```
    1: /**
    2:  *  AirportServer.h
    3:  *  This class defines the methods called by the Airplanes
    4:  */
    5: #ifndef AIRPORT_SERVER_H
    6: #define AIRPORT_SERVER_H
    7:
    8: #include <mutex>
    9:
   10: #include <random>
   11:
   12: #include <condition_variable>
   13:
   14: #include "AirportRunways.hpp"
   15:
   16:
   17: class AirportServer {
   18:     public:
   19:
   20:         /**
   21:          * Default constructor for AirportServer class
   22:          */
   23:         AirportServer() {
   24:             // ***** Initialize any Locks and/or Condition Variables here as
 necessary *****
   25:             lck15L = std::unique_lock < std::mutex > (run15L);
   26:             lck15R = std::unique_lock < std::mutex > (run15R);
   27:             lck4L = std::unique_lock < std::mutex > (run4L);
   28:             lck4R = std::unique_lock < std::mutex > (run4R);
   29:             lck14 = std::unique_lock < std::mutex > (run14);
   30:             lck9 = std::unique_lock < std::mutex > (run9);
   31:
   32:         } // end AirportServer default constructor
   33:
   34:     /**
   35:      * Called by an Airplane when it wishes to land on a runway
   36:      */
   37:     void reserveRunway(int airplaneNum, AirportRunways::RunwayNumber runway)
;
   38:     /**
   39:      * Called by an Airplane when it is finished landing
   40:      */
   41:     void releaseRunway(int airplaneNum, AirportRunways::RunwayNumber runway)
;
   42:
   43:     private:
   44:
   45:         // Constants and Random number generator for use in Thread sleep cal
ls
   46:         static
   47:     const int MAX_TAXI_TIME = 10; // Maximum time the airplane will occupy t
he requested runway after landing, in milliseconds
   48:     static
   49:     const int MAX_WAIT_TIME = 100; // Maximum time between landings, in mill
iseconds
   50:
   51:     /**
   52:     AirportServer.h Tue Apr 23 19:36:55 2019 2
   53:     * Declarations of mutexes and condition variables
   54:     */
   55:     mutex runwaysMutex; // Used to enforce mutual exclusion for acquiring &
```

releasing runways

```
   56:     /**
   57:     * ***** Add declarations of your own Locks and Condition Variables
   58:      here *****
   59:     */
   60:     std::mutex run15L;
   61:     std::mutex run15R;
   62:     std::mutex run4L;
   63:     std::mutex run4R;
   64:     std::mutex run14;
   65:     std::mutex run9;
   66:
   67:     std::unique_lock < std::mutex > lck15L;
   68:     std::unique_lock < std::mutex > lck15R;
   69:     std::unique_lock < std::mutex > lck4L;
   70:     std::unique_lock < std::mutex > lck4R;
   71:     std::unique_lock < std::mutex > lck14;
   72:     std::unique_lock < std::mutex > lck9;
   73:
   74:     std::condition_variable cv;
   75:
   76: }; // end class AirportServer
   77:
   78: #endif
```