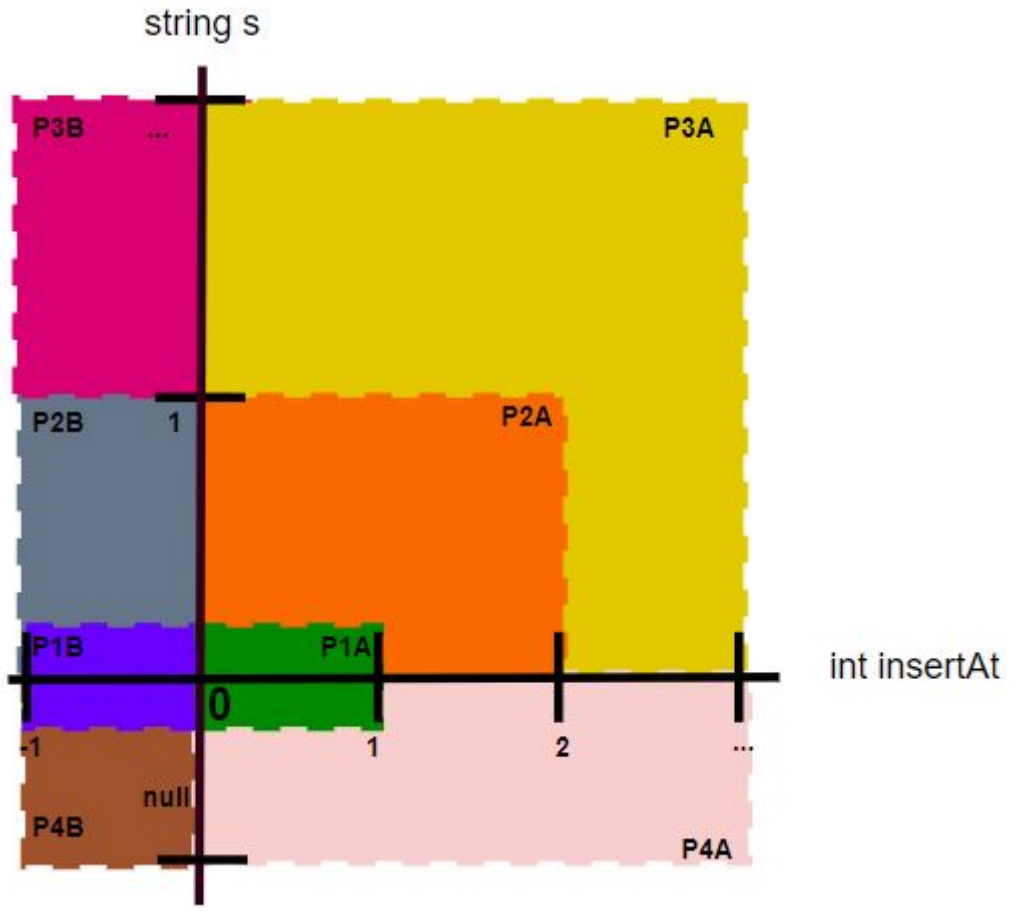
Pacote: com.todotxt.todotxttouch.util
Classe: Strings
Método: insertPaddedIfNeeded
Parâmetros: (String s, int insertAt, String stringToInsert)
Descrição:
O método *insertPaddedIfNeeded* têm como objetivo inserir uma *String* dentro de outra, numa determinada posição, e devolver o resultado da concatenação das mesmas. O último parâmetro não pode estar presente na *String s*.
Tendo em conta que os 3 parâmetros da função são interdependentes, em que o índice depende do tamanho da primeira *String* e o último parâmetro não pode estar presente no primeiro, e devido à combinação de testes necessários para apurar todos os inputs possíveis (que alcançaria cerca de 70 testes) decidimos apenas a analisar as combinações possíveis entre a *String s* e *int insertAt,* sendo o último parâmetro igual a "xdfcgc".
Também, o terceiro parâmetro (*String stringToInser*t) podia ser adicionado ao gráfico abaixo, adicionando um novo eixo, correspondente a este input. Contudo, adicionar uma nova dimensão ao gráfico e representar as partições dessa forma só tornaria o gráfico confuso e pouco perceptível, o que sustentou ainda mais a nossa decisão de não contemplar este parâmetro para avaliar a validade do método.
Partições:



Explicação:
Tendo em conta os parâmetros do método, decidimos criar partições da inter-dependência entre a *String s* e o *int insertAt*, visto que o índice dado como input é verificado com base na primeira *String* dada.

Partições:

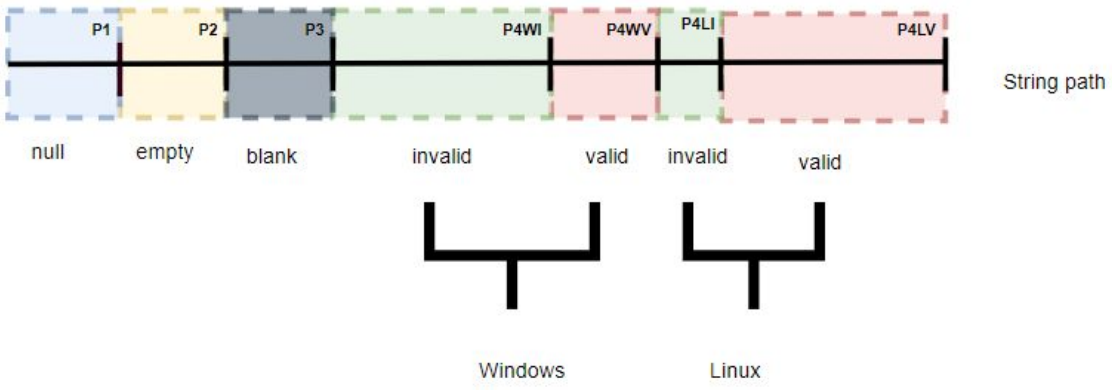
- **Variância dos parâmetros:**
  - O *int insertAt* vai variar entre 0, o tamanho da *String s* e o tamanho da *String s* + 1, de forma a ver se o sistema se comporta para valores extremos (insertAt = 0 e insertAt = s.length) e um valor que já se encontra fora do intervalo de valores aceite (insertAt = s.length+1).
- **Características dos parâmetros:**
  - String s:
    - Sendo uma String, pode tomar valores:
      - null;
      - (0) String vazia
      - (1) Ter um elemento;
      - (...) Ter vários elementos;
  - Int insertAt:
    - Sendo um Integer, está compreendido entre:
      - (-1) Valores negativos em que apenas -1 interessa para a fronteira da partição;
      - (0) Ter valor zero;
      - (1) Ter valor um;
      - (2) Ter valor dois;
      - (...) Ter valor superior a dois;
- **P1A**
  - Esta partição representa o conjunto de casos em que a *String s* é vazia.
  - A variável insertAt varia entre 0 e 1.
  - Testes:
    - Para os testes realizados, o sistema teve o comportamento esperado.
- **P2A**
  - Esta partição representa o conjunto de casos em que a *String s* tem um caractere.
  - A variável insertAt varia entre 0, 1 e 2.
  - Testes:
    - Para os testes realizados, o sistema teve o comportamento esperado.
- **P3A**
  - Esta partição representa o conjunto de casos em que a *String s* mais que um caractere. Para efeitos demonstrativos, s = “aaa”
  - A variável insertAt varia entre 0, 3 e 4.
  - Testes:
    - Para os testes realizados, o sistema teve o comportamento esperado.
- **P4A**
  - Esta partição representa o conjunto de casos em que a *String s* é nula. Tendo em conta que não há documentação quanto a esta restrição, achámos importante ver se o programa suporta este input.
  - A variável insertAt varia entre 0 e 1.
  - Testes:
    - Os dois testes que realizámos, para os dois valores possíveis de *insertAt*, **chumbaram**. A partir da informação retirada do JUnit, podemos concluir que o programa tenta obter o tamanho da *String s* e, como neste caso, ela é null, o programa lança um NullPointerException.
- **P1B**
  - Esta partição representa o conjunto de casos em que a *String s* é vazia.
  - A variável insertAt é -1.
  - Teste:
    - Para o teste realizado, o sistema teve o comportamento esperado.

- **P2B**
    - Esta partição representa o conjunto de casos em que a *String s* tem um caractere.
    - A variável insertAt é -1.
    - Teste:
        - Para o teste realizado, o sistema teve o comportamento esperado.
- **P3B**
    - Esta partição representa o conjunto de casos em que a *String s* mais que um caractere. Para efeitos demonstrativos, s = “aaa”
    - A variável insertAt é -1.
    - Teste:
        - Para o teste realizado, o sistema teve o comportamento esperado.
- **P4B**
    - Esta partição representa o conjunto de casos em que a *String s* é nula.
    - A variável insertAt é -1.
    - Testes:
        - O teste realizado **chumbou**. É o mesmo problema que o teste da Partição “paralela” que têm o mesmo input no primeiro parâmetro. Tena aceder à variável e, como é null, o programa crasha, afetando gravemente a disponibilidade do sistema e, por isso, ser um problema a ser tratado pelos developers.

Pacote: com.todotxt.todotxttouch.util
Classe: Path
Método: fileName
Parâmetros: (String path)
Descrição:
O método fileName não contém documentação, por isso analisámos a implementação e deu para perceber que este método tem como objetivo filtrar o caminho para um ficheiro, de modo a retornar só o nome do mesmo.
Escolhemos esta função pois recebe um parâmetro String que será filtrado e que será usado para criar o nome de um ficheiro. Assim, devido às restrições impostas pelos sistemas operativos a nomes de ficheiros, decidimos testar a validade deste método. Também as restrições inerentes ao input, não só limita as partições ao número de caracteres, assim como aos caracteres usados.
Para averiguar a validade de alguns caracteres, testámos 2 sistemas operativos: Windows e Linux. Cada um destes têm as suas restrições quanto a nomes de ficheiros e directorias, por isso testámos uma amostra de inputs para cada, de modo a validar se o programa suporta o "naming" de ficheiros corretamente, de acordo com o estipulado pelos SO's. Em abaixo estão os links nos quais nos baseámos para verificar que inputs podem ser aceites, para cada sistema operativo:

- Naming Conventions for Windows
- Naming Conventions for Linux

Partições:



Explicação:
Tendo em conta o parâmetro *String path* e as restrições inerentes ao nome de um ficheiro, foram feitas várias partições de possíveis inputs.
Partições:

- **Características dos parâmetros:**
  - String path:
    - Como String que irá ser utilizada para nomear um ficheiro, é importante verificar os inputs:
      - null;
      - (empty) String vazia;
      - (blank) String composta só por espaços;
      - (invalid) String com caracteres inválidos para o Windows;
      - (valid) String com caracteres válidos para o Windows;
      - (invalid) String com caracteres inválidos para o Linux;
      - (valid) String com caracteres válidos para o Linux;
- **P1**
  - Nesta partição, a variável toma valor null.

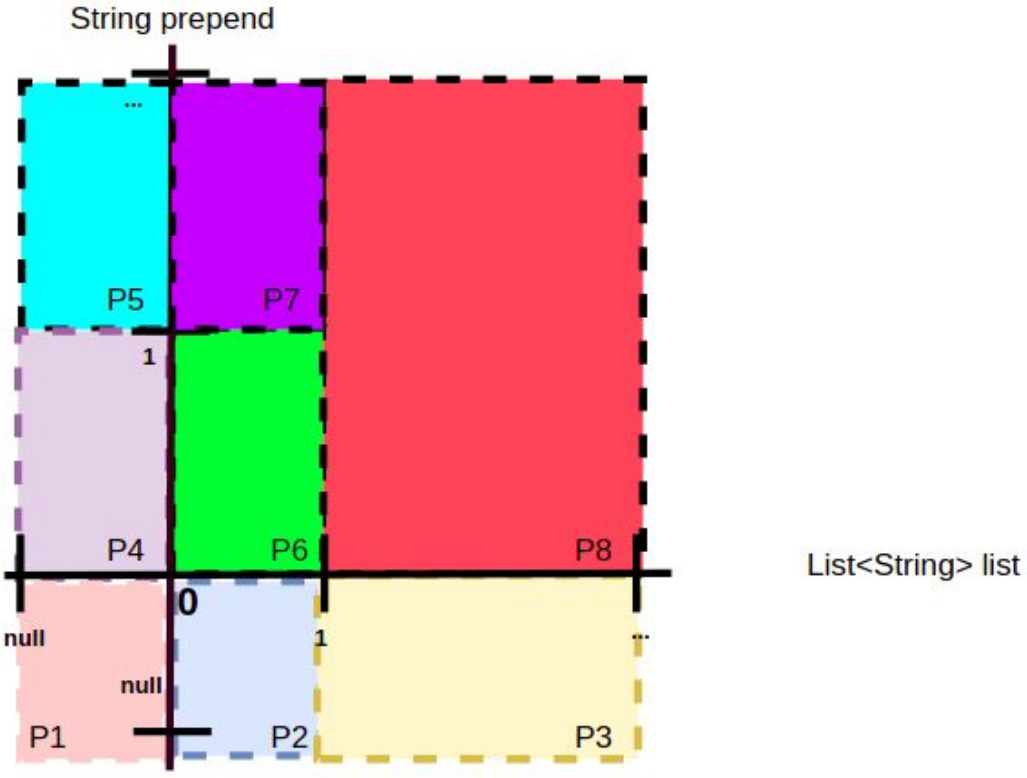
    - Testes:
        - No teste realizado para este input a *String path* tinha valor *null*. O sistema devolve uma *String* vazia.
- **P2**
    - Nesta partição, a *String path* é empty.
    - Testes:
        - O sistema devolveu uma *String* vazia.
        - Ora, não é possível criar um ficheiro com nome vazio e, por isso, deveria ser lançada uma excepção para os casos em que retorne uma *String* vazia, como no caso anterior.
- **P3**
    - Nesta partição, a *String path*.é um conjunto de espaços.
    - Decidimos verificar este caso porque não é possível criar ficheiros e directorias com nome composto só de espaços.
    - Testes:
        - O sistema devolveu uma *String* vazia.
        - Assim, como nos testes anteriores, tirámos as mesmas conclusões. Em casos que o input não seja aceite pelo programa, não deveria ser retornada uma *String* vazia, visto que não é um nome válido para ficheiros.
- **P4W**
    - Esta partição corresponde ao conjunto de casos especialmente desenhados para o sistema operativo Windows. Para isso foram criada duas sub-partições:
        - Uma para os casos em que o input é inválido só para este sistema operativo;
        - Outra para os casos em que o input é válido só para este sistema operativo
    - Decidimos verificar os casos para este sistema operativo visto que um programa deve ser portável (Capaz de correr em diversos ambientes, como por exemplo Linux e Windows) e o sistema de ficheiros difere nos vários sistemas operativos.
    - **Sub-Partição P4WI (P4WI = Partição 4 Windows Inválido)**
        - Corresponde ao conjunto de inputs inválidos para o sistema Windows.
        - Testes:
            - De acordo com o site a que recorremos para obter informações acerca das restrições para o nome de ficheiros, caracteres como “<”, “>”, “:”, “?” são inválidos. Por isso, quando estes caracteres estão presentes no input, seria expectável que o programa reconhecesse qual o ambiente de execução em que se encontra e verificar os caracteres. Contudo, o output ignora estes caracteres e devolve o último nome dado do *path* para ser o nome do ficheiro. Claramente, o processamento de inputs com estes caracteres resultarão numa falha no programa que não está preparado para lidar com nomes inválidos para ficheiros, causando erros. O programa **chumbou** neste teste.
            - Também testámos para o nome “Nul” que não é aceite pela sistema de ficheiros, como é exemplarmente explicado: “Now, it’s over forty years later and we still can’t name files “con.txt” or “aux.mp3” because Windows wants to stay compatible with ancient programs that might be using this feature. It’s a good example of how intensely Microsoft is committed to backwards compatibility” em https://www.howtogeek.com/fyi/windows-10-still-wont-let-you-use-these-file-names-reserved-in-1974/. Visto que em Windows, não é permitido criar ficheiros com nome “Nul”, deveria ser lançada uma excepção e não retornar a própria *String*. Logo, o programa **chumbou** neste teste.

- **Sub-Partição P4WV (P4WV = Partição 4 Windows válido)**
  - Corresponde ao conjunto de inputs válidos para o sistema Windows.
  - Testes:
    - Para estes testes, utilizámos inputs de uma só palavra, assim como um caminho com "/" para delimitar a pasta do nome final do ficheiro.
    - Também, como valor extremo da partição, testámos com "/" no final do *path*, resultando na *String* corretamente delimitada.
    - O sistema teve o comportamento esperado.

- **P4L**
  - Esta partição corresponde ao conjunto de casos especialmente desenhados para o sistema operativo Linux. Para isso foram criada duas sub-partições:
    - Uma para os casos em que o input é inválido só para este sistema operativo;
    - Outra para os casos em que o input é válido só para este sistema operativo
  - Decidimos verificar os casos para este sistema operativo visto que um programa deve ser portável (Capaz de correr em diversos ambientes, como por exemplo Linux e Windows) e o sistema de ficheiros difere nos vários sistemas operativos.
  - **Sub-Partição P4WI (P4LI = Partição 4 Linux Inválido)**
    - Corresponde ao conjunto de inputs inválidos para o sistema Windows.
    - Testes:
      - De acordo com o site a que recorremos para obter informações acerca das restrições para o nome de ficheiros, só o caracter "/" é inválido. Por isso, quando este caractere está presente no input, seria expectável que o programa reconhecesse qual o ambiente de execução em que se encontra e verificar o caractere. Curiosamente, tendo em conta que o programa, considera o nome do ficheiro qualquer String após o último "/", em Linux, o programa evita problemas na directoria de ficheiros, porque não têm em consideração o sistema operativo em que corre e por isso, retorna sempre a última *String*.
  - **Sub-Partição P4LI (P4LI = Partição 4 Linux válido)**
    - Corresponde ao conjunto de inputs válidos para o sistema Linux.
    - Testes:
      - Para estes testes, utilizámos o conjunto de caracteres(<,>,:,?, etc...) não suportado pelo Windows e, como esperado, o programa devolveu a String com os mesmos. Como é em ambiente Linux e estes caracteres são permitidos, o programa teve o comportamento expectável.
      - Fazendo a análise do programa em Windows (apesar de testarmos o comportamento do sistema para caracteres em Linux) quisemos testar se o programa estava bem implementado para um *path* "normal" em Linux, que contém "\". Contudo, o programa devolveu a *String* toda. Sem analisar o software, pode-se concluir que o programa foi desenvolvido para Windows e não Linux, visto que o único File Separator que tem em conta é "/", partindo dos resultados obtidos na partição P4WV. **Chumbando** assim neste teste.
      - Uma possível solução para adaptar o programa ao sistema operativo, seria usar o File Separator oferecido pelo Java, que tem em conta o sistema operativo e não "hard-coded" o separador.

Pacote: com.todotxt.todotxttouch.util
Classe: Util
Método: PrependString
Parâmetros: (ArrayList<String> list,String prepend)
Descrição:
O método tem como objetivo inserir a String prepend como prefixo de todas as strings da lista fornecida.

Partições:



Tendo em conta os parâmetros do método, decidimos criar partições da inter-dependência entre a *String prepend* e a lista *list* , visto que a string é inserida em todos os valores dentro da lista, dependendo do seu tamanho.

Variância dos parâmetros:

- prepend:
  - A String pode variar entre null e tamanho de 1 caractere ou superior.
- list:
  - Esta lista pode variar entre null e tamanho de 1 elemento ou superior.

Características dos parâmetros:

- String prepend:
  - Sendo uma String, pode tomar valores:
    - null;
    - (0) String vazia
    - (1) Ter um elemento;
    - (...) Ter vários elementos;
- List lista:
  - Sendo uma Lista, pode tomar valores:
    - null
    - (0) Lista vazia
    - (1) Ter um elemento
    - (...) Ter vários elementos

Partições:

- **P1:**
  - Nesta partição ambas as variáveis tomam o valor null. Espera-se receber uma exceção (NullPointerException), que se confirmou.
- **P2:**
  - Nesta partição a string prepend toma o valor null. A Lista toma o valor de uma lista vazia. Espera-se receber uma lista vazia, que se confirmou.
- **P3:**
  - Nesta partição a String prepend toma o valor null. A lista toma o valor de uma lista com um ou mais elementos. A palavra “null” será adicionada como prefixo a todas as palavras da Lista. Algo que se confirmou.
- **P4:**
  - Nesta partição, a String prepend toma o valor de uma String vazia. A lista toma o valor de null. Espera-se receber uma exceção (NullPointerException), que se confirmou.
- **P5:**
  - Nesta partição, a String prepend toma o valor de uma String com um ou mais caracteres. A lista toma o valor de null. Espera-se receber uma exceção (NullPointerException), que se confirmou.
- **P6:**
  - Nesta partição ambas as variáveis tomam o valor 0, lista vazia e String vazia. Espera-se receber uma lista vazia, algo que se confirmou.
- **P7:**
  - Nesta partição a String prepend toma o valor de uma String com um ou mais caracteres. A Lista toma o valor de uma lista vazia. Espera-se receber uma lista vazia, algo que se confirmou.
- **P8:**
  - Nesta partição a String prepend toma o valor de uma String com um ou mais caracteres, e a Lista toma o valor de uma lista com um ou mais caracteres. Espera-se que a String prepend seja inserida como prefixo de todas as Strings da lista, algo que se confirmou.