Simulation Outline

Airport Class:

The Airport class will simulate the operation of an airport. It will manage the landing and takeoff queues; control the assignment and release of airplanes to/from the runway; keep track of landings, takeoffs, and crashes; and calculate report statistics.

Fields:

landingProbability: probability of an airplane entering the landing queue
takeoffProbability: probability of an airplane entering the takeoff queue
maxLandingQueueWait: maximum time an airplane can remain in the landing queue without running out of fuel and crashing (in minutes)
runway: an object of the Runway class, used for landings and takeoffs
landingCount: total number of planes that landed in the simulation
takeoffCount: total number of planes that took off in the simulation
crashCount: total number of planes that crashed as a result of waiting in the landing queue too long
timeInLandingQueue: accumulator for the total time spent in the landing queue for all planes
timeInTakeoffQueue: accumulator for the total time spent in the take off queue for all planes

Methods:

public static void main(String[] args)

Will create an instance of the Airport class, and use the reference variable to: call the inputData() method to get the parameters needed to run the simulation, call the runSimulation method to run the simulation, and call the ouputResults method to display the results of running the simulation.
Preconditions: none
Postconditions: the simulation has run to completion and the desired statistics have been returned.

public void inputData()

Prompt the user to enter the following input data in the specified order:
The amount of time needed for one plane to land (in minutes): int landingTime
The amount of time needed for one plane to take off (in minutes): int takeOffTime
The probability of an arrival of a plane into the landing queue (this will be a decimal number between 0 and 1): double arrivalProbability
The probability of an arrival of a plane into the takeoff queue (this will be a decimal number between 0 and 1): double takeoffProbability
The maximum amount of time that a plane can stay in the landing queue without running out of
Preconditions: None.
Postconditions: The user has provided input values for the simulation.

public void runSimulation(int simulationLength)

Will initialize variables and counters for tracking the number of planes that took off, landed, and crashed.
Enter a loop that simulates the passage of time in minutes, starting from minute 1 and continuing until the specified total length of time to be simulated is reached.
Check if the runway is currently occupied.
Check if a plane should arrive in the landing queue based on the specified probability of an arrival.
Check if a plane should arrive in the takeoff queue based on the specified probability of an arrival.
Preconditions: The airport and its components (runway, landing queue, takeoff queue) are initialized and set up with appropriate values.
Postconditions: The simulation of the airport is completed, and the results are ready for output.
Parameter: simulationLength is an integer to represent the total length of time to simulate in minutes.

public void outputResults()

Will print out the following statistics:
Output the number of planes that took off in the simulated time.
Output the number of planes that landed in the simulated time.
Output the number of planes that crashed because they ran out of fuel before they could land.
Output the average time that a plane spent in the takeoff queue
Output the average time that a plane spent in the landing queue
Precondition: The simulation has completed and the required data has been saved.
Postcondition: the following information has been printed out to the console:
the number of planes that took off in the simulation time,
the number of planes that landed in the simulation time,
the number of planes that exceed the maximum length of time allowed in the queue, and crashed,
the average length of time that a plane spent in the takeoff queue,
the average length of time that a plane spent in the landing queue.

Airplane Class:

The Airplane class will represent an airplane. It will store information; including arrival time and remaining fuel; provide methods to retrieve the arrival time and fuel remaining; and may contain additional methods related to airplane operations, such as refueling or updating the remaining fuel.

Fields:

tineInQueue: Time the airplane entered a queue

Methods:

public Airplane(int startTime)

The constructor will save the startTime into the field
Precondition: startTime is a non-negative integer.
Postcondition: An instance of the Airplane class is created with a start time.

Parameter: startTime is an integer representing the start time of the airplane.

public int getTimeInQueue()

Getter to retrieve the amount of time an object has spent in a queue.
Precondition: the airplane has entered a queue and the time in has been saved into the feld.
Postcondition: the timeInQueue has been returned.

public int setTimeInQueue (int startTime)

Setter to set the amount of time an object has spent in a queue.
Precondition: None.
Postcondition: The timeInQueue field of the Airplane object is set to the startTime value. The startTime parameter is passed as an argument.
Parameter: startTime is an integer representing the time in which the airplane entered the queue.

Runway Class:

The runway class will represent the runway in an airport. It will keep track of if the runway is occupied by an airplane; provide methods to assign an airplane to the runway, and release the runway, after the runway has finished its operation; and potentially handle runway maintenance and availability checks.

Fields:

occupied: false if the runway is not currently in use
currentPlaneTakingOff: true if runway is currently in use by an airplane taking off
currentPlaneLanding: true if runway is currently in use by a landing airplane
takeoffTime: length of time to takeoff (in minutes)
landingTime: length of time to land (in minutes)
startTimeOnRunway: beginning time for an airplanes runway time (minutes)

Methods:

public Runway(int tTime, int lTime)

Constructors for initializing the fields are like so:
takeoffTime = tTime
landingTime = lTime
isOccupied = false
inUseForTakeoff = false
inUseForLanding = false
start TimeOnRunway = -1
Preconditions:
tTime is a non-negative integer.
lTime is a non-negative integer.
Postconditions:
An instance of the Runway class is created with the specified takeoff time and landing time.
Parameter: tTime represents required take off time (in minutes).

Parameter: lTime represents required landing time (in minutes).

public boolean isRunwayOccupied(int currentTime)
Determines if the runway is occupied. If it is occupied, then it is not yet ready for the next plane. currentTime is the current time in the simulation.
Returns true if the runway is occupied.
Preconditions: currentTime is a non-negative integer representing the current time in minutes.
Postconditions: Returns a boolean value indicating whether the runway is occupied at the current time.
Parameter: currentTime represents current time in minutes (int).

public void setAvailable(boolean avail)
Setter for available field. avail is true if the runway is available for use
Preconditions: avail is a boolean value representing the availability status of the runway.
Postconditions: The availability status of the runway is updated based on the value of avail.
Parameter: avail is a boolean value to represent the availability status of the runway.

public void setCurrentPlaneTakingOff(boolean inUse)
Setter for the inUseForTakeoff field. inUse is true if the runway is in use by an airplane taking off.
Preconditions: inUse is a boolean value representing whether the runway is currently being used for a plane taking off (true) or not (false).
Postconditions: The state of the runway is updated to indicate whether it is currently in use for a plane taking off or not.
Parameter: inUse represents runway status.

public void setCurrentPlaneLanding(boolean inUse)
Setter to update the runway's status on if it is currently in use or not.
Precondition: None.
Postcondition: Based on inUse parameter, the runway status is updated. It indicates whether or not the runway is currently in use for landing.
Parameter: inUse is a boolean to represent the runway's status for landing. If inUse is true, the runway is currently in use for landing. If inUse is false, the runway is not currently in use for landing.

public int setStartTimeOnRunway(int time)
Setter method used to start the time for the runway.
Precondition: None.
Postcondition: The start time for the runway is updated to the time parameter.
Parameter: time is an int to represent the start time that is set for the runway.

Airplane
- timeInQueue: int
+ Airplane(startTime: int) + getTimeInQueue(): int + setTimeInQueue(startTime: int): int

Runway
- occupied: boolean - currentPlaneTakingOff: boolean - currentPlaneLanding: boolean - takeoffTime: int - landingTime: int - startTimeOnRunway: int
+ Runway(tTime: int, lTime: int) + isRunwayOccupied(currentTime: int): boolean + setAvailable(avail: boolean): void + setCurrentPlaneTakingOff(InUse: boolean): void + setCurrentPlaneLanding(inUse: boolean): void + setStartTimeOnRunway(time: int): int

Airport
- landingProbability: double - takeoffProbability: double - maxLandingQueueWait: int - runway: Runway - landingCount: int - takeoffCount: int - crashCount: int - timeInLandingQueue: int - timeInTakeoffQueue: int
+ main(args: String []): void <<static>> - inputData(): void - runSimulation(simulationLength: int): void - outputResults(): void