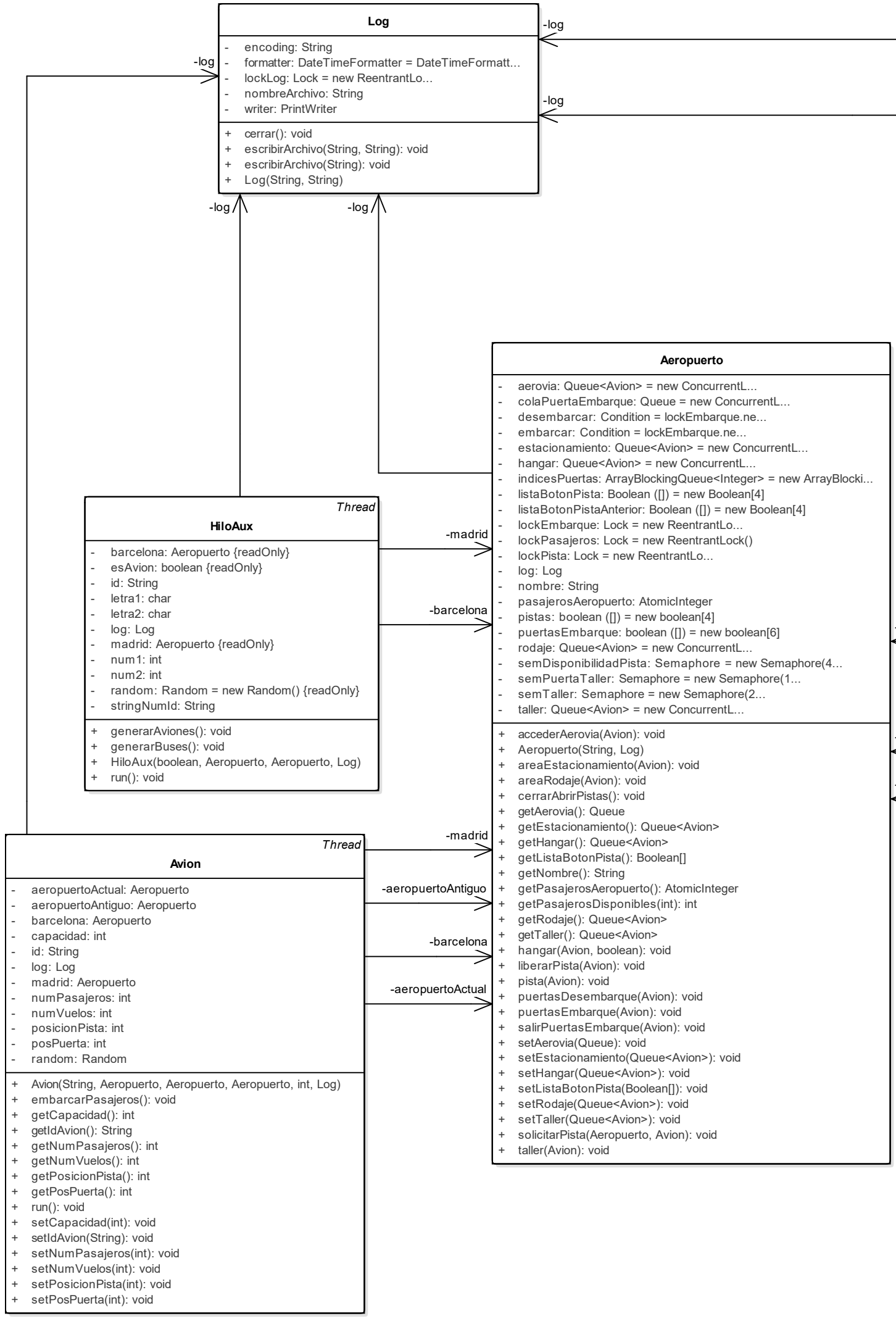
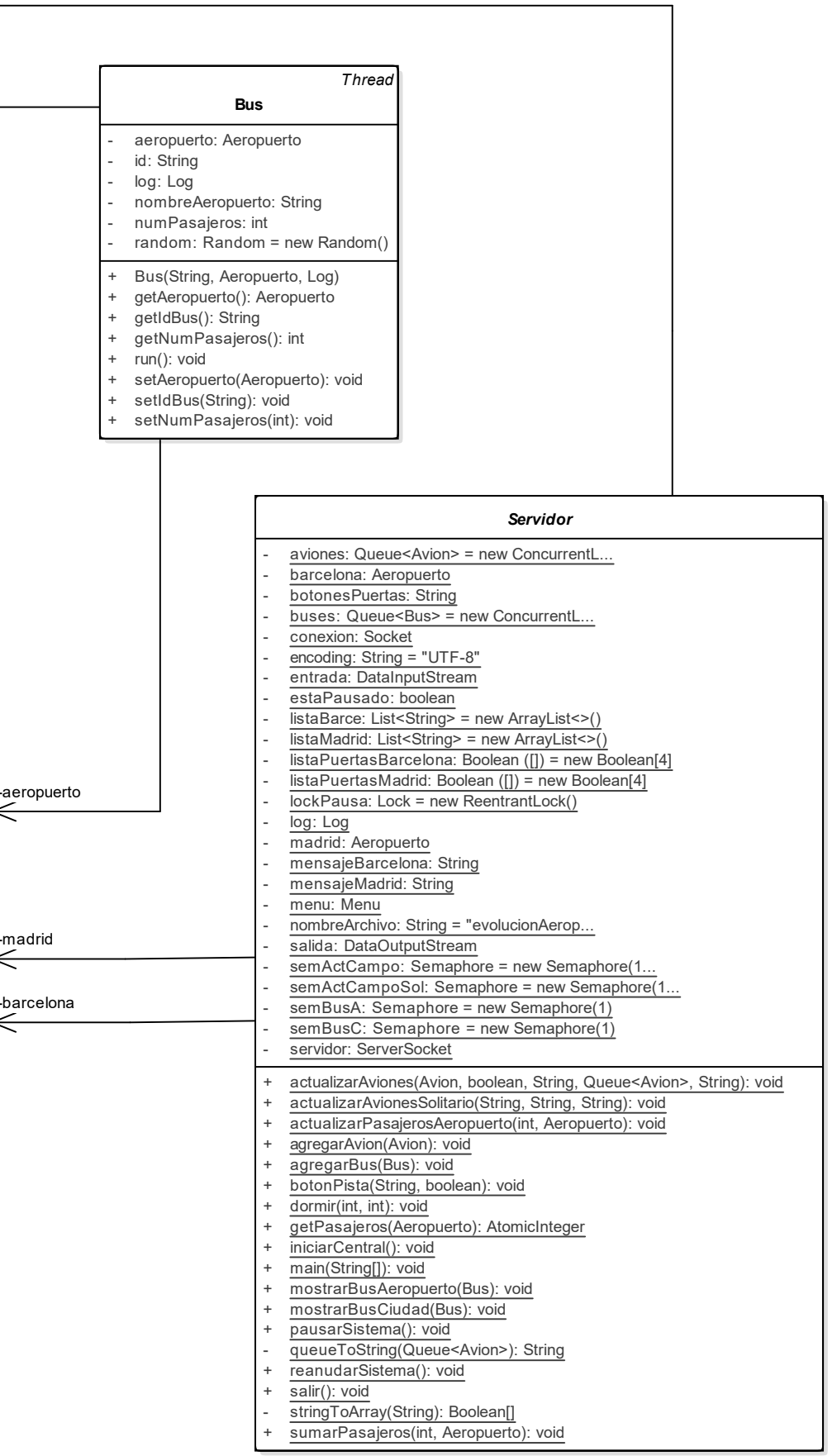

Log
- encoding: String
- formatter: DateTimeFormatter = DateTimeFormatt...
- lockLog: Lock = new ReentrantLo...
- nombreArchivo: String
- writer: PrintWriter
+ cerrar(): void
+ escribirArchivo(String, String): void
+ escribirArchivo(String): void
+ Log(String, String)
-log
-log
-log
-log
-log
Thread
HiloAux
- barcelona: Aeropuerto {readOnly}
- esAvion: boolean {readOnly}
- id: String
- letra1: char
- letra2: char
- log: Log
- madrid: Aeropuerto {readOnly}
- num1: int
- num2: int
- random: Random = new Random() {readOnly}
- stringNumId: String
+ generarAviones(): void
+ generarBuses(): void
+ HiloAux(boolean, Aeropuerto, Aeropuerto, Log)
+ run(): void
-madrid
-barcelona
Aeropuerto
- aerovia: Queue<Avion> = new ConcurrentL...
- colaPuertaEmbarque: Queue = new ConcurrentL...
- desembarcar: Condition = lockEmbarque.ne...
- embarcar: Condition = lockEmbarque.ne...
- estacionamiento: Queue<Avion> = new ConcurrentL...
- hangar: Queue<Avion> = new ConcurrentL...
- indicesPuertas: ArrayBlockingQueue<Integer> = new ArrayBlocki...
- listaBotonPista: Boolean ([]) = new Boolean[4]
- listaBotonPistaAnterior: Boolean ([]) = new Boolean[4]
- lockEmbarque: Lock = new ReentrantLo...
- lockPasajeros: Lock = new ReentrantLock()
- lockPista: Lock = new ReentrantLo...
- log: Log
- nombre: String
- pasajerosAeropuerto: AtomicInteger
- pistas: boolean ([]) = new boolean[4]
- puertasEmbarque: boolean ([]) = new boolean[6]
- rodaje: Queue<Avion> = new ConcurrentL...
- semDisponibilidadPista: Semaphore = new Semaphore(4...
- semPuertaTaller: Semaphore = new Semaphore(1...
- semTaller: Semaphore = new Semaphore(2...
- taller: Queue<Avion> = new ConcurrentL...
+ accederAerovia(Avion): void
+ Aeropuerto(String, Log)
+ areaEstacionamiento(Avion): void
+ areaRodaje(Avion): void
+ cerrarAbrirPistas(): void
+ getAerovia(): Queue
+ getEstacionamiento(): Queue<Avion>
+ getHangar(): Queue<Avion>
+ getListaBotonPista(): Boolean[]
+ getNombre(): String
+ getPasajerosAeropuerto(): AtomicInteger
+ getPasajerosDisponibles(int): int
+ getRodaje(): Queue<Avion>
+ getTaller(): Queue<Avion>
+ hangar(Avion, boolean): void
+ liberarPista(Avion): void
+ pista(Avion): void
+ puertasDesembarque(Avion): void
+ puertasEmbarque(Avion): void
+ salirPuertasEmbarque(Avion): void
+ setAerovia(Queue): void
+ setEstacionamiento(Queue<Avion>): void
+ setHangar(Queue<Avion>): void
+ setListaBotonPista(Boolean[]): void
+ setRodaje(Queue<Avion>): void
+ setTaller(Queue<Avion>): void
+ solicitarPista(Aeropuerto, Avion): void
+ taller(Avion): void
Thread
Avion
- aeropuertoActual: Aeropuerto
- aeropuertoAntiguo: Aeropuerto
- barcelona: Aeropuerto
- capacidad: int
- id: String
- log: Log
- madrid: Aeropuerto
- numPasajeros: int
- numVuelos: int
- posicionPista: int
- posPuerta: int
- random: Random
+ Avion(String, Aeropuerto, Aeropuerto, Aeropuerto, int, Log)
+ embarcarPasajeros(): void
+ getCapacidad(): int
+ getIdAvion(): String
+ getNumPasajeros(): int
+ getNumVuelos(): int
+ getPosicionPista(): int
+ getPosPuerta(): int
+ run(): void
+ setCapacidad(int): void
+ setIdAvion(String): void
+ setNumPasajeros(int): void
+ setNumVuelos(int): void
+ setPosicionPista(int): void
+ setPosPuerta(int): void
-madrid
-aeropuertoAntiguo
-barcelona
-aeropuertoActual


Thread
Bus
- aeropuerto: Aeropuerto
- id: String
- log: Log
- nombreAeropuerto: String
- numPasajeros: int
- random: Random = new Random()
+ Bus(String, Aeropuerto, Log)
+ getAeropuerto(): Aeropuerto
+ getIdBus(): String
+ getNumPasajeros(): int
+ run(): void
+ setAeropuerto(Aeropuerto): void
+ setIdBus(String): void
+ setNumPasajeros(int): void
Servidor
- aviones: Queue<Avion> = new ConcurrentL...
- barcelona: Aeropuerto
- botonesPuertas: String
- buses: Queue<Bus> = new ConcurrentL...
- conexion: Socket
- encoding: String = "UTF-8"
- entrada: DataInputStream
- estaPausado: boolean
- listaBarce: List<String> = new ArrayList<>()
- listaMadrid: List<String> = new ArrayList<>()
- listaPuertasBarcelona: Boolean ([]) = new Boolean[4]
- listaPuertasMadrid: Boolean ([]) = new Boolean[4]
- lockPausa: Lock = new ReentrantLock()
- log: Log
- madrid: Aeropuerto
- mensajeBarcelona: String
- mensajeMadrid: String
- menu: Menu
- nombreArchivo: String = "evolucionAerop...
- salida: DataOutputStream
- semActCampo: Semaphore = new Semaphore(1...
- semActCampoSol: Semaphore = new Semaphore(1...
- semBusA: Semaphore = new Semaphore(1)
- semBusC: Semaphore = new Semaphore(1)
- servidor: ServerSocket
+ actualizarAviones(Avion, boolean, String, Queue<Avion>, String): void
+ actualizarAvionesSolitario(String, String, String): void
+ actualizarPasajerosAeropuerto(int, Aeropuerto): void
+ agregarAvion(Avion): void
+ agregarBus(Bus): void
+ botonPista(String, boolean): void
+ dormir(int, int): void
+ getPasajeros(Aeropuerto): AtomicInteger
+ iniciarCentral(): void
+ main(String[]): void
+ mostrarBusAeropuerto(Bus): void
+ mostrarBusCiudad(Bus): void
+ pausarSistema(): void
- queueToString(Queue<Avion>): String
+ reanudarSistema(): void
+ salir(): void
- stringToArray(String): Boolean[]
+ sumarPasajeros(int, Aeropuerto): void
-aeropuerto
-madrid
-barcelona