Trabalho 2 – Grupo 12: Alexandre Monteiro - 51023 / João Afonso - 51111:

Escolha das funções, propósito e justificação:

Tendo em consideração que o objetivo do trabalho consistia na realização de black-box testing, optou-se por escolher funções que estavam associadas com alguma fonte de conhecimento para poder testar o software sem conhecer a implementação, design e estrutura interna do mesmo. Tirou-se partido da documentação para escolher as funções e realizar os testes.

Foram escolhidas as seguintes funções:

1. Classe – Strings.java / Método – insertPaddedIfNeeded (String s, int insertAt, String stringToInsert):
    - **Propósito:** Esta função é usada para inserir uma palavra numa dada posição de uma string, mas tal só acontece se ainda não existir nenhuma palavra igual na string onde vai ser feita a inserção (a palavra inserida vai estar sempre entre espaços, só se adicionando um espaço antes e/ou depois da palavra se não existirem espaços a separar a string inserida do texto que já estava escrito)
    - **Justificação:** Os parâmetros desta função permitem escolher combinações de valores que podem ser possivelmente problemáticos (por exemplo, o que acontece quando se tenta adicionar uma palavra que já existe na string onde é feita a inserção), sendo interessante realizar testes para averiguar se surgem (ou não) falhas
2. Classe –Strings.java / Método – insertPadded (String s, int insertAt, String stringToInsert):
    - **Propósito:** Esta função é usada para inserir uma palavra (string) numa dada posição de uma string, não importando se a palavra que se quer adicionar já existe (a palavra inserida vai estar sempre entre espaços, só se adicionando um espaço antes e/ou depois da palavra se não existirem espaços a separar a string inserida do texto que já estava escrito)
    - **Justificação:** Esta função aparenta ser semelhante à função referida no ponto 1, mas o seu comportamento difere quando se tenta adicionar uma palavra que já existe na string onde é feita a inserção (neste caso, não importa se a já existe uma palavra igual na string destino) e pode ser interessante averiguar qual vai ser o comportamento nas mesmas circunstâncias da função anterior (isto implica também testar a função nos casos em que se insere na string uma palavra que já existe)
3. Classe – CursorPositionCalculator.java / Método – calculate(int priorCursorPosition, String priorValue, String newValue):
    - **Propósito:** Esta função é usada para calcular a nova posição de um cursor quando uma string é alterada (de acordo com a posição original do cursor na string antes das mudanças)
    - **Justificação:** Tal como nos casos anteriores, existem vários casos de teste interessantes associados com situações que podem ser possivelmente problemáticas (por exemplo, averiguar se o software realmente tem o comportamento descrito na documentação quando priorValue ou newValue têm valor null)

4. Classe – Strings.java / Método – isBlank (String s):
    - **Propósito:** Esta função é usada para averiguar se uma string tem valor null / está vazia / só tem caracteres em branco
    - **Justificação:** Uma função que parece ser simples, podendo ser utilizada em vários pontos do código (é uma função relativamente genérica) e cujo funcionamento incorreto pode ter um largo impacto no programa, sendo apropriado averiguar se o seu comportamento é adequado;
5. Classe – Strings.java / Método – isEmptyOrNull (String s):
    - **Propósito:** Esta função é usada para averiguar se uma string está vazia ou se tem valor null
    - **Justificação:** Função semelhante à função referida no ponto 4, mas que não retorna true se a string estiver em branco, sendo interessante averiguar se o seu comportamento difere como seria esperado.

Category-Partition:

1. Classe – Strings.java / Método – insertPaddedIfNeeded(String s, int insertAt, String stringToInsert):
    1.1. Tem-se 3 parâmetros:
        1.1.1. s – A string na qual se vai inserir a outra string
        1.1.2. insertAt – A posição na qual se vai inserir a string
        1.1.3. stringToInsert – A string que se vai inserir
    1.2. Para cada parâmetro define-se as seguintes características:
        1.2.1. s – Uma string que pode, ou não, conter uma palavra igual à string a inserir
        1.2.2. insertAt – A posição pode corresponder a um número maior ou igual a zero
        1.2.3. stringToInsert – Uma string que pode, ou não, já corresponder a uma palavra existente na string onde se vai fazer a inserção
    1.3. Restrições:
        - O índice não pode ser negativo nem superior ao tamanho da string onde quer fazer a inserção;
        - A string onde vai ser feita a inserção não pode ser vazia.
    1.4. Pode-se combinar as características referidas para obter os seguintes testes:
        - Índice positivo e string a adicionar não corresponde a uma palavra existente na string onde vai ser feita a inserção
        - Índice positivo e string a adicionar corresponde a uma palavra existente na string onde vai ser feita a inserção
        - Índice zero e string a adicionar não corresponde a uma palavra existente na string onde vai ser feita a inserção
        - Índice zero e string a adicionar corresponde a uma palavra existente na string onde vai ser feita a inserção
        - Índice negativo e não importa se a string a adicionar já existe na string se quer fazer a inserção
        - Índice positivo superior ao tamanho da string onde quer fazer a inserção e a string a adicionar não importa
        - String onde é feita a inserção vazia e não importa a string a inserir nem o índice

2. Classe – Strings.java / Método – insertPadded (String s, int insertAt, String stringToInsert):
   2.1. Tem-se 3 parâmetros:
      2.1.1. s – A string na qual se vai inserir a outra string
      2.1.2. insertAt – A posição na qual se vai inserir a string
      2.1.3. stringToInsert – A string que se vai inserir
   2.2. Para cada parâmetro define-se as seguintes características:
      2.2.1. s – Uma string que pode, ou não, conter uma palavra igual à string a inserir
      2.2.2. insertAt – A posição pode corresponder a um número maior ou igual a zero
      2.2.3. stringToInsert – Uma string que pode, ou não, já corresponder a uma palavra existente na string onde se vai fazer a inserção
   2.3. Restrições:
      - O índice não pode ser negativo nem superior ao tamanho da string onde quer fazer a inserção;
      - A string onde vai ser feita a inserção não pode ser vazia.
   2.4. Pode-se combinar as características referidas para obter os seguintes testes:
      - Índice positivo e string a adicionar não corresponde a uma palavra existente na string onde vai ser feita a inserção
      - Índice positivo e string a adicionar corresponde a uma palavra existente na string onde vai ser feita a inserção
      - Índice zero e string a adicionar não corresponde a uma palavra existente na string onde vai ser feita a inserção
      - Índice zero e string a adicionar corresponde a uma palavra existente na string onde vai ser feita a inserção
      - Índice negativo e não importa se a string a adicionar já existe na string se quer fazer a inserção
      - Índice positivo superior ao tamanho da string onde quer fazer a inserção e a string a adicionar não importa
      - String onde é feita a inserção vazia e não importa a string a inserir nem o índice

3. Classe – CursorPositionCalculator.java / Método – calculate(int priorCursorPosition, String priorValue, String newValue):
   3.1. Tem-se 3 parâmetros:
      3.1.1. priorCursorPosition – Posição do cursor antes da mudança
      3.1.2. priorValue – Valor da string antes da mudança
      3.1.3. newValue – Novo valor da string
   3.2. Para cada parâmetro define-se as seguintes características:
      3.2.1.priorCursorPosition – A posição pode corresponder a um inteiro (pode ser um número positivo, negativo ou igual a zero)
      3.2.2. priorValue – Pode ser uma string (incluído string vazia) ou pode ter valor null
      3.2.3. newValue – Pode ser uma string (incluído string vazia) ou pode ter valor null
   3.3. Restrições:
      - Não existem restrições
   3.4. Pode-se combinar as características referidas para obter os seguintes testes:
      - Valor de priorCursorPosition positivo e priorValue e newValue correspondem a strings

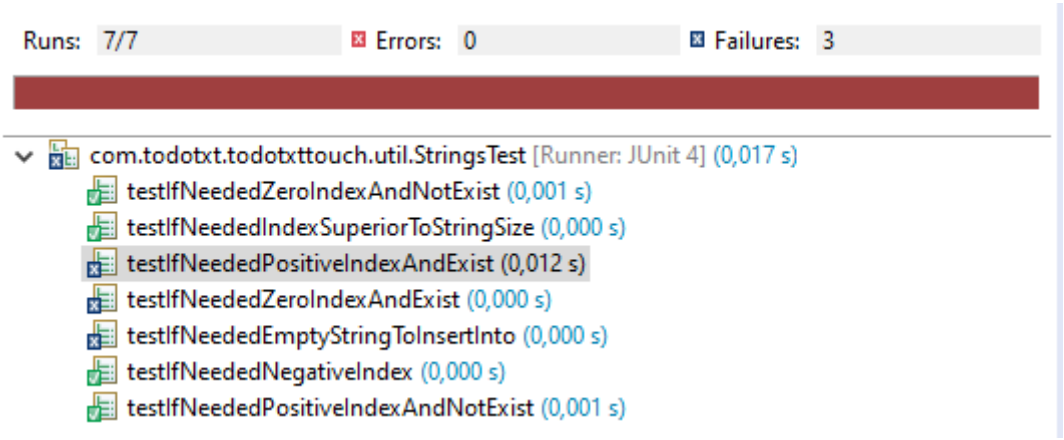
- Valor de priorCursorPosition negativo e priorValue e newValue correspondem a strings
- Valor de priorCursorPosition igual a zero e priorValue e newValue correspondem a strings
- O valor priorValue corresponde é null e newValue corresponde a uma string
- O valor newValue corresponde é null e priorValue corresponde a uma string
- Tem-se que newValue e priorValue correspondem a null
- O valor de priorValue corresponde a uma string vazia
- O valor de newValue corresponde a uma string vazia

4. Classe – Strings.java / Método – isBlank (String s):
   4.1. Tem-se 1 parâmetro:
      4.1.1. s – String que se vai analisar
   4.2. Para cada parâmetro define-se as seguintes características:
      4.2.1. s – A string pode corresponder a null, string vazia, string em branco ou a uma palavra normal
   4.3. Restrições:
      - Não existem restrições
   4.4. Pode-se combinar as características referidas para obter os seguintes testes:
      - A string s é null
      - A string s está vazia
      - A string s corresponde a uma sequência de caracteres em branco
      - A string s é uma string normal

5. Classe – Strings.java / Método – isEmptyOrNull (String s):
   5.1. Tem-se 1 parâmetro:
      5.1.1. s – String que se vai analisar
   5.2. O parâmetro tem as seguintes características:
      5.2.1. s – A string pode corresponder a null, string vazia ou a uma palavra normal
   5.3. Restrições:
      - Não existem restrições para date
   5.4. Pode-se combinar as características referidas para obter os seguintes testes:
      - A string s é null
      - A string s está vazia
      - A string s é uma string normal

Testes e resultados:

1. Classe – Strings.java / Método – insertPaddedIfNeeded(String s, int insertAt, String stringToInsert):

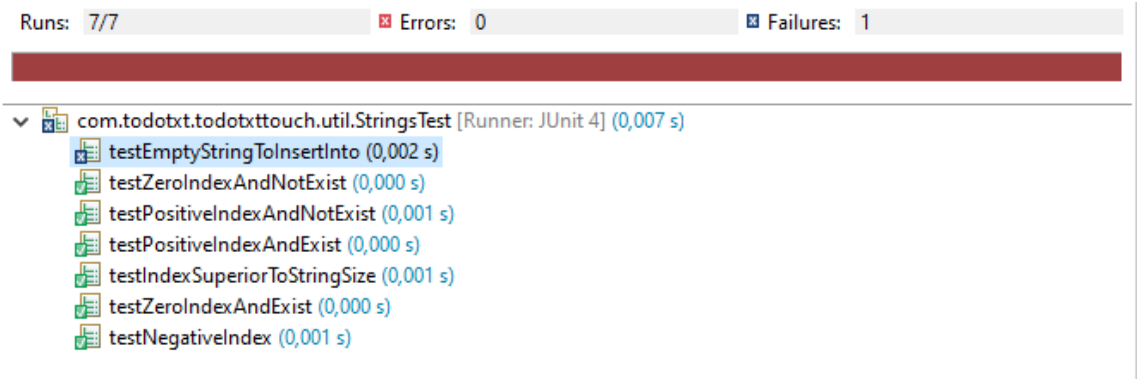


Descrição dos testes e resultado:

- testIfNeededPositiveIndexAndNotExist:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 3 (valor positivo)
    - O resultado esperado (expected) corresponde a "aaa ccc bbb" (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededPositiveIndexAndExist:
    - Criaram-se as strings s = " aaa ccc bbb" e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 3 (valor positivo)
    - O resultado esperado (expected) corresponde a "aaa ccc bbb" (de acordo com a documentação, se a palavra já existe na string onde se quer fazer a inserção, a string original não deve de ser alterada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se um valor indesejado
    - O actual value obtido foi “aaa ccc bbb ”, tendo-se acrescentado um espaço no fim da string e, de acordo com a documentação, a string não devia de ter sido alterado porque a palavra já existia, concluindo-se que isto corresponde a uma falha
- testIfNeededZeroIndexAndNotExist:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a "ccc aaabbb" (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededZeroIndexAndExist:

    - Criaram-se as strings s = "ccc aaabbb " e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a "ccc aaabbb " (de acordo com a documentação, se a palavra já existe na string onde se quer fazer a inserção, a string original não deve de ser alterada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se um valor indesejado
    - O actual value obtido foi “ccc aaabbb ”, tendo-se acrescentado um espaço no fim da string e, de acordo com a documentação, a string não devia de ter sido alterado porque a palavra já existia, concluindo-se que isto corresponde a uma falha
- testIfNeededNegativeIndex:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = -2
    - De acordo com a documentação (insertAt < 0), deve de ocorrer um IndexOutOfBoundsException
    - Chamou-se a função e, tal como esperado, occoreu um IndexOutOfBoundsException
- testIfNeededIndexSuperiorToStringSize:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 10
    - De acordo com a documentação (insertAt maior do que o tamanho de s), deve de ocorrer um IndexOutOfBoundsException
    - Chamou-se a função e, tal como esperado, occoreu um IndexOutOfBoundsException
- testIfNeededEmptyStringToInsertInto:
    - Criaram-se as strings s = "" e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a " ccc "
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se um valor indesejado
    - O actual value obtido foi “ccc “ tendo-se acrescentado um espaço no fim (sem haver uma justificação para tal estar a acontecer) concluindo-se que isto corresponde a uma falha

2. Classe – Strings.java / Método – insertPadded (String s, int insertAt, String stringToInsert):

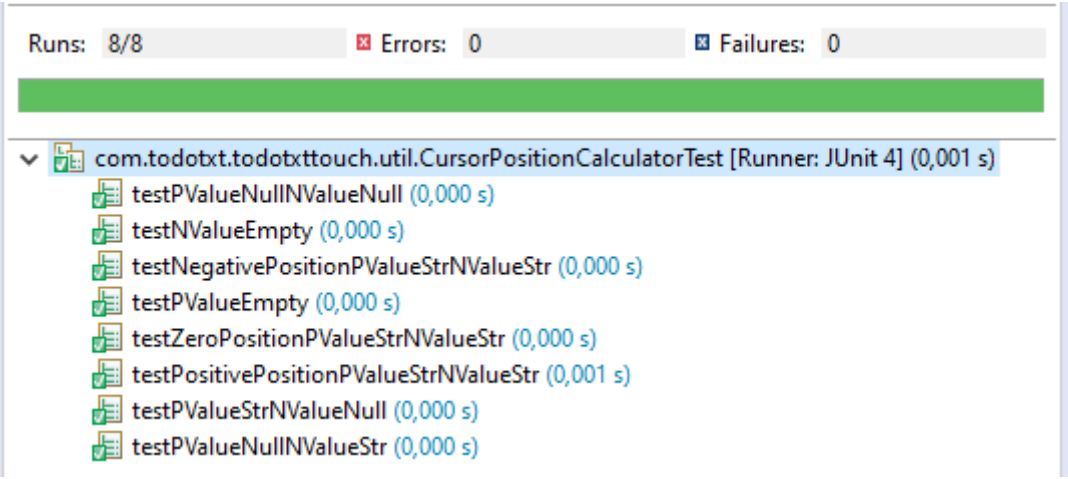


Descrição dos testes e resultado:

- testIfNeededPositiveIndexAndNotExist:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 3 (valor positivo)
    - O resultado esperado (expected) corresponde a "aaa ccc bbb" (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededPositiveIndexAndExist:
    - Criaram-se as strings s = " aaa ccc bbb" e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 3 (valor positivo)
    - O resultado esperado (expected) corresponde a "aaa ccc ccc bbb" (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededZeroIndexAndNotExist:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a "ccc aaabbb" (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededZeroIndexAndExist:
    - Criaram-se as strings s = "ccc aaabbb " e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a "ccc ccc aaabbb " (adiciona-se a palavra na posição indicada)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testIfNeededNegativeIndex:

    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = -2
    - De acordo com a documentação (insertAt < 0), deve de ocorrer um IndexOutOfBoundsException
    - Chamou-se a função e, tal como esperado, occoreu um IndexOutOfBoundsException
- testIfNeededIndexSuperiorToStringSize:
    - Criaram-se as strings s = "aaabbb" e stringToInsert = "ccc" (a palavra a adicionar não fazer parte da string em que se vai fazer a inserção) e um inteiro insertAt = 10
    - De acordo com a documentação (insertAt maior do que o tamanho de s), deve de ocorrer um IndexOutOfBoundsException
    - Chamou-se a função e, tal como esperado, occoreu um IndexOutOfBoundsException
- testIfNeededEmptyStringToInsertInto:
    - Criaram-se as strings s = "" e stringToInsert = "ccc" (a palavra a adicionar já faz parte da string em que se vai fazer a inserção) e um inteiro insertAt = 0
    - O resultado esperado (expected) corresponde a "ccc"
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se um valor indesejado
    - O actual value obtido foi “ccc “ tendo-se acrescentado um espaço no fim (sem haver uma justificação para tal estar a acontecer) concluindo-se que isto corresponde a uma falha

3. Classe – CursorPositionCalculator.java / Método – calculate(int priorCursorPosition, String priorValue, String newValue):



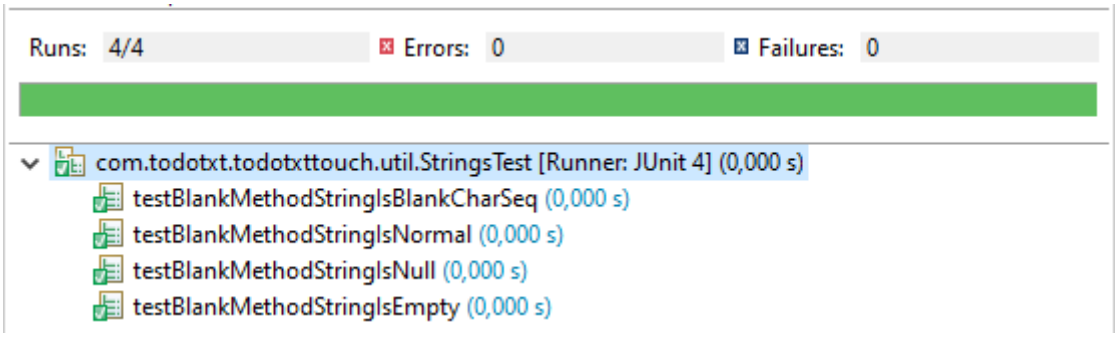
Descrição dos testes e resultado:

- testPositivePositionPValueStrNValueStr:
    - Criaram-se as strings priorValue = "qwe" e newValue = "qwerty" (strings com tamanho maior que zero) e um inteiro priorCursorPosition = 1 (valor positivo)
    - O resultado esperado (expected) corresponde a 4

    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testNegativePositionPValueStrNValueStr:
    - Criaram-se as strings priorValue = "qwe" e newValue = "qwerty" (strings com tamanho maior que zero) e um inteiro priorCursorPosition= -1 (valor negativo)
    - O resultado esperado (expected) corresponde a 0 porque a função não deve de poder retornar o cursor a pontar para uma posição inferior a 0
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testZeroPositionPValueStrNValueStr:
    - Criaram-se as strings priorValue = "qwe" e newValue = "qwerty" (strings com tamanho maior que zero) e um inteiro priorCursorPosition = 0
    - O resultado esperado (expected) corresponde a 3 (o facto de a posição inicial do cursor ser zero não deve de influenciar a forma como é calculada a alteração da sua posição)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testPValueStrNValueNull:
    - Criaram-se as strings priorValue = "qwe" e newValue = null e um inteiro priorCursorPosition = 1 (neste caso, este valor não importa)
    - O resultado esperado (expected) corresponde a 0 (de acordo com a documentação, se newValue tiver valor null, a posição deve ser 0)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testPValueNullNValueStr:
    - Criaram-se as strings priorValue = null e newValue = "qwerty" e um inteiro priorCursorPosition = 1 (neste caso, este valor não importa)
    - O resultado esperado (expected) corresponde a 6 (de acordo com a documentação, se priorValue tiver valor null, a posição calculada deve corresponder à posição logo a seguir à string)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testPValueNullNValueNull:
    - Criaram-se as strings priorValue = null e newValue = null e um inteiro priorCursorPosition = 0 (neste caso, este valor não importa)
    - O resultado esperado (expected) corresponde a 0 (devia de ser equivalente ao que acontecia no caso de apenas newValue estar a null)
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testPValueEmpty:
    - Criaram-se as strings priorValue = "" e newValue = "qwerty" e um inteiro priorCursorPosition = 0
    - O resultado esperado (expected) corresponde a 6
    - Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado
- testNValueEmpty:

- Criaram-se as strings priorValue = "qwerty" e newValue = "" e um inteiro priorCursorPosition = 1
- O resultado esperado (expected) corresponde a 0
- Calculou-se o valor verdadeiro (actual) e fez-se assertEquals(expected, actual), obtendo-se o valor desejado

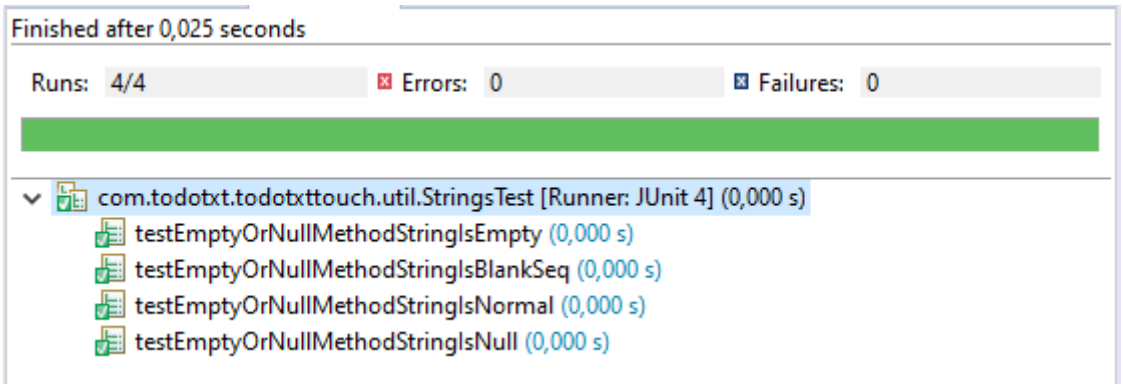
4. Classe – Strings.java / Método – isBlank (String s):



Descrição dos testes e resultado:

- testBlankMethodStringIsNull:
  - Se a string s tiver valor null – resultado deve ser true
  - Criou-se uma string com valor null e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testBlankMethodStringIsEmpty:
  - Se a string s for vazia – resultado deve ser true
  - Criou-se uma string vazia e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testBlankMethodStringIsBlankCharSeq:
  - Se a string s só tiver caracteres em branco – resultado deve ser true
  - Criou-se uma string com valor " " (sequência de caracteres brancos) e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testBlankMethodStringIsNormal:
  - Se a string s não corresponder a nenhum dos casos anteriores – resultado deve ser false
  - Criou-se uma string com valor "qwerty"e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado

5. Classe – Strings.java / Método – isEmptyOrNull (String s):



Descrição dos testes e resultado:

- testEmptyOrNullMethodStringIsNull:
  - Se a string s tiver valor null – resultado deve ser true
  - Criou-se uma string com valor null e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testEmptyOrNullMethodStringIsEmpty:
  - Se a string s for vazia – resultado deve ser true
  - Criou-se uma string vazia e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testEmptyOrNullMethodStringIsBlankSeq:
  - Se a string s só tiver caracteres em branco – resultado deve ser false
  - Criou-se uma string com valor " " (sequência de caracteres brancos) e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado
- testEmptyOrNullMethodStringIsNormal:
  - Se a string s não corresponder a nenhum dos casos anteriores – resultado deve ser false
  - Criou-se uma string com valor "qwerty"e fez-se assertTrue(Strings.isBlank(s)), tendo-se obtido o resultado esperado