Best Route to deliver package

Team 6 |

Application

FirstPage

+ isPhase1Selected: boolean
+ isPhase2Selected: boolean

+ start(primaryStage: Stage): void
- openSimulationPage(): void
+ main(args: String[]): void

MainGUISimulation

-CounterTimeLabel: Label
+CounterDistanceLabel: Label
+ CounterCostLabel: Label
+ CounterNo_SimulationLabel: Label
+ percentLabelP2: Label
+CounterDistanceLabe1_phase1_to_2 : Label
+ CounterGasolinLabe1_phase1_to_2: Label
+ time_phase1_in2_lable: Label
+Ideal_rectangle_lable: Label
+NAME_OF_Alhamdanya: Label
+ NAME_OF_Almanar: Label
+ NAME_OF_Alfalwah: Label
+Name_house_green: Label
- AllGroups: Pane
- timer: Timer
+secondsPassed: int
+ isStartClicked: boolean
+ isPaused: boolean
+ numOfSumulation: int
+ ChosenBuilding: Rectangle[]
+ car: Rectangle
+ root: StackPane
+ totalTimeP1: int
+ totalTimeP2: int
+originalBuildingColors: Map<Rectangle, Color>

+ start(primaryStage: Stage)
+ initializeBuildingDelivery(): Rectangle[]
+ buildings(AllGroups: Pane):void
+ streets(AllGroups: Pane):void
+ Labels(AllGroups: Pane, isPhase1Selected: Boolean):void
+ Buttons(AllGroups: Pane, primaryStage: Stage):void
+ calculatePercentImprovement(): double
+ calculateTotalTimeForAllPhases():void
+ startSimulation():void
- resetPackageDeliveryStatus():void
- stopCurrentSimulation():void
+ pauseSimulation():void
+ resumeSimulation():void
+ allDelivered(driver: DeliveryDriver): boolean
- formatTime(seconds: int): String
+ formatDistance(distance: double): String
+ formatCounterNo_Simulation(num: int): String
+ formatGasolineCost(cost: double): String
+ formatPercent(percent: double): String
- openFirstPage():void
+ endSimulation():void
+ restSumation():void
- initializeOriginalColors():void
- resetChosenBuildingColors():void
+ openPhase2():void
+TooltipOfBuildings():void
+ addToolTipsToAllBuildings(): void
+ highlightBuildingsWithPackages(): void
+ updatePhase1Metrics (): void
+ resetBuildingColors():void
+ openPhase2 () : void

MainProgram

+ driver : DeliveryDriver
+ destetionBuilding : List<Intersection>
+ choosenBuilding : List<Intersection>
+ customers : List<Customer>
+ packages : List<Package>
+ intersections : List<List<Intersection>>
+ graph : Map<Intersection,List<Intersection>>
+ organizeBuilding : List<Intersection>

+ initializeObjects() : void
+ destinationBuildings() : List<Intersection>
+ getShuffledBuildings() : List<Intersection>
+ getOrganizedBuildings() : List<Intersection>
+ createGraph() : Map<Intersection,List<Intersection>>
- getOrderedConnectionKey(Intersection intersection1, Intersection intersection2) : String
+ createIntersectionsLayout() : List<List<Intersection>>
+ copyPackages(List<Package> originalPackages) : List<Package>
+ AllBuildings() : List<Building>
+ generateRandomCustomers() : List<Customer>
+ generateRandomPackages(List<Customer> customers) : List<Package>
- isPackageIdUse(List<Package> packages, int packageId) : boolean
- connectDestinationBuildings(Map<Intersection, List<Intersection>> graph, List<Intersection> intersections, List<Intersection> destinationBuildings) : void
- findClosestLeftIntersection(Intersection destination, List<Intersection> intersections) : Intersection
- findClosestRightIntersection(Intersection destination, List<Intersection> intersections) : Intersection
+ initializeChoosenIntersections() : List<Intersection>

Customer

- ID: int
- building: Building
- assignedPackages : List <Package>

+ Customer(ID:int, building:Building)
+ getBuilding(): Building
+ getID(): int
+ getAssignedPackages() : List<Package>
+ addAssignedPackage (Package newpackage) : void

Building

- buildingNumber: int
- location: Intersection
- guiElement: Rectangle
- Package : assignedPackage

+ Building(buildingNumber: int, location: Intersection, guiElement: Rectangle)
+ getBuildingNumber(): int
+ getLocation(): Intersection
+ getGuiElement(): Rectangle
+ assignPackage (pkg : Package) : void
+ getAssignedPackage() : Package

Rectangle

CustomerWithContact

- customerNumber: String

+ CustomerWithContact(Customer original)
+ CustomerWithContact(CustomerWithContact original)
+ getPhoneNumber(): String

<<interface>> reashable

+getPhoneNumber():String

Intersection

- x: int
- y: int

+ Intersection(x:int, y:int)
+ getX(): int
+ getY(): int
+ getDistanceTo(nextIntersection: Intersection): double
+ calculateGasolineCost(nextIntersection: Intersection): double

Package

- customer1: Customer
+ isDelivered: boolean
- packageId: int
- path: List<Intersection>
+ delay: int

+ Package(customer1: Customer, packageId: int, path: List<Intersection>, delay: int)
+ Package(original: Package, newPath: List<Intersection>)
+ *getPackageInformation()*: String
+ getPackageId(): int
+ getCustomer(): Customer
+ getPath(): List<Intersection>
+ setPath(path: List<Intersection>): void

DeliveryDriver

- packages: List<Package>
- currentX: int
- currentY: int
+ pathTransition: PathTransition
- gasolineCost: double
- incrementGasoline: double
- incrementDistance: double
- distance: double
+isTransitionPaused:boolean

+ setGasolineCost(gasolineCost: double): void
+ setDistance(distance: double): void
+ setPackages(packages: List<Package>): void
+ getPackages(): List<Package>
+ calculateTotalDistance(packages: List<List<Intersection>>): double
- delayByIndex(index: int): int
+ calculateTotalGasolineCost(packages: List<List<Intersection>>): double
+ moveCarTo(packages: List<List<Intersection>>): void
+ calculateShortestPathsBetweenDestinations(List<Intersection> destinations) : List<List<Intersection>>
+ findShortestPath(Map<Intersection, List<Intersection>> graph, Intersection start, Intersection destination) : List<Intersection>
+ createPathForPackages(packages: List<List<Intersection>>): void
- playPathTransitions(packages: List<List<Intersection>>, index: int): void
- generatePath(Intersections: List<Intersection>): Path
+ calcluateTotalTime(): int
+ moveDriver(path: Path, onFinish: Runnable, delay: int): void
- calculateTotalDistance(elements: ObservableList<PathElement>): double |-
deliverPackage(currentX: int, currentY: int): void
- moveToNextPackage(): void
- hasNextPackage(currentPackage: Package): boolean

PathTransition

Normal

+ Normal(customer: Customer, packageId: int, path: List<Intersection>, delay: int)
+ Normal(original: Normal, newPath: List<Intersection>)
+ getPackageInformation(): String
- getTypeOfPackage(): String

Offical_paper

+ Offical_paper(customer: Customer, packageId: int, path: List<Intersection>, delay: int)
+ Offical_paper(original: Offical_paper, newPath: List<Intersection>)
+ getPackageInformation(): String
+ contactCustomer(): void
- getTypeOfPackage(): String