExamTest
                                          simpleTest()
                                          simpleTest2()
 Generated by IntelliJ IDEA on 12/11/24, 18:21
```

Con questo semplice test osservo che il metodo funzioni come mi aspetto

Step 3: Explore inputs, outputs and identify partitions

Le classi di test per l'input sono:

- Str:
 - 1. Una stringa vuota
 - 2. Null
 - 3. Lunghezza della stringa 1
 - 4. Lunghezza della stringa maggiore 1
- maskChar
 - 1. Un singolo carattere
 - 2. Nessun carattere
- Interval
 - 1. Positivo
 - 2. Negativo
- maxLength
 - 1. Positivo
 - 2. Negativo

Le combinazioni di input sono:

maxLenght < lunghezza str

interval > lunghezza str

interval > lunghezza str e maxLenght < lunghezza str

interval > lunghezza str e maxLenght > lunghezza str

I test totali sono 4x2x2x2x4= 128 test

Step 4: Identify boundary cases

Andiamo a identificare i boundary cases:

if (str == null || maxLength <= 0 || interval <= 0)
 per str:
 on point: null, rende la condizione vera quindi sarà anche in point
 off point: lunghezza di str >= 1, rende la condizione falsa quindi sarà anche out
 point
 per maxLength:
 on point: 0, rende la condizione vera quindi sarà anche in point
 off point: -1, rende la condizione falsa quindi sarà anche out point
 per interval:
 on point: 0, rende la condizione vera quindi sarà anche in point
 off point: -1, rende la condizione falsa quindi sarà anche out point

- if(str.isEmpty())
 on point: empty, rende la condizione vera, quindi, sarà anche in point
 off point: lunghezza di str >= 1, rende la condizione falsa quindi sarà anche out
 point
- if ((i + 1) % interval == 0)
 on point: 0, rende la condizione vera, quindi, sarà anche in point
 off point: (i + 1) % interval == 0 + 1, rende la condizione falsa quindi sarà anche out point
- if (maskedStr.length() >= maxLength)
 on point: maxLength, rende la condizione vera, quindi, sarà anche in point
 T24: of point: maxLength-1, rende la condizione falsa quindi sarà anche out point
- if (result.length() < maxLength)
 on point: maxLength, rende la condizione falsa
 off point: maxLength+1, rende la condizione vera
 in point: maxLength+1
 out point: maxLength

Step 5: Devise test cases

Testo un sottoinsieme di possibili test andando ad ottimizzare la suite di test

T1: str empty

T2: str null

T3: 1 char

T4: empty char

T5: interval negative

T6: maxLength negative

T7: maxLenght < lunghezza str

T8: interval > lunghezza str e maxLenght < lunghezza str

T9: interval > lunghezza str e maxLenght > lunghezza str

Andando a testare 9 casi e non 128.

Step 6: Automate test cases

}

Utilizzo i test parametrici per evitare duplicazione di codice all'interno della suite di test

```
public class MaskAndRepeatUtilsTest {
  @ParameterizedTest
  @MethodSource("generator")
  @DisplayName("I test the maskAndFillToLength method of the MaskAndRepeatUtils
class")
  public void testMaskAndRepeat(final String str, final char maskChar, final int interval,
final int maxLength, String expected) {
    Assertions.assertEquals(MaskAndRepeatUtils.maskAndFillToLength(str, maskChar,
interval, maxLength), expected);
  }
  static Stream<Arguments> generator() {
    return Stream.of(
         of("", "*", 1, 3, ""),
         of(null, "*", 1, 3, null),
         of("pippo", "*", 2, 5, "p*p*o"),
         of("pippo", " ", 2, 5, "p p o"),
         of("", "*", -1, 3, null),
         of("", "*", 1, -3, null),
         of("Ciao", "X", 2, 3, "CXa"),
         of("Ciao", "X", 5, 2, "Ci"),
         of("Ciao", "X", 5, 6, "CiaoCi")
    );
```

	Collapse
MaskAndRepeatUtilsTest	
I test the maskAndFillToLength method of the MaskAndRepeatUtils class	
[1] , *, 1, 3,	passed
[2] null, *, 1, 3, null	passed
[3] pippo, *, 2, 5, p*p*o	passed
[4] pippo, , 2, 5, p p o	passed
[5] , -^, -1, 3, null	passed
[6] , *, 1, -3, null	passed
[7] Ciao, X, 2, 3, CXa	passed
[8] Ciao, X, 5, 2, Ci	passed
[9] Ciao, X, 5, 6, CiaoCi	passed

I test passano tutti con successo

Step 7: Augment the test suite with creativity and experience

Aggiungo un test dove char sarà un carattere speciale

T10: char = "§"

Un test in cui interval è il numero di macchina più grande

T11: interval = MAX VALUE

Un test in cui interval è il numero di macchina più piccolo

T12: interval = MIN VALUE

Un test in cui maxLength è il numero di macchina più piccolo

T13: maxLength = MIN_VALUE

Creo anche dei test per le combinazioni di interval e maxLength in cui sono

T14: maxLength = MAX_VALUE e interval = MIN_VALUE

T15: maxLength = MIN_VALUE e interval = MAX_VALUE

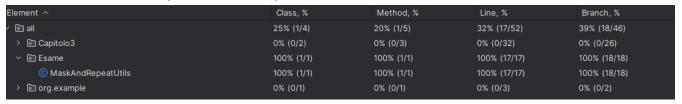


I test passano tutti con successo

Structural testing

Code coverage

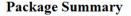
Utilizzo la code coverage di intellij per controllare se i test coprono tutte le righe di codice e i branch, in questo caso copre il 100% di entrambe

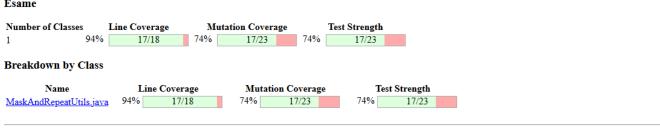


Mutation test

Utilizzato il framework **PITest** <u>per</u> effettuare i mutation test. Osserviamo che la suite di test sopravvive a quasi tutti i mutanti, questo indica che la suite è robusta.

Pit Test Coverage Report





Report generated by PIT 1.15.8

```
Mutations

1. changed conditional boundary → KILLED
2. changed conditional boundary → KILLED
3. negated conditional → KILLED
4. negated conditional → KILLED
5. negated conditional → KILLED
17. 1. replaced return value with "" for Esame/MaskAndRepeatUtils::maskAndFillToLength → KILLED
18. 1. negated conditional → TIMED_OUT
19. 1. replaced return value with "" for Esame/MaskAndRepeatUtils::maskAndFillToLength → SURVIVED
19. 1. changed conditional boundary → KILLED
20. 1. negated conditional → TIMED_OUT
21. replaced integer addition with subtraction → KILLED
22. negated conditional → TIMED_OUT
23. negated conditional → KILLED
24. Replaced integer modulus with multiplication → KILLED
25. negated conditional → KILLED
26. negated conditional → KILLED
27. negated conditional → KILLED
28. negated conditional → KILLED
29. negated conditional → KILLED
20. negated conditional → KILLED
21. changed conditional → KILLED
22. Replaced integer addition with subtraction → KILLED
23. negated conditional → KILLED
24. negated conditional → KILLED
25. negated conditional → KILLED
26. negated conditional → KILLED
27. negated conditional → KILLED
28. Replaced integer addition with subtraction → KILLED
39. negated conditional → KILLED
40. negated conditional → KILLED
41. negated conditional → KILLED
42. negated conditional → KILLED
43. negated conditional → KILLED
44. negated integer subtraction with addition → KILLED
45. Replaced integer subtraction with addition → KILLED
46. negated return value with "" for Esame/MaskAndRepeatUtils::maskAndFillToLength → KILLED
47. replaced return value with "" for Esame/MaskAndRepeatUtils::maskAndFillToLength → KILLED
```