Tests de caja negra

Se han implementado un total de 11 tests (3 de ellos happy paths). A continuación, se describen en profundidad:

- **testForEmptyGraph()**:
  - Clases cubiertas: 'Graph.java' e 'In.java'.
  - Se basa en la premisa de que al invocar al constructor de Graph, se intentará leer de un filename (en nuestro caso). Si este fichero es inválido (null en este caso), en la clase 'In.java' se elevará la excepción IllegalArgumentException. Comprobamos así si salta esa excepción. La única caracterización sería filename válido o no (true o false).
- **testForNoDelimiter()**:
  - Es muy parecido al anterior, pero teniendo en cuenta ahora el argumento *delimiter* (fijado a null en este ejemplo). De nuevo, la única caracterización sería *delimiter* válido o no (true o false).
- **testForNoFile()**:
  - Clases cubiertas: 'Graph.java' e 'In.java', nuevamente.
  - Al llamar al constructor de Graph, se intentará leer de un **filename** (en nuestro caso). Si este fichero es inválido (no existe), en la clase 'In.java' se elevará la excepción IllegalArgumentException. Comprobamos así si salta esa excepción. La única caracterización sería filename existente o no (true o false).
- **testForIncompatibleTypes()**:
  - Clase cubierta: 'PathFinder.java'.
  - Comprobamos si salta la excepción oportuna cuando se llama al constructor de Pathfinder(graph,name), al intentar añadir en su atributo **dist** (del tipo ST) el string "" (name). Queda explicado en el propio 'GraphFuncionalityTest.java'. La única caracterización sería **name** válido o no (true o false).
- **testForSizeZeroGraph()**:
  - Clase cubierta: 'GraphFuncionality.java'.
  - Métodos cubiertos: `doDistance`, `doGraphFilter` o `doRanking`, ya que todos tienen como entrada un argumento de tipo Graph.
  - Se basa en la premisa de que un grafo con 0 vértices no es válido (posible **filename** erróneo). Utilizaremos un fichero vacío como argumento para el constructor de Graph, y un delimitador correcto. El grafo creado tendrá 0 vértices, y se elevará la excepción. La única caracterización sería **graph** válido o no (true o false).

- **testForCloseConnection()**:
    - Clase cubierta: 'GraphFuncionality.java'.
    - Método cubierto: `nameChecker(Connection conn, String name)`.
    - Tratamos el caso de que si no introducimos un Connection correcto (porque está cerrado), se eleve la excepción correspondiente. La única caracterización sería **conn** válido o no (cerrado o abierto, true o false).
- **testForInvalidName()**:
    - Clase cubierta: 'GraphFuncionality.java'.
    - Método cubierto: `nameChecker(Connection conn, String name)`.
    - Test para comprobar que salta la excepción si por lo que sea a namechecker le llega un name="" (inválido). Es muy poco probable, ya que en `doDistance` y `doGraphFilter` comprobamos si sus strings de entrada no son "". Aún así, por si "la liamos" y cambiamos **name**, estamos cubiertos en este caso. La única caracterización es **name** válido o no (true o false).
- **testForInvalidName2()**:
    - Clase cubierta: 'GraphFuncionality.java'.
    - Método cubierto: `doRanking(Graph g, String number)`.
    - Análogo al anterior, pero para otro método. Comprueba que salta la excepción si por lo que sea al método le llega un number="" (inválido). La única caracterización es **number** válido o no (true o false).

Y ahora, los 3 happy paths:

- **validGraph()**: test para comprobar que se crea un grafo correctamente si filename y delimiter son correctos.
- **validPathFinder()**: test para comprobar que se crea un pathfinder correcto (con ruta) dado dos nombres relacionados.
- **validNameChecker()**: test para comprobar si namechecker retorna un ArrayList válido dado un nombre valido.

Se han dejado comentados otros tests. No se han incluído porque hacen referencia a los distintos `IllegalArgumentException` lanzados por los métodos `doDistance` y `doGraphFilter`, ya que so tratados con catch (son Checked Exceptions).

Tests de caja blanca

Dados los grafos de cada método, se expondrán las rutas seguidas por cada test (que cubra la clase GraphFuncionality):

- **testForSizeZeroGraph()**: utilizamos `doDistance` como método de ejemplo (misma ruta en `doGraphFilter`). Ruta: [1,2].
- **testForCloseConnection()**: método `nameChecker`. Ruta: [1,3,10].
- **testForInvalidName()**: método `nameChecker`. Ruta: [1,2].
- **testForInvalidName2()**: método `doRanking`. Ruta: [1,2]
- **validGraph()**: Main.java. No tenemos su grafo.
- **validPathFinder()**:
- **validNameChecker()**: