MTR Traversal Coursework UML

Dom Meyer, Jarnail Singh and Matthew Barber

graph

Edge `C,W`

- a: Node<C, W>
- b: Node<C, W>
- weight: W

+ Edge(a: Node<C, W>, b: Node<C, W>, weight: W)
+ getNode(node: Node<C, W>): Node<C, W>
+ getNodes(): Node<C, W>[]
+ getWeight(): W
+ toString(): String
+ toString(node: Node<C, W>): String

C: Content
W: Weight

Node `C,W`

- content: C
- edges: Set<Edge<C, W>>

+ Node()
+ Node(content: C)
+ Node(content: C, edge: Edge<C, W>)
+ Node(content: C, edges: Set<Edge<C, W>>)
+ setContent(content: C): void
+ setEdges(edges: Set<Edge<C, W>>): void
+ getContent() : C
+ getEdges(): Set<Edge<C, W>>
+ addEdge(edge: Edge<C, W>): boolean
+ toString(): String

+ bfs(start: Node<C, W>, end: Node<C, W>): <C, W> List<Edge<C, W>>

<<imports>>

Java In-built Exceptions

<<imports>>

<<imports>>

Java Collections Framework

C as String represents station name
W as String represents the line that connects stations

<<imports>>

<<imports>>

mtr

WorkingController

- lineMap: Map<String, String[]>
- nodesMap: Map<String, Node<String, String>>

+ WorkingController(path: String)
- generateLineMap(path: String): HashMap<String, String[]>
- generateNodesMap(lineMap: Map<String, String[]>): HashMap<String, Node<String, String>>
+ listAllTermini: String
+ listAllStationsInLine(line: String): String
+ listAllDirectlyConnectedLines(line: String): String
+ showPathBetween(stationA: String, stationB: String): String

TUI main method explicitly
initialises WorkingController
as type Controller

TUI

- path: String
- controller: Controller
- stdIn: Scanner

+ TUI(Controller)
- displayMenu(): void
- getAndProcessUserOption(): void
- header(): String
- menu(): String

+ main(args: String[]): void

«interface»
Controller

+ listAllTermini(): String
+ listStationsInLine(line: String): String
+ listAllDirectlyConnectedLines(line: String): String
+ showPathBetween(stationA: String, stationB: String): Stri