Creator luo uuden Airportin perustuen configtiedostoon, jolla peli määritetään. Airport-luokka Runway on "pelin-ydin" ja pyörittää kaikkia pelin val length: Int tapahtumia. var usable: Boolean Gate var isInUse: Boolean val number: Int var condition: Double val plane: Option[Airplane] val number: Int var isInUse: Boolean var plane: Option[Airplane] Flight reserve(airplane: Airplane): Unit reserve(airplane: Airplane): Unit val destination: String Creator unreserve: Unit unreserve: Unit val departure: String createAirport + field: type + field: type val shortForm: String crateFlight val flightTime: Int createAirplane val plane: Airplane var completion: Double + field: type Airport var time: Int Airplane var planes: Buffer[Airplane] val airline: String var runways: Buffer[Runway] var fuel: Int Tässä kaaviossa ei oteta huomioon var crossingRunways: Map[Runway --> Vector[Runways]] var fuelCapacity: Int var fuelLevel = fuel / fuelCapacity : Double var gates: Vector[Gate] graafista käyttöliittymää, val queuesOnGround: VectorLandQueue] val fuelConsumption: Double vaan enneminkin kuvataan val queuesInAir: Vector[InAirQueue] var altitude: Int var points: Int var timeToDestination: Int muun ohjelman luokkarakennetta. var endingReason: Option val totalCargoWeight: Int var notifications: Buffer[String] val minRunwayLength: Int checkPlanes: Unit val passengers: Int checkRunways: Unit var currentFlight: Option[Flight] addNotification: Unit var nextFlight: Option[Flight] changeAltitude(newAltitude: Int): Unit createPlane(): Unit onTick():Unit moveAirplane: Unit Getters checkAirplane: Unit crash: Unit ascend(runway: Runway) ascendingOperations: Unit descend(runway: Runway) descendingOperations: Unit sendToQueue(number: Int) <<Abstract>> Queue queueOperations: Unit val planes: Buffer[Planes] sendToGate(number: Int) val capacity: Int refuel: Unit val idNumber: String + field: type InAirQueue val altitude: Int + field: type + field: type + field: type

> + field: type + field: type