Descripción del juego de pruebas de la clase “distancia.java”:

• **Objeto de la prueba:** Este juego de pruebas engloba la clase distancia. Dicha clase representa el conjunto de distancias de las teclas del teclado y es una extensión de la clase distanciaCluster, por lo que hereda todos sus atributos y métodos. Su uso se basa en tener organizadas todas las distancias que hay entre las teclas del teclado, para después consultarlas o modificarlas.

• **Otros elementos integrados en la prueba:** Se ha creado el archivo de compilación y ejecución “makefile”. Esto es, al escribir “make” en el terminal, se compilará y ejecutará la clase con el juego de pruebas.

• **Drivers:** Se ha creado el driver “driverDistancia.java” para probar el correcto funcionamiento de la clase. En él se ha creado la función “main” para probar todas sus funcionalidades. Al ejecutar el mismo, se muestra por pantalla la enumeración de todos los métodos que la clase permite llevar a cabo.

• **Stubs:** No se han creado stubs para esta clase.

• **Fichero de datos necesario:** El archivo “driverDistancia.in” contiene los datos necesarios para probar la clase desde un fichero. En éste se prueban todas las funcionalidades de la clase para corroborar su correcto funcionamiento. El código contiene excepciones que tienen en cuenta los posibles errores. Éstas informaran de los mismos cuando sea debido.

• **Valores estudiados:** Se han estudiado valores aleatorios para hacer la prueba de esta clase.

1. La primera funcionalidad crea una distancia por defecto, por lo que no se le pasan atributos.
2. La segunda funcionalidad crea una distancia con todos los parámetros necesarios, es decir, su forma, el número de filas, el número de columnas y el número de posiciones.
3. La tercera funcionalidad cambia la forma del teclado, pasándole la nueva forma por parámetro (“String”). Lo mismo sucede con la quinta, séptima, novena, decimoprimera, decimotercera, decimoquinta y decimoséptima funcionalidad, siendo respectivamente para cambiar el numero de filas (“Integer”), el numero de columnas (“Integer”), el numero de posiciones (“Integer”), un elemento del vector de las primeras posiciones de cada fila (“Integer, Integer”), el vector de las primeras posiciones de cada fila (“Integer[]”), un elemento de la matriz de distancias (“Integer, Integer, Integer”) y la matriz de distancias (“Integer[]”).
4. La cuarta funcionalidad consulta la forma del teclado, sin pasarle ningún parámetro explícito. Lo mismo sucede con la sexta, octava, décima, duodécima, decimocuarta, decimosexta y decimoctava funcionalidad, siendo respectivamente para consultar el numero de filas, el numero de columnas, el numero de posiciones, un elemento del vector de primeras posiciones de cada fila (“Integer”), el vector de primeras posiciones de cada fila, un elemento de la matriz de distancias (“Integer, Integer”) y la matriz de distancias.

• **Operativa:** El funcionamiento del ejecutable se lleva a cabo mediante la introducción repetida de números seguidos de los datos con los que se quiere operar. Al ejecutarse, aparecen por pantalla todos los métodos de la clase:

DriverDistancia

Elige una opicion:

1) distancia()");
2) distancia(formaTeclado, numeroFilas, numeroColumnas, numeroPosiciones)");
3) setFormaTeclado(formaTeclado)");
4) getFormaTeclado()");
5) setNumeroFilas(numeroFilas)");
6) getNumeroFilas()");
7) setNumeroColumnas(numeroColumnas)");
8) getNumeroColumnas()");
9) setNumeroPosiciones(numeroPosiciones)");
10) getNumeroPosiciones()");
11) setPrimeraPosicionForma(numeroFila, posicion)");
12) getPrimeraPosicionForma(numeroFila)");
13) setTodasPrimerasPosicionesForma(v[numeroColumnas] = {...})");
14) getTodasPrimerasPosicionesForma()");
15) setDistanciaMatriz(posicionA, posicionB, distancia)");
16) getDistanciaMatriz(posicionA, posicionB)");
17) setTodasDistanciasMatriz(v[numeroPosiciones] = [...])");
18) getTodasDistanciasMatriz()");
0) Salir");