«interface» SchedulerSate
+ handle(context:SchedulerStateMachine, eventStatus: EventStatus, requestData: String, requestPacket: DatagramPacket): void

Error
+ handle(context: droneStateMachine):void

IdleState
+ handle(context: droneStateMachine):void

Completed
+ handle(context: droneStateMachine):void

Fire
+ handle(context: droneStateMachine):void

Confirmation
+ handle(context: droneStateMachine):void

DroneStatus
- droneId, port: int
- state: String
- currentFire: FireEvent
- fireComplete: boolean
+ DroneStatus(droneId:int, port: int, state: String, currentFire: FireEvent)
+ getDroneId(): int
+ getPort(): int
+ getState(): String
+ getCurrentFire(): FireEvent
+ setState(state: String): void
+ setCurrentFire(currentFire: FireEvent): void
+ isFireComplete(): boolean

EventStatus
- command: String
- droneStatus: DroneStatus
+ EventStatus(command:String)
+ EventStatus(command:String, droneStatus: DroneStatus)
+ getCommand(): String
+ getDroneStatus(): DroneStatus

BoundedBuffer
+ SIZE: int
- buffer: Object[]
- inIndex, outIndex, count: int
- writeable: boolean
- readable: boolean
+ addLast(item: object): void {synchronized}
+ removeFirst(): Object {synchronized}
+ getCount(): int {synchronized}
+ toString(): String
+ removeFireEventByID(fireID: int): Object {synchronized}

SchedulerStateMachine
- currentState: SchedulerState
- scheduler: Scheduler
- states: Map<String,SchedulerState>
+ SchedulerStateMachine(scheduler: Scheduler)
+ handleState(eventStatus: EventStatus, requestData: String, requestPacket:DatagramPacket): void
+ setState(stateName:String): void
+ getState(): String
+ getScheduler(): Scheduler

FireEvent
- time: String
- zoneId: int
- eventType: String
- severity: String
- remainingWaterNeeded: int
- fireID: int
- failure: String
- state: String
+ FireEvent(fireID:int, time: String, zoneId: int, eventType: String, severity: String, failure: String)
+ getTime(): String
+ getZoneId(): int
+ getEventType(): String
+ getSeverity(): String
+ getFireID(): int
+ extinguish(agentAmount: int): void
+ getRemainingWaterNeeded(): int
+ getState(): String
+ getFailureFlag(): boolean
+ getWaterRequired(): int
+ toString(): String {override}

«Runnable» DroneSubsystem
- droneID, agentLevel, DRONE_PORT: int
- currentFireEvent: FireEvent
- lastFireEvent: FireEvent
- fireEventComplete: Boolean
- MAX_AGENT_CAP, SPEED, BASE_PORT, SCHEDULER_PORT: int
- sendSocket, receiveSocket: DatagramSocket
- droneX, droneY: double
- state: String
- BASE_ZONE: Zone
+ DroneSubsystem(droneID: int)
+ run(): void {override}
+ rpc_send(requestPacket:DatagramPacket, receivePacket: DatagramPacket): void {synchronzied}
+ fetchFireTask(): FireEvent
+ parseDataToFireEvent(data:String): FireEvent
+ sendAck(ack: String): void
+ returnFireCompleted(): void
+ returnFailure(): void
+ refillAgent(): void
+ simulateDroneTravel(zone:Zone): void
+ isAgentEmpty(): boolean
+ completeTask(task:FireEvent): void
+ toString(): String {override}
+ main(args: String[]): void

Zone
- ID: int
- startX, startY, endX, endY: double
+ Zone(name:int, startX: double, startY: double, endX:double, endY:double)
+ getID(): int
+ getStartX(): double
+ getStartY(): double
+ getEndX(): double
+ getEndY(): double

«Runnable» Scheduler
- zoneMap: Map<Integer,Zone>
- RECEIVE_PORT: int
- receiveSocket, sendSocket: DatagramSocket
- fireToDroneBuffer, droneToFireBuffer: BoundedBuffer
- drones: LinkedList<DroneStatus>
- receiveThread: Thread
- zoneFilePath: String
+ Scheduler()
+ run(): void {override}
+ RCP_Receive(requestPacket: DatagramPacket): void
+ handleEvent(data:String): EventStatus
+ extractFireEvent(data: String): String
+ getDronePort(droneID: int): int
+ getAvailableDrone(): DroneStatus {synchronized}
+ updateDroneState(doneID: int, newState: String): void {synchronized}
+ addSortFires(droneID: int, newState: String): void
+ loadZonesFromCSV(filePath: String): Map<Integer, Zone>
+ getZone(zoneID: int): Zone
+ main(args: String[]): void

«Runnable» FireIncidentSubsystem
- csvFilePath: String
- sendReceiveSocket: DatagramSocket
- SCHEDULER_PORT: int
- nextFireID: int
- port: int
+ FireIncidentSubsystem(csvFilePath: String)
+ run(): void {override}
+ extractFireEventFromLine(line:String): FireEvent
- rpc_send(dataPacket: DatagramPacket, replyPacket: DatagramPacket, fireEvent: FireEvent): void
+ getPort(): int
+ main(args: String[]): void

Simulation
+ main(args: String[]): void

DroneStateMachine
- states: Map<String, DroneState>
- currentState: DroneState
- drone: DroneSubsystem
+ DroneStateMachine(drone:DroneSubsystem)
+ handleState(): void
+ setState(stateName: String): void
+ getDrone(): DroneSubsystem

Faulted
+ handle(context: droneStateMachine):void

Idle
+ handle(context: droneStateMachine):void

EnRoute
+ handle(context: droneStateMachine):void

DroppingAgent
+ handle(context: droneStateMachine):void

Refilling
+ handle(context: droneStateMachine):void

«interface» DroneState
+ handle(context:DroneStateMachine): void