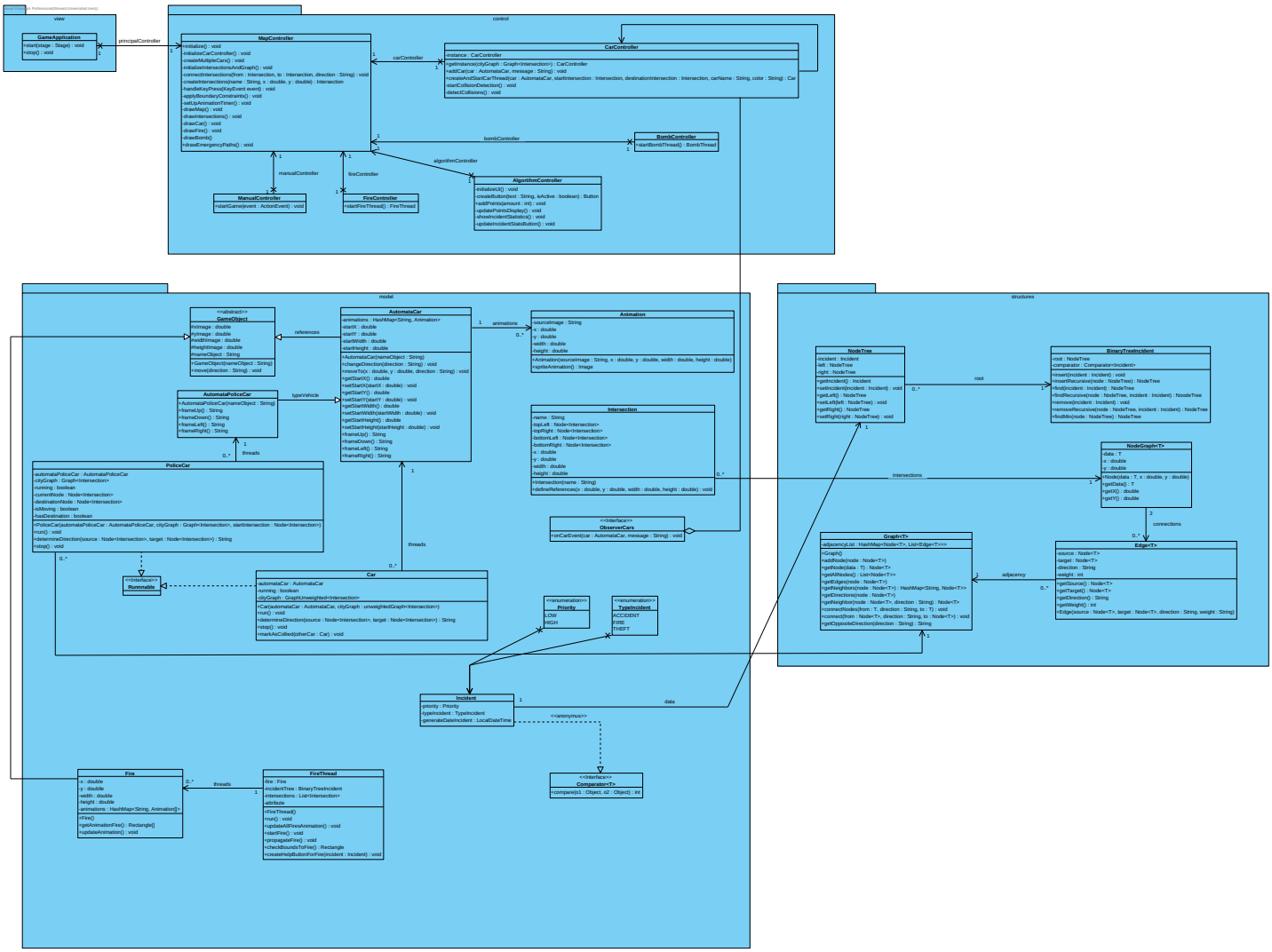
view
GameApplication
+start(pStage : Stage) : void
+stop() : void
principalController
control
MapController
+initialize() : void
+initializeCarController() : void
-createMultipleCars() : void
-initializeIntersectionsAndGraph() : void
-connectIntersections(from : Intersection, to : Intersection, direction : String) : void
-createIntersections(name : String, x : double, y : double) : Intersection
-handleKeyPress(KeyEvent event) : void
-applyBoundaryConstraints() : void
-setUpAnimationTimer() : void
-drawMap() : void
-drawIntersections() : void
-drawCar() : void
-drawFire() : void
-drawBomb()
-drawEmergencyPaths() : void
carController
CarController
-instance : CarController
+getInstance(cityGraph : Graph<Intersection>) : CarController
+addCar(car : AutomataCar, message : String) : void
+createAndStartCarThread(car : AutomataCar, startIntersection : Intersection, destinationIntersection : Intersection, carName : String, color : String) : Car
-startCollisionDetection() : void
-detectCollisions() : void
bombController
BombController
+startBombThread() : BombThread
manualController
ManualController
+startGame(event : ActionEvent) : void
fireController
FireController
+startFireThread() : FireThread
algorithmController
AlgorithmController
+initializeUI() : void
+createButton(text : String, isActive : boolean) : Button
+addPoints(amount : int) : void
-updatePointsDisplay() : void
-showIncidentStatistics() : void
-updateIncidentStatsButton() : void
model
<<abstract>>
GameObject
#xImage : double
#yImage : double
#widthImage : double
#heightImage : double
#nameObject : String
+GameObject(nameObject : String)
+move(direction : String) : void
references
AutomataCar
-animations : HashMap<String, Animation>
-startX : double
-startY : double
-startWidth : double
-startHeight : double
+AutomataCar(nameObject : String)
+changeDirection(direction : String) : void
+moveTo(x : double, y : double, direction : String) : void
+getStartX() : double
+setStartX(startX : double) : void
+getStartY() : double
+setStartY(startY : double) : void
+getStartWidth() : double
+setStartWidth(startWidth : double) : void
+getStartHeight() : double
+setStartHeight(startHeight : double) : void
+frameUp() : String
+frameDown() : String
+frameLeft() : String
+frameRight() : String
1
animations
0..*
Animation
-sourceImage : String
-x : double
-y : double
-width : double
-height : double
+Animation(sourceImage : String, x : double, y : double, width : double, height : double)
+spriteAnimation() : Image
AutomataPoliceCar
+AutomataPoliceCar(nameObject : String)
+frameUp() : String
+frameDown() : String
+frameLeft() : String
+frameRight() : String
typeVehicle
threads
1
0..*
PoliceCar
-automataPoliceCar : AutomataPoliceCar
-cityGraph : Graph<Intersection>
-running : boolean
-currentNode : Node<Intersection>
-destinationNode : Node<Intersection>
-isMoving : boolean
-hasDestination : boolean
+PoliceCar(automataPoliceCar : AutomataPoliceCar, cityGraph : Graph<Intersection>, startIntersection : Node<Intersection>)
+run() : void
+determineDirection(source : Node<Intersection>, target : Node<Intersection>) : String
+stop() : void
Intersection
-name : String
-topLeft : Node<Intersection>
-topRight : Node<Intersection>
-bottomLeft : Node<Intersection>
-bottomRight : Node<Intersection>
-x : double
-y : double
-width : double
-height : double
+Intersection(name : String)
+defineReferences(x : double, y : double, width : double, height : double) : void
0..*
<<interface>>
ObserverCars
+onCarEvent(car : AutomataCar, message : String) : void
<<interface>>
Runnable
Car
-automataCar : AutomataCar
-running : boolean
-cityGraph : GraphUnweighted<Intersection>
+Car(automataCar : AutomataCar, cityGraph : unweightedGraph<Intersection>)
+run() : void
+determineDirection(source : Node<Intersection>, target : Node<Intersection>) : String
+stop() : void
+markAsCollided(otherCar : Car) : void
<<enumeration>>
Priority
LOW
HIGH
<<enumeration>>
TypeIncident
ACCIDENT
FIRE
THEFT
Incident
-priority : Priority
-typeIncident : TypeIncident
-generateDateIncident : LocalDateTime
data
<<anonymous>>
<<interface>>
Comparator<T>
+compare(o1 : Object, o2 : Object) : int
Fire
-x : double
-y : double
-width : double
-height : double
-animations : HashMap<String, Animation[]>
+Fire()
+getAnimationFire() : Rectangle[]
+updateAnimation() : void
threads
FireThread
-fire : Fire
-incidentTree : BinaryTreeIncident
-intersections : List<Intersection>
-attribute
+FireThread()
+run() : void
+updateAllFiresAnimation() : void
+startFire() : void
+propagateFire() : void
+checkBoundsToFire() : Rectangle
+createHelpButtonForFire(incident : Incident) : void
structures
NodeTree
-incident : Incident
-left : NodeTree
-right : NodeTree
+getIncident() : Incident
+setIncident(incident : Incident) : void
+getLeft() : NodeTree
+setLeft(left : NodeTree) : void
+getRight() : NodeTree
+setRight(right : NodeTree) : void
root
0..*
1
BinaryTreeIncident
-root : NodeTree
-comparator : Comparator<Incident>
+insert(incident : Incident) : void
+insertRecursive(node : NodeTree) : NodeTree
+find(incident : Incident) : NodeTree
+findRecursive(node : NodeTree, incident : Incident) : NodeTree
+remove(incident : Incident) : void
+removeRecursive(node : NodeTree, incident : Incident) : NodeTree
+findMin(node : NodeTree) : NodeTree
intersections
NodeGraph<T>
-data : T
-x : double
-y : double
+Node(data : T, x : double, y : double)
+getData() : T
+getX() : double
+getY() : double
connections
GraphT>
-adjacencyList : HashMap<Node<T>, List<Edge<T>>>
+Graph()
+addNode(node : Node<T>)
+getNode(data : T) : Node<T>
+getAllNodes() : List<Node<T>>
+getEdges(node : Node<T>)
+getNeighbors(node : Node<T>) : HashMap<String, Node<T>>
+getDirections(node : Node<T>)
+getNeighbor(node : Node<T>, direction : String) : Node<T>
+connectNodes(from : T, direction : String, to : T) : void
+connect(from : Node<T>, direction : String, to : Node<T>) : void
+getOppositeDirection(direction : String) : String
adjacency
Edge<T>
-source : Node<T>
-target : Node<T>
-direction : String
-weight : int
+getSource() : Node<T>
+getTarget() : Node<T>
+getDirection() : String
+getWeight() : int
+Edge(source : Node<T>, target : Node<T>, direction : String, weight : String)