Trip<E>

-from :City<E>
-to: City<E>

+<<Constructor>>Trip
+getTo() : City<E>
+getFrom():City<E>
+getCost():int
+setCost():int

State<E>

-currentNode:City<E>
-pState:State<E>
-gCost:int
-hCost:int
-fCost:int

+<<Constructor>>State
+getPathString(s:String):String
+printCurrentPathAndCosts():void
+getCurrentNode():City<E>
+calculateFcost():int
+getPstate():State<E>
+getgCost():int
+setgCost(gCost:int):void
+sethCost(hCost:int):void
+gethCost():int
+setfCost():void
+getfCost():int

TripPlanner<E>

-cities:ArrayList<City<E>>
-finalState:State<E>
-trips:ArrayList<Trip<E>>
-nnodes: int
-startCity: City<E>
-heuristic:LegitHeuristic<E>

+main
+cityInit(n: City<E>) : void
+addtrip(t: trip<E>) : void
+addconnection(input:String[]):void
+setStartCity(start:City<E>):void
+initStartCity() : void
+setNnodes(nodes:int):void
+setFinalState(current :State<E>) :void
+copylist() : ArrayList<Trip<E>>
+addEdge(from :City<E>, to: City<E>, c:int):void
+getaCity(name:String):City<E>
+getTrips() : ArrayList<Trip<E>>
+closestEdge(): int
+printresult() :void
+aStarSearch():void

<<Interface>>
Heuristic

+getEstimate(tripplanner:TripPlanner, state:State<E>):int

LegitHeuristic

+<<Constructor>LegitHeuristic
+getEstimate(t: TripPlanner<E>, s :State<E>):int
+getnumofLinkTrip(unvisited:ArrayList<Trip<E>>, c:City<E>):int
+getEdgeCost(trips:ArrayList<Trip<E>>) :int
+getlefttransfer(unvisited:ArrayList<E>>):int
+isEnd(c:City<E>) :boolean

City<E>

-name : E
-edge:ArrayList<Edge<E>>
-transfer:int

+<<Constructor>>City<E>
+addEdge(from:City<E>,to:City<E>,cost:int):void
+getConnected():ArrayList<City<E>>
+addEdge(to:City<E>):Edge<E>
+getName():E
+getEdgeCost(to:City<E>):int
+getEdge():ArrayList<Edge<E>>
+getTransfer():int

Edge<E>

-from:City<E>
-to :City<E>
-cost: int

+<<Constructor>>Edge<E>
+getTo():City<E>
+getFrom():City<E>
+getOtherEnd(node:City<E>
+getCost():int