UML CLass Diagram FileIO fileToGraph(fileName: String): DSAGraph<String> graphToFile(fileName: String, graph: DSAGraph<String>): void readFile(fileName: String, lines: DSAQueue<String>): void UserInputs + load(fileName: String): DSAGraph<String> UnitTestQueue writeCSV(filename: String, startVertex: String, endVertex: String, userInput(min: int, max: int): int overallPath: DSALinkedList<String>, cost:double): void + userInput(min: double, max: double): double + main(args: String): void lineToGraph(line: String, graph: DSAGraph<String>): void userInput(min: float, max: float): float + testGetCount(): void userInput(min: long, max: long): long + testIsEmpty(): void userInput(min: char, max: char): char UnitTestGraph + testEnqueue(): void + userInput(): String + testDequeue(): void main(args: String[]): void UnitTestLinkedList testConstructor(): void testAddVertex(): void - main(args: String): void testAddVertex(): void + testInsertStart(): void testGetVertexCount(): void + testRemoveStart(): void testGetEdgeCount(): void keyMeUP · interactiveMenu(): void + testInsertLast(): void testGetVertex(): void DSAQueue<T> UnitTestLinkedListIterator + silentMode(keyFile: String, strFile: String, pathFile: String): void + testRemoveLast(): void testGetEdge(): void - main(args: String[]): void interactiveMode(): void testCount(): void testGetAdjacentVertex(): void queue: DSALinkedList<T> main(args: String): void optionFile(): void + testIsEmpty(): void testIsAdjacent(): void + testIteratorConstructor(): void nodeOperation(): void + testPeekStart(): void testDisplayAsList(): void - iterator(): Iterator<T> + testIterating():void edgeOperation(): void + testPeekLast(): void testDisplayAsMatrix(): void + getCount(): int isEmpty(): boolean + enqueue(data: T): void + dequeue(): T + peek(): T «interface» Iterable<T> DSALinkedList<T> **DSAGraphEdge** DSAGraph<T> head: DSAListNode vertices: DSALinkedList from: DSAGraphEdge - to: DSAGraphEdge tail: DSAListNode **DSALinkedListIterator** - edges: DSALinkedList counter: int overallPath: DSALinkedList<T> - label: T «interface» nextNode: DSAListNode - dijkstras: DijkstrasStacks value: Double Iterable<T> iterator(): Iterator<T> current: DSAlinkNode + insertStart(item: Object): void + addVertex(label: T, value: Double): void + getLabel(): T hasNext(): boolean removeStart(): T + addVertex(label: T): void + getValue(): Double + insertLast(item: T): void next(): Object removeVertex(label: T): void getFrom(): DSAGraphVertex + addEdge(fromLabel: T, toLabel: T, edgeValue: Double): void + getTo(): DSAGraphVertex removeLast(): T remove(): void + removeMiddle(nodeData: T): T + addEdge(fromLabel: T, toLabel: T): void + isDirected(): boolean getCount(): int removeEdge(fromLabel: T, toLabel: T): void + toString(): String + isEmpty(): boolean + hasVertex(label: T): boolean -----Use-----+ peekFirst(): T + hasEdge(fromLabel: T, toLabel: T): boolean + peekLast(): + getVertexCount(): int «interface» display(): void + getEdgeCount(): int Iterator<T> **DSAGraphVertex** + getVertex(label: T): DSAGraphVertex + getEdge(label: T): DSAGraphEdge - label: T + getDijkstrasStacks(): DijkstrasStacks - distance: Double setDijkstrasStacks(dijkstrasStack: DijkstrasStacks): void - links: DSALinkedList<T> + getOverallPath(): DSALinkedList<T> visited: boolean DSAListNode + getAdjacentVertex(label: T): DSALinkedList<DSAGraphVertex> - previousVertex: DSAGraphVertex + isAdjacent(label1: T, label2: T): boolean DSAStack<T> «interface» data: T + displayAsList(): void + getLabel(): T - prev: DSAlinkNode Serializable + displayAsMatrix(): void stack: DSALinkedList<T> + getDistance(): Double next: DSAlinkNode + displayInfo(): void getAdjacent(): DSALinkedList<DSAGraphVertex> + BFS(): DSAQueue + iterator(): Iterator<T> - addEdge(vertex: DSAGraphVertex): void getData(): T + DFS(): DSAQueue getCount(): int - setVisited(): void getNext(): DSAListnode + Dijkstras(startLabel: T, endLabel: T): void isEmpty(): boolean + clearVisited(): void setNext(next: DSAListNode): void + displayFinal(): void push(data: T): void getVisited(): boolean + getPrev(): DSAListNode - getNotVisited(vertex: DSAGraphVertex): DSALinkedList<DSAGraphVertex> + pop(): T setDistance(newDistance: Double): void setPrev(prev: DSAListNode): void - tracking(startVertex: DSAGraphVertex, endVertex: DSAGraphVertex): void + top(): T setPreviousVertex(vertex: DSAGraphVertex): void getPreviousVertex(): DSAGraphVertex toString(): String + removeAdjacent(vertex: DSAGraphVertex) DijkstrasStacks compareTo(other: Double): int $\nabla \dot{\nabla} \dot{\nabla}$ - finalStack: DSAStack<T> - distanceStack: DSAStack<Double> «interface» UnitTetstQueue Serializable setFinalStack(finalStack: DSAStack<T>): void main(args: String): void setDistanceStack(distanceStack: DSAStack<Double>): void testGetCount(): void «interface» getFinalStack(): DSAStack<T> testIsEmpty(): void Comparable<T> + getDisanceStack(): DSAStack<Double> testPush(): void testPop(): void