```
::::::::::::::
AirportException.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Airport Exception
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
public class AirportException extends RuntimeException {
    /**
     * Creates a new AirportException with a description of what happened.
     * @param s The description of what happened.
     */
    public AirportException(String s) {
        super(s);
    } // end constructor
}
::::::::::::::
AirportSystem.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: AirportSystem Class
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
public class AirportSystem
{
    private String name;
    private ListRAB<String> activeFlights;
    private ListRAB<Plane> waiting;
    private ListRAB<Runway> runways;
    private boolean allowLanding;
    private int takeoffs;
    private int landings;
    private int nextRunway;

    /**
     * Initializes the AirportSystem.
     * @param name name of the airport, used for checking if a flight will be land
ing.
     * @param enable boolean to see if landings will be enabled. Currently unused.
     */
    public AirportSystem(String name, boolean enable)
    {
        this.name = name;
        allowLanding = enable;
        activeFlights = new ListRAB();
        waiting = new ListRAB();
        runways = new ListRAB();
        takeoffs = 0;
        landings = 0;
        nextRunway = 0;
    }

    /**
     * Checks if the named runway exists in the AirportSystem.
     * @param runName name of the runway to find.
     * @return i integer of -1 to runways.size(), depending on result.
     */
    private int checkRunway(String runName)
    {
        int result = -1;
        for(int i = 0; i < runways.size() && result < 0; i++)
        {
            if(runways.get(i).getName().compareTo(runName) == 0)
            {
                result = i;
            }
        }
        return result;
    }

    /**
     * Checks if the named Runway exists in the AirportSystem.
     * @param runName name of the Runway to find.
     * @return boolean of if the Runway exists.
     */
    public boolean runwayValid(String runName)
    {
        return (checkRunway(runName) < 0) ? false : true;
    }

    /**
     * Removes the next Departure from a Runway of runName, and also removes it fr
om the log of activeFlight numbers.
     * @param runName name of the Runway to find.
     * @return p the next plane in the departure Queue.
     * @throws AirportException if something goes wrong while removing.
     */
    public Plane removeRunwayDeparture(String runName) throws AirportException
    {
        int check = checkRunway(runName);
        if(check >= 0)
        {
            if(runways.get(check).peekDepartures() != null)
            {
                Plane temp = runways.get(check).removeDeparture();
                activeFlights.remove(checkFlights(temp.getFlightNumber()));
                return temp;
            }
            else
            {
                return null;
            }
        }
        else
            return null;
    }

    /**
     * Removes the next Arrival from a Runway of runName, and also removes it from
 the log of activeFlight numbers.
     * @param runName name of the Runway to find.
     * @return p the next plane in the arrival Queue.
     * @throws AirportException if something goes wrong while removing.
     */
```

```
    public Plane removeRunwayArrival(String runName) throws AirportException
    {
        int check = checkRunway(runName);
        if(check >= 0)
        {
            Plane temp = runways.get(check).removeArrival();
            activeFlights.remove(checkFlights(temp.getFlightNumber()));
            return temp;
        }
        else
            return null;
    }

//    /**
//     * Returns the next plane in the waitlist, nondestructively.
//     * @param runName name of the Runway to find.
//     * @return p the next plane in the waiting list.
//     */
//    public Plane peekWaitingPlane(String runName) throws AirportException
//    {
//        int check = checkWaiting(runName);
//        if(check >= 0)
//        {
//            Plane temp = waiting.get(check);
//            return temp;
//        }
//        else
//            return null;
//    }

    /**
     * Returns if the waiting List is empty.
     * @return b the state of the waiting List.
     */
    public boolean waitIsEmpty()
    {
        return waiting.isEmpty();
    }

    /**
     * Returns the waitlist for direct modification (required by Driver.removeRunw
ay().)
     * @return l the List of all currently waiting Planes.
     */
    public ListRAB<Plane> getWaiting()
    {
        return waiting;
    }

    /**
     * Returns the index of the flightNumber in the tracking List, or -1.
     * @param flightName a String of the flightNumber being searched for.
     * @return i the index of the flightNumber, or -1.
     */
    private int checkFlights(String flightName)
    {
        int result = -1;
        for(int i = 0; i < activeFlights.size() && result < 0; i++)
        {
            if(activeFlights.get(i).compareTo(flightName) == 0)
            {
                result = i;
            }
        }
        return result;
    }

    /**
     * Returns the index of the flightNumber in the waiting List, or -1.
     * @param flightName a String of the flightNumber being searched for.
     * @return i the index of the flightNumber, or -1.
     */
    private int checkWaiting(String flightName)
    {
        int result = -1;
        for(int i = 0; i < waiting.size() && result < 0; i++)
        {
            if(waiting.get(i).getFlightNumber().compareTo(flightName) == 0)
            {
                result = i;
            }
        }
        return result;
    }

    /**
     * Returns whether the flightNumber is in the waiting List.
     * @param flightName a String of the flightNumber being searched for.
     * @return b a boolean of whether the flightName is present in the waiting Lis
t.
     */
    public boolean waitValid(String flightName)
    {
        return (checkWaiting(flightName) < 0) ? false : true;
    }

    /**
     * Adds a Runway of name runName.
     * @param runName the name of the new Runway to make.
     */
    public void addRunway(String runName)
    {
        int found = checkRunway(runName);
        if(found < 0)
        {
            runways.add(runways.size(),new Runway(runName));
        }
        else
        {
            throw new AirportException("Runway already exists.");
        }
    }

    /**
     * Removes a Runway of name runName.
     * @param runName the name of the Runway to remove.
     * @throws AirportException if the runway is not found.
     */
    public void removeRunway(String runName) throws AirportException
    {
        int found = checkRunway(runName);
        if(found >= 0)
        {
            runways.remove(found);
```

```
        }
        else
        {
            throw new AirportException("Runway not found.");
        }
    }

    /**
     * Returns the next Runway in the List, arranged circularly.
     * @return r the next Runway allowed to process a plane.
     */
    private Runway nextRunway()
    {
        Runway result = runways.get(nextRunway);
        nextRunway = ++nextRunway%runways.size();
        return result;
    }

    /**
     * Returns the specified Runway for direct modification (required by Driver.re
moveRunway().)
     * @param runName the name of the runway to find.
     * @return the Runway identified.
     */
    public Runway getRunway(String runName)
    {
        int found = checkRunway(runName);
        if(found >=0)
        {
            return runways.get(found);
        }
        else
        {
            throw new AirportException("Runway not found.");
        }
    }

    /**
     * Returns the next Runway allowed to operate, without incrementing the counte
r.
     * @param isTakeoff boolean selecting if we are searching for a takeoff or lan
ding operation.
     * @return the Runway identified.
     */
    private Runway peekNextActionableRunway(boolean isTakeoff)
    {
        Runway result = null;
        boolean empty = true;
        int count = nextRunway;
        while(empty == true && count < runways.size()+nextRunway)
        {
            result = runways.get(count);
            if(isTakeoff == true)
            {
                empty = result.noDepartures();
            }
            else
            {
                empty = result.noArrivals();
            }
            count++;
        }
        if(empty == false)
        {
            return result;
        }
        else
        {
            throw new AirportException("No available planes on any runway.");
        }
    }

    /**
     * Returns the next Runway allowed to operate, and increments the counter.
     * @param isTakeoff boolean selecting if we are searching for a takeoff or lan
ding operation.
     * @return the Runway identified.
     * @throws AirportException if there are no available planes on any runway.
     */
    private Runway nextActionableRunway(boolean isTakeoff) throws AirportException
    {
        Runway result = null;
        boolean empty = true;
        int count = 0;
        while(empty == true && count < runways.size())
        {
            result = nextRunway();
            if(isTakeoff == true)
            {
                empty = result.noDepartures();
            }
            else
            {
                empty = result.noArrivals();
            }
            count++;
        }
        if(empty == false)
        {
            return result;
        }
        else
        {
            throw new AirportException("No available planes on any runway.");
        }
    }

    /**
     * Returns the next Plane that will be processed, without incrementing the cou
nter.
     * @param isTakeoff boolean selecting if we are searching for a takeoff or lan
ding operation.
     * @return the Plane identified.
     * @throws AirportException if there are no planes on any runway.
     * @throws Exception if something goes wrong in the airport.
     */
    public Plane peekNextPlane(boolean isTakeoff) throws AirportException, Excepti
on
    {
        Plane temp = null;
        try {
            if(isTakeoff == true)
            {
```

```
                temp =  peekNextActionableRunway(isTakeoff).peekDepartures();
            }
            else
            {
                temp =  peekNextActionableRunway(isTakeoff).peekArrivals();
            }
        }
        catch(Exception e)
        {
        }
        if(temp != null)
        {
            return temp;
        }
        else
        {
            throw new AirportException("No plane on any runway.");
        }
    }

    /**
     * Returns the next Plane that will be processed, and increments the counter.
     * @param isTakeoff boolean selecting if we are searching for a takeoff or lan
ding operation.
     * @return the Plane identified.
     * @throws AirportException if something goes wrong while removing.
     */
    public Plane getNextPlane(boolean isTakeoff) throws AirportException
    {
        Runway result = nextActionableRunway(isTakeoff);
        //get next runway that has a plane in the appropriate queue
        if(isTakeoff == true)
        {
            return result.removeDeparture();
        }
        else
        {
            return result.removeArrival();
        }
    }

    /**
     * Moves the designated flightNum from the waiting List to its Runway.
     * @param flightNum the name of the flight to add.
     * @throws AirportException if the specified plane is not within the waiting l
ist.
     * @throws Exception if something goes wrong within the airport.
     */
    public void reenter(String flightNum) throws AirportException, Exception
    {
        Plane temp = null;
        int num = waiting.size();
        int index = -1;
        for(int i = 0; i< num && index < 0; i++)
        {
            if(waiting.get(i).getFlightNumber().compareTo(flightNum) == 0)
            {
                index = i;
                temp = waiting.get(i);
            }
        }
        if(index >= 0)
        {
            waiting.remove(index);
            activeFlights.remove(checkFlights(flightNum));
            addPlane(temp);
        }
        else
        {
            throw new AirportException("Plane specified not found in wait list.");
        }
    }

    /**
     * Adds the designated Plane to its Runway.
     * @param airplane the Plane to add.
     * @throws AirportException if the runway isn't found or if there is an identi
cal plane.
     * @throws Exception if something goes wrong within the airport.
     */
    public void addPlane(Plane airplane) throws AirportException, Exception
    {
        String fn = airplane.getFlightNumber();
        int check = checkFlights(fn);
        if(check < 0)
        {
            int run = checkRunway(airplane.getRunway());
            if(run >= 0)
            {
                activeFlights.add(activeFlights.size(),fn);
                Runway temp = runways.get(run);
                if(airplane.getDestination().compareTo(this.name) == 0)
                {
                    temp.addArrival(airplane);
                }
                else
                {
                    temp.addDeparture(airplane);
                }
            }
            else
            {
                throw new AirportException("Runway not found.");
            }
        }
        else
        {
            throw new AirportException("Plane with identical flight number already
 in system");
        }
    }

    /**
     * Removes the designated Plane from its Runway, either to delete or to add to
 the waitlist.
     * @param isTakeoff is this a departure or arrival.
     * @param allow will it complete the action or move to the waitlist.
     * @throws AirportException if no planes are able to proceed.
     */
    public void processPlane(boolean isTakeoff, boolean allow) throws AirportExcep
tion
    {
        Plane temp = getNextPlane(isTakeoff);
        if(temp != null)
```

```
        {
            if(allow == false)
            {
                waiting.add(waiting.size(),temp);
            }
            else
            {
                activeFlights.remove(checkFlights(temp.getFlightNumber()));
                if(isTakeoff == true)
                {
                    takeoffs++;
                }
                else
                {
                    landings++;
                }
            }
        }
        else
        {
            throw new AirportException("No planes on any runways able to proceed."
);
        }
    }

    /**
     * Returns the airport name.
     * @return the String of the airport name
     */
    public String getName()
    {
        return name;
    }

    /**
     * Returns the number of takeoffs.
     * @return the int of total takeoffs
     */
    public int getTakeoffs()
    {
        return takeoffs;
    }

    /**
     * Returns the number of landings.
     * @return the int of total landings
     */
    public int getLandings()
    {
        return landings;
    }

    /**
     * Returns info about the Planes trying to take off.
     * @return the String of each Plane trying to take off, by Runway
     */
    public String displayTakeoff()
    {
        StringBuilder sb = new StringBuilder();
        int num = runways.size();
        for(int i = 0; i<num; i++)
        {
            sb.append(runways.get(i).listDepartures() + "\n\n");
        }
        sb.delete(sb.length()-2,sb.length());
        return sb.toString();
    }

    /**
     * Returns info about the Planes trying to land.
     * @return the String of each Plane trying to land, by Runway
     */
    public String displayLanding()
    {
        StringBuilder sb = new StringBuilder();
        int num = runways.size();
        for(int i = 0; i<num; i++)
        {
            sb.append(runways.get(i).listArrivals() + "\n\n");
        }
        sb.delete(sb.length()-2,sb.length());
        return sb.toString();
    }

    /**
     * Returns info about the Planes waiting to be sent back to a Runway.
     * @return the String of each Plane in the waiting List
     */
    public String displayWaiting()
    {
        StringBuilder sb = new StringBuilder();
        int num = waiting.size();
        if(num == 0)
        {
            return "No flights are waiting for clearance.";
        }
        else
        {
            sb.append("These flights are waiting for clearance:\n");
            for(int i = 0; i<num; i++)
            {
                sb.append(waiting.get(i) + "\n");
            }
            sb.delete(sb.length()-1,sb.length());
            return sb.toString();
        }
    }
}
```

::::::::::::::
Driver1.java
::::::::::::::

```
/*
 * Purpose: Data Structure and Algorithms Project: Driver Class
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
import java.io.*;

public class Driver1
{
```

```
    static private AirportSystem airport;
    static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.
in));

    /**
     * Operates the AirportSystem Class.
     */
    public static void main(String[] args)
    {
        try {
            System.out.println("Initializing Airport");
            System.out.print("Enter Airport name: ");
            String name = stdin.readLine();
            System.out.println(name);
            airport = new AirportSystem(name,false);
            System.out.print("Enter number of runways: ");
            int numRun = Integer.parseInt(stdin.readLine());
            System.out.println(numRun);
            for(int i = 0; i < numRun; i++)
            {
                try {
                    System.out.print("Enter name for runway #" + (i+1) + ": ");
                    String runName = stdin.readLine();
                    System.out.println(runName);
                    airport.addRunway(runName);
                }
                catch(Exception e)
                {
                    System.out.println(e.getMessage());
                }
            }
            System.out.println("Select from the following menu:\n\t0. Exit program
.\n\t1. Plane enters the system.\n\t2. Plane attempts takes off.\n\t3. Plane is al
lowed to re-enter a runway.\n\t4. Runway opens.\n\t5. Runway closes.\n\t6. Display
 info about planes waiting to take off.\n\t7. Display info about planes waiting to
 be allowed to re-enter a runway.\n\t8. Display number of planes who have taken of
f.\n\t9. Plane attemps landing.\n\t10. Display info about planes waiting to land.\
n\t11. Display number of planes who have landed.");
            boolean contin = true;
            int selection;
            while(contin == true)
            {
                try {
                    System.out.print("Make your selection now: ");
                    selection = Integer.parseInt(stdin.readLine());
                    System.out.println(selection);
                    switch (selection)
                    {
                    case 0:
                        System.out.println("Goodbye.");
                        contin = false;
                        break;
                    case 1:
                        addPlane();
                        break;
                    case 2:
                        takeoff();
                        break;
                    case 3:
                        reenter();
                        break;
                    case 4:
                        addRunway();
                        break;
                    case 5:
                        closeRunway();
                        break;
                    case 6:
                        displayTakeoff();
                        break;
                    case 7:
                        displayWaiting();
                        break;
                    case 8:
                        numTakeoff();
                        break;
                    case 9:
                        landing();
                        break;
                    case 10:
                        displayLanding();
                        break;
                    case 11:
                        numLanding();
                        break;
                    }
                    System.out.println();
                }
                catch(Exception e)
                {
                    System.out.println(e.getMessage());
                }
            }
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }

    /**
     * Adds a plane dynamically to the AirportSystem.
     */
    static public void addPlane() throws AirportException, Exception
    {
        System.out.print("Enter flight number: ");
        String fn = stdin.readLine();
        System.out.println(fn);
        System.out.print("Enter destination: ");
        String d = stdin.readLine();
        System.out.println(d);
        String r;
        boolean val = true;
        do
        {
            System.out.print("Enter runway: ");
            r = stdin.readLine();
            System.out.println(r);
            val = airport.runwayValid(r);
            if(val != true)
            {
                System.out.println("No such runway.");
            }
        } while(val != true);
```

```
        airport.addPlane(new Plane(fn,d,r));
        System.out.println(airport.getName() + " " + airport.getName().compareTo(d
));
        System.out.println("Flight " + fn + " is now waiting for clearance on runw
ay " + r + ".");

    }

    /**
     * processes a plane for takeoff in the AirportSystem.
     */
    static public void takeoff() throws AirportException, QueueException, Exceptio
n
    {
        Plane temp = airport.peekNextPlane(true);
        System.out.print("Is " + temp + " clear for takeoff(Y/N): ");
        String input = stdin.readLine().toUpperCase();
        System.out.println(input);
        boolean allow = (input.compareTo("Y") == 0 ) ? true : false;
        airport.processPlane(true,allow);
        if(allow ==true)
        {
            System.out.println(temp.getFlightNumber() + " has taken off from runwa
y " + temp.getRunway() + ".");
        }
        else
        {
            System.out.println(temp.getFlightNumber() + " has been denied clearanc
e.");
        }
    }

    /**
     * processes a plane for landing in the AirportSystem.
     */
    static public void landing() throws AirportException, Exception
    {
        Plane temp = airport.peekNextPlane(true);
        System.out.print("Is " + temp + " clear for landing(Y/N): ");
        String input = stdin.readLine().toUpperCase();
        System.out.println(input);
        boolean allow = (input.compareTo("Y") == 0 ) ? true : false;
        airport.processPlane(false,allow);
        if(allow ==true)
        {
            System.out.println(temp.getFlightNumber() + " has landed on runway " +
 temp.getRunway() + ".");
        }
        else
        {
            System.out.println(temp.getFlightNumber() + " has been denied clearanc
e.");
        }
    }

    /**
     * re-adds a plane to it's Runway and departure/arrival Queue in AirportSystem
.
     */
    static public void reenter() throws AirportException, Exception
    {
        if(airport.waitIsEmpty() != true)
        {
            String toAdd;
            boolean val;
            do
            {
                System.out.print("Enter the the flight number: ");
                toAdd = stdin.readLine();
                System.out.println(toAdd);
                val = airport.waitValid(toAdd);
                if(val == false)
                {
                    System.out.println(toAdd + " is not waiting for clearance.");
                }
            }
            while(val == false);
            airport.reenter(toAdd);
        }
        else
        {
            throw new AirportException("No planes in wait list.");
        }

    }

    /**
     * Adds a Runway to the AirportSystem.
     */
    static public void addRunway() throws AirportException, Exception
    {
        String toAdd;
        boolean val = true;
        do
        {
            System.out.print("Enter the name of the runway to open: ");
            toAdd = stdin.readLine();
            System.out.println(toAdd);
            val = airport.runwayValid(toAdd);
            if(val == true)
            {
                System.out.println("Runway already exists.");
            }
        } while(val == true);
        airport.addRunway(toAdd);
        System.out.println(toAdd + " was added.");

    }

    /**
     * Closes a Runway in the AirportSystem, moving assigned Planes to other Runwa
ys.
     */
    static public void closeRunway() throws AirportException, Exception
    {
        String toRem;
        boolean val = true;
        do
        {
            System.out.print("Enter the name of the runway to close: ");
            toRem = stdin.readLine();
            System.out.println(toRem);
            val = airport.runwayValid(toRem);
            if(val == false)
```

```
            {
                System.out.println("Runway does not exist.");
            }
        }
        while(val == false);
        System.out.println(toRem);

        Runway run = airport.getRunway(toRem);
        Plane temp = null;
        while(run.noDepartures() == false)
        {
            temp = airport.removeRunwayDeparture(toRem);
            String newRun = null;
            int comp = 0;
            boolean exists = false;
            do
            {
                System.out.print("Enter new runway for " + temp.getFlightNumber()
+ ": ");
                newRun = stdin.readLine();
                System.out.println(newRun);
                exists = airport.runwayValid(newRun);
                comp = newRun.compareTo(toRem);
                if(comp == 0)
                {
                    System.out.println("This is the runway that will close.");
                }
                if(exists == false)
                {
                    System.out.println("No such runway.");
                }
            } while(comp == 0 || exists == false);
            temp.setRunway(newRun);
            airport.addPlane(temp);
            System.out.println(temp.getFlightNumber() + " is now assigned to runwa
y " + newRun + ".");
        }
        while(run.noArrivals() == false)
        {
            temp = airport.removeRunwayDeparture(toRem);
            String newRun = null;
            int comp = 0;
            boolean exists = false;
            do
            {
                System.out.print("Enter new runway for " + temp.getFlightNumber()
+ ": ");
                newRun = stdin.readLine();
                System.out.println(newRun);
                exists = airport.runwayValid(newRun);
                comp = newRun.compareTo(toRem);
                if(comp == 0)
                {
                    System.out.println("This is the runway that will close.");
                }
                if(exists == false)
                {
                    System.out.println("No such runway.");
                }
            } while(comp == 0 || exists == false);
            temp.setRunway(newRun);
            airport.addPlane(temp);
            System.out.println(temp.getFlightNumber() + " is now assigned to runwa
y " + newRun + ".");
        }

        ListRAB<Plane> wait = airport.getWaiting();
        for(int i = 0; i<wait.size(); i++)
        {
            temp = wait.get(i);
            if(temp.getRunway().compareTo(toRem) == 0)
            {
                String newRun = null;
                int comp = 0;
                boolean exists = false;
                do
                {
                    System.out.print("Enter new runway for " + temp.getFlightNumbe
r() + ": ");
                    newRun = stdin.readLine();
                    System.out.println(newRun);
                    exists = airport.runwayValid(newRun);
                    comp = newRun.compareTo(toRem);
                    if(comp == 0)
                    {
                        System.out.println("This is the runway that will close.");
                    }
                    if(exists == false)
                    {
                        System.out.println("No such runway.");
                    }
                } while(comp == 0 || exists == false);
                temp.setRunway(newRun);
                System.out.println(temp.getFlightNumber() + " is now assigned to r
unway " + newRun);
            }
        }

        airport.removeRunway(toRem);
        System.out.println(toRem + " has been closed.");

    }

    /**
     * Prints info about the Planes trying to takeoff.
     */
    static public void displayTakeoff()
    {
        System.out.println(airport.displayTakeoff());
    }

    /**
     * Prints info about the Planes trying to land.
     */
    static public void displayLanding()
    {
        System.out.println(airport.displayLanding());
    }

    /**
     * Prints info about the planes currently waiting to be added to a Runway.
     */
    static public void displayWaiting()
    {
```

```
        System.out.println(airport.displayWaiting());
    }

    /**
     * Prints the number of takeoffs.
     */
    static public void numTakeoff()
    {
        System.out.println(airport.getTakeoffs() + " takeoffs have occurred.");
    }

    /**
     * Prints the number of landings.
     */
    static public void numLanding()
    {
        System.out.println(airport.getLandings() + " landings have occurred.");
    }
}
:::::::::::::::
Driver.java
:::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Driver Class
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
import java.io.*;

public class Driver
{
    static private AirportSystem airport;
    static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.
in));

    /**
     * Operates the AirportSystem Class.
     */
    public static void main(String[] args)
    {
        try {
            System.out.println("Initializing Airport");
            airport = new AirportSystem("",false);
            System.out.print("Enter number of runways: ");
            int numRun = Integer.parseInt(stdin.readLine());
            System.out.println(numRun);
            for(int i = 0; i < numRun; i++)
            {
                try {
                    System.out.print("Enter name for runway #" + (i+1) + ": ");
                    String runName = stdin.readLine();
                    System.out.println(runName);
                    airport.addRunway(runName);
                }
                catch(Exception e)
                {
                    System.out.println(e.getMessage());
                }
            }
            System.out.println("Select from the following menu:\n\t0. Exit program
.\n\t1. Plane enters the system.\n\t2. Plane attempts takes off.\n\t3. Plane is al
lowed to re-enter a runway.\n\t4. Runway opens.\n\t5. Runway closes.\n\t6. Display
 info about planes waiting to take off.\n\t7. Display info about planes waiting to
 be allowed to re-enter a runway.\n\t8. Display number of planes who have taken of
f.\n\t9. Plane attemps landing.\n\t10. Display info about planes waiting to land.\
n\t11. Display number of planes who have landed.");
            boolean contin = true;
            int selection;
            while(contin == true)
            {
                try {
                    System.out.print("Make your selection now: ");
                    selection = Integer.parseInt(stdin.readLine());
                    System.out.println(selection);
                    switch (selection)
                    {
                    case 0:
                        System.out.println("Goodbye.");
                        contin = false;
                        break;
                    case 1:
                        addPlane();
                        break;
                    case 2:
                        takeoff();
                        break;
                    case 3:
                        reenter();
                        break;
                    case 4:
                        addRunway();
                        break;
                    case 5:
                        closeRunway();
                        break;
                    case 6:
                        displayTakeoff();
                        break;
                    case 7:
                        displayWaiting();
                        break;
                    case 8:
                        numTakeoff();
                        break;
                    }
                    System.out.println();
                }
                catch(Exception e)
                {
                    System.out.println(e.getMessage());
                }
            }
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }

    /**
```

```
     * Adds a plane dynamically to the AirportSystem.
     */
    static public void addPlane() throws AirportException, Exception
    {
        System.out.print("Enter flight number: ");
        String fn = stdin.readLine();
        System.out.println(fn);
        System.out.print("Enter destination: ");
        String d = stdin.readLine();
        System.out.println(d);
        String r;
        boolean val = true;
        do
        {
            System.out.print("Enter runway: ");
            r = stdin.readLine();
            System.out.println(r);
            val = airport.runwayValid(r);
            if(val != true)
            {
                System.out.println("No such runway.");
            }
        } while(val != true);
        airport.addPlane(new Plane(fn,d,r));
        System.out.println("Flight " + fn + " is now waiting for takeoff on runway
 " + r + ".");

    }

    /**
     * processes a plane for takeoff in the AirportSystem.
     */
    static public void takeoff() throws AirportException, QueueException, Exceptio
n
    {
        Plane temp = airport.peekNextPlane(true);
        System.out.print("Is " + temp + " clear for takeoff(Y/N): ");
        String input = stdin.readLine().toUpperCase();
        System.out.println(input);
        boolean allow = (input.compareTo("Y") == 0 ) ? true : false;
        airport.processPlane(true,allow);
        if(allow ==true)
        {
            System.out.println(temp.getFlightNumber() + " has taken off from runwa
y " + temp.getRunway() + ".");
        }
        else
        {
            System.out.println(temp.getFlightNumber() + " has been denied clearanc
e.");
        }
    }

    /**
     * processes a plane for landing in the AirportSystem.
     */
    static public void landing() throws AirportException, Exception
    {
        Plane temp = airport.peekNextPlane(true);
        System.out.print("Is " + temp + " clear for landing(Y/N): ");
        String input = stdin.readLine().toUpperCase();
        System.out.println(input);
        boolean allow = (input.compareTo("Y") == 0 ) ? true : false;
        airport.processPlane(true,allow);
        if(allow ==true)
        {
            System.out.println(temp.getFlightNumber() + " has taken off from runwa
y " + temp.getRunway() + ".");
        }
        else
        {
            System.out.println(temp.getFlightNumber() + " has been denied clearanc
e.");
        }
    }

    /**
     * re-adds a plane to it's Runway and departure/arrival Queue in AirportSystem
.
     */
    static public void reenter() throws AirportException, Exception
    {
        if(airport.waitIsEmpty() != true)
        {
            String toAdd;
            boolean val;
            do
            {
                System.out.print("Enter the the flight number: ");
                toAdd = stdin.readLine();
                System.out.println(toAdd);
                val = airport.waitValid(toAdd);
                if(val == false)
                {
                    System.out.println(toAdd + " is not waiting for clearance.");
                }
            }
            while(val == false);
            airport.reenter(toAdd);
        }
        else
        {
            throw new AirportException("No planes in wait list.");
        }

    }

    /**
     * Adds a Runway to the AirportSystem.
     */
    static public void addRunway() throws AirportException, Exception
    {
        String toAdd;
        boolean val = true;
        do
        {
            System.out.print("Enter the name of the runway to open: ");
            toAdd = stdin.readLine();
            System.out.println(toAdd);
            val = airport.runwayValid(toAdd);
            if(val == true)
            {
                System.out.println("Runway already exists.");
            }
```

```
        } while(val == true);
        airport.addRunway(toAdd);
        System.out.println(toAdd + " was added.");

    }

    /**
     * Closes a Runway in the AirportSystem, moving assigned Planes to other Runwa
ys.
     */
    static public void closeRunway() throws AirportException, Exception
    {
        String toRem;
        boolean val = true;
        do
        {
            System.out.print("Enter the name of the runway to close: ");
            toRem = stdin.readLine();
            System.out.println(toRem);
            val = airport.runwayValid(toRem);
            if(val == false)
            {
                System.out.println("Runway does not exist.");
            }
        }
        while(val == false);
        System.out.println(toRem);

        Runway run = airport.getRunway(toRem);
        Plane temp = null;
        while(run.noDepartures() == false)
        {
            temp = airport.removeRunwayDeparture(toRem);
            String newRun = null;
            int comp = 0;
            boolean exists = false;
            do
            {
                System.out.print("Enter new runway for " + temp.getFlightNumber()
+ ": ");

                newRun = stdin.readLine();
                System.out.println(newRun);
                exists = airport.runwayValid(newRun);
                comp = newRun.compareTo(toRem);
                if(comp == 0)
                {
                    System.out.println("This is the runway that will close.");
                }
                if(exists == false)
                {
                    System.out.println("No such runway.");
                }
            } while(comp == 0 || exists == false);
            temp.setRunway(newRun);
            airport.addPlane(temp);
            System.out.println(temp.getFlightNumber() + " is now assigned to runwa
y " + newRun + ".");
        }
        while(run.noArrivals() == false)
        {
            temp = airport.removeRunwayDeparture(toRem);
            String newRun = null;
            int comp = 0;
            boolean exists = false;
            do
            {
                System.out.print("Enter new runway for " + temp.getFlightNumber()
+ ": ");
                newRun = stdin.readLine();
                System.out.println(newRun);
                exists = airport.runwayValid(newRun);
                comp = newRun.compareTo(toRem);
                if(comp == 0)
                {
                    System.out.println("This is the runway that will close.");
                }
                if(exists == false)
                {
                    System.out.println("No such runway.");
                }
            } while(comp == 0 || exists == false);
            temp.setRunway(newRun);
            airport.addPlane(temp);
            System.out.println(temp.getFlightNumber() + " is now assigned to runwa
y " + newRun + ".");
        }

        ListRAB<Plane> wait = airport.getWaiting();
        for(int i = 0; i<wait.size(); i++)
        {
            temp = wait.get(i);
            if(temp.getRunway().compareTo(toRem) == 0)
            {
                String newRun = null;
                int comp = 0;
                boolean exists = false;
                do
                {
                    System.out.print("Enter new runway for " + temp.getFlightNumbe
r() + ": ");
                    newRun = stdin.readLine();
                    System.out.println(newRun);
                    exists = airport.runwayValid(newRun);
                    comp = newRun.compareTo(toRem);
                    if(comp == 0)
                    {
                        System.out.println("This is the runway that will close.");
                    }
                    if(exists == false)
                    {
                        System.out.println("No such runway.");
                    }
                } while(comp == 0 || exists == false);
                temp.setRunway(newRun);
                System.out.println(temp.getFlightNumber() + " is now assigned to r
unway " + newRun);
            }
        }

        airport.removeRunway(toRem);
        System.out.println(toRem + " has been closed.");

    }
```

```
    /**
     * Prints info about the Planes trying to takeoff.
     */
    static public void displayTakeoff()
    {
        System.out.println(airport.displayTakeoff());
    }

    /**
     * Prints info about the Planes trying to land.
     */
    static public void displayLanding()
    {
        System.out.println(airport.displayLanding());
    }

    /**
     * Prints info about the planes currently waiting to be added to a Runway.
     */
    static public void displayWaiting()
    {
        System.out.println(airport.displayWaiting());
    }

    /**
     * Prints the number of takeoffs.
     */
    static public void numTakeoff()
    {
        System.out.println(airport.getTakeoffs() + " takeoffs have occurred.");
    }

    /**
     * Prints the number of landings.
     */
    static public void numLanding()
    {
        System.out.println(airport.getLandings() + " landings have occurred.");
    }
}
:::::::::::::::
ListAB.java
:::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: ListAB Class
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */


// ******************************************************
// Array-based implementation of the ADT list.
// ******************************************************
public class ListAB<E> implements ListInterface<E>
{

    private static final int MAX_LIST = 3;
    protected Object[] items;  // an array of list items
    protected int numItems;  // number of items in list

    public ListAB()
    {
        items = (E[]) new Object[MAX_LIST];
        numItems = 0;
    }  // end default constructor

    /**
     * Returns if the List is empty.
     * @return boolean The state of the list.
     */
    public boolean isEmpty()
    {
        return (numItems == 0);
    } // end isEmpty

    /**
     * Returns the current size of the list.
     * @return s The size of the list.
     */
    public int size()
    {
        return numItems;
    }  // end size

    /**
     * Clears the List.
     */
    public void removeAll()
    {
        // Creates a new array; marks old array for
        // garbage collection.
        items = (E[]) new Object[MAX_LIST];
        numItems = 0;
    } // end removeAll

    /**
     * Adds an item to the list.
     * @param index The index to add the new item too
     * @param item The item to add to this location
     */
    public void add(int index, E item)
    throws  ListIndexOutOfBoundsException
    {
        if (numItems==items.length) //fixes implementation errors //fixes programm
ing style
        {
            throw new ListException("ListException on add");
        }  // end if
        if (index >= 0 && index <= numItems)
        {
            // make room for new element by shifting all items at
            // positions >= index toward the end of the
            // list (no shift if index == numItems+1)
            for (int pos = numItems-1; pos >= index; pos--)  //textbook code modif
ied to eliminate logic error causing ArrayIndexOutOfBoundsException
            {
                items[pos+1] = items[pos];
            } // end for
            // insert new item
            items[index] = item;
```

```
            numItems++;
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on add");
        }  // end if
    } //end add

    /**
     * Returns an item from the list.
     * @param index the index to retrieve an item from.
     * @return item the item in the given index.
     */
    @SuppressWarnings("unchecked")
    public E get(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            return (E) items[index];
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on get");
        }  // end if
    } // end get

    /**
     * Removes an item from the list.
     * @param index the index to remove.
     */
    public void remove(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            // delete item by shifting all items at
            // positions > index toward the beginning of the list
            // (no shift if index == size)
            for (int pos = index+1; pos < numItems; pos++) //textbook code modifie
d to eliminate logic error causing ArrayIndexOutOfBoundsException

            {
                items[pos-1] = items[pos];
            }  // end for
            numItems--;
            items[numItems] = null; //fixes memory leak
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on remove");
        }  // end if
    } //end remove
}

::::::::::::::
```

```
ListException.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: ListException
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
public class ListException extends RuntimeException {
    public ListException(String s) {
        super(s);   // end ListException
    } // end constructor
}
::::::::::::::
ListIndexOutOfBoundsException.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project:  ListIndexOutOfBoundsException
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
public class ListIndexOutOfBoundsException extends IndexOutOfBoundsException
{
    /**
     * Creates a new ListIndexOutOfBoundsException with a description of what happ
ened.
     * @param s The description of what happened.
     */
    public ListIndexOutOfBoundsException(String s)
    {
        super(s);
    } // end constructor
} // end ListIndexOutOfBoundsException
::::::::::::::
ListInterface.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: List Interface
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
public interface ListInterface<E>
{
    /**
     * Returns if the List is empty.
     * @return boolean The state of the list.
     */
    boolean isEmpty();
    /**
     * Returns the current size of the list.
     * @return s The size of the list.
```

```
     */
    int size();

    /**
     * Adds an item from the list.
     * @param index The index to add the new item too
     * @param item The item to add to this location
     */
    void add(int index, E item)
    throws ListIndexOutOfBoundsException;

    /**
     * Returns an item from the list.
     * @param index the index to retrieve an item from.
     * @return item the item in the given index.
     */
    E get(int index)
    throws ListIndexOutOfBoundsException;

    /**
     * Removes an item from the list.
     * @param index the index to remove.
     */
    void remove(int index)
    throws ListIndexOutOfBoundsException;

    /**
     * Clears the List.
     */
    void removeAll();
}  // end ListInterface
::::::::::::::
ListRAB.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: ListRAB Class
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Nicholas Bovee
 * @version: 2019.12.03
 */
import java.util.*;

public class ListRAB<E> extends ListAB<E> implements ListInterface<E>
{
//      private int assignments = 0;

    public ListRAB()
    {
        super();
    }

    /**
     * Adds an item to the list.
     * @param index The index to add the new item too
     * @param item The item to add to this location
     */
    public void add(int index, E item) throws ListIndexOutOfBoundsException //revised add
    {
        if (index >= 0 && index <= numItems)
        {
            if (numItems==items.length) //fixes implementation errors //fixes programming style
            {
                int newSize =(int)(items.length * 3 / 2);
                Object[] newArray = (E[]) new Object[newSize];

                for(int i = 0, j = 0; j < items.length; i++, j++)
                {
                    if(i == index)
                    {
                        j--;
                    }
                    else
                    {
                        newArray[i] = items[j];

                    }
                }
                newArray[index] = item;
                items = newArray;
            }
            else {
                // make room for new element by shifting all items at
                // positions >= index toward the end of the
                // list (no shift if index == numItems+1)
                for (int pos = numItems-1; pos >= index; pos--)  //textbook code modified to eliminate logic error causing ArrayIndexOutOfBoundsException
                {
                    items[pos+1] = items[pos];
                } // end for
                // insert new item
                items[index] = item;
            }
            numItems++;
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on add");
        }  // end if
    } //end add

    /**
     * Returns a String representation of the List.
     * @return s the String representation of the list.
     */
    public String toString()
    {
        StringBuilder builder = new StringBuilder();
        for(int i = 0; i<numItems; i++)
        {
            builder.append(items[i] + " ");
        }
        return builder.toString();
    }

    /**
     * Reverses the order of the list.
     */
```

```
    public void reverse()
    {
        //below is the most efficient reverse method tested. See conclusions.
        reverseMemDirect();
    }

    /**
     * Specific implementation of List reversal.
     */
    private void reverseMemDirect()
    {
        Object[] newItems = (E[]) new Object[numItems];
        for(int i = 0; i<numItems; i++)
        {
            newItems[i] = items[numItems-i-1];
            //assignments++;//1 values
        }
        items = newItems;
    }

    /**
     * Resizes the list to accommodate more items. Does not currently reduce size.
     */
    private void resize()
    {
        int newSize =(int)(items.length * 3 / 2);
        Object[] newArray = new Object[newSize];
        for(int i = 0; i < items.length; i++)
        {
            newArray[i] = items[i];
        }
        items = newArray;

    }
}

:::::::::::::::
Node.java
:::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Node class, as part of Lists.
 * Status: Complete and thoroughly tested
 * Last update: 11/24/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.11.24
 */

public class Node<T>
{
    private T item;
    private Node<T> next;

    /**
     * Creates a new Node with an item, and no Node ahead.
     * @param newItem The Node's Item.
     */
    public Node(T newItem)
    {
        item = newItem;
        next = null;
    } // end constructor

    /**
     * Creates a new Node with an item and another Node ahead of this Node.
     * @param newItem The Node's Item.
     * @param nextNode The Node after this Node.
     */
    public Node(T newItem, Node<T> nextNode)
    {
        item = newItem;
        next = nextNode;
    } // end constructor

    /**
     * Sets a new Item for this Node.
     * @param newItem The new item.
     */
    public void setItem(T newItem)
    {
        item = newItem;
    } // end setItem

    /**
     * Returns this Node's Item.
     * @return The item.
     */
    public T getItem()
    {
        return item;
    } // end getItem

    /**
     * Sets a Node to be the one ahead of this Node.
     * @param nextNode The Node ahead of this Node.
     */
    public void setNext(Node<T> nextNode)
    {
        next = nextNode;
    } // end setNext

    /**
     * Returns the Node ahead of this Node.
     * @return The Node ahead of this Node.
     */
    public Node<T> getNext()
    {
        return next;
    } // end getNext
} // end class Node
:::::::::::::::
Plane.java
:::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Plane that takes off of / lands
 on runways.
 * Status: Complete and thoroughly tested
 * Last update: 12/03/19
 * Submitted: 12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.12.03
 */
```

```
public class Plane
{
    private String flightNumber;
    private String destination;
    private String runway;

    /**
     * Constructs a Plane object with its flight number, destination, and the runw
ay it should use.
     * @param fn The flight number.
     * @param d The destination.
     * @param r The runway.
     */
    public Plane(String fn, String d, String r)
    {
        flightNumber = fn;
        destination = d;
        runway = r;
    }

    /**
     * Returns the flight number of the plane.
     * @return The plane's flight number.
     */
    public String getFlightNumber()
    {
        return flightNumber;
    }

    /**
     * Sets a new flight number for the plane.
     * @param fn The new flight number.
     */
    public void setFlightNumber(String fn)
    {
        flightNumber = fn;
    }

    /**
     * Returns the plane's destination.
     * @return The plane's destination.
     */
    public String getDestination()
    {
        return destination;
    }

    /**
     * Sets a new destination for the plane.
     * @param d The new destination.
     */
    public void setDestination(String d)
    {
        destination = d;
    }

    /**
     * Returns the plane's runway to be used.
     * @return The plane's runway.
     */
    public String getRunway()
    {
        return runway;
    }

    /**
     * Sets a new runway for the plane.
     * @param r The new runway.
     */
    public void setRunway(String r)
    {
        runway = r;
    }

    /**
     * Returns an overall description of the plane as a String.
     * @return The plane description.
     */
    public String toString()
    {
        StringBuilder str = new StringBuilder("Flight " + flightNumber + " to " +
destination + ".");
        return str.toString();
    }

}
::::::::::::::
QueueCSLS.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: CSLS-based Queue built on the Q
ueueInterface.
 * Status: Complete and thoroughly tested
 * Last update: 11/24/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.11.24
 */

public class QueueCSLS<T> implements QueueInterface<T> {

    private Node<T> tail;

    /**
     * Returns whether the Queue is empty or not.
     * @return True if the queue is empty; False otherwise.
     */
    public boolean isEmpty()
    {
        return tail == null;
    }

    /**
     * Enqueues an Item to the back of the Queue and updates the tail.
     * @param item The item to be enqueued.
     */
    public void enqueue(T item) // appends the passed item to the end of the colle
ction
    {
        if(tail == null)
        {
            tail = new Node<T>(item);
```

```
            tail.setNext(tail);
        }
        else
        {
            tail.setNext(new Node<T>(item, tail.getNext()));
            tail = tail.getNext();
        }
    }

    /**
     * Dequeues the item at the front of the Queue.
     * @throws QueueException when attempting to dequeue from an empty queue.
     * @return The dequeued item.
     */
    public T dequeue() throws QueueException
    {
        T dequeued = null;

        if(tail == null)
        {
            throw new QueueException("Queue exception at dequeue");
        }
        else if(tail.getNext() == tail)
        {
            dequeued = tail.getItem();
            tail = null;
        }
        else
        {
            Node<T> front = tail.getNext();
            dequeued = front.getItem();
            tail.setNext(front.getNext());
        }

        return dequeued;
    }

    /**
     * Dequeues all items in the queue.
     */
    public void dequeueAll()
    {
        tail = null;
    }

    /**
     * Returns whatever item is at the front of the Queue.
     * @throws QueueException when attempting to peek at an empty queue.
     * @return The item at the front.
     */
    public T peek() throws QueueException
    {
        if(tail == null)
        {
            throw new QueueException("Queue exception at peek");
        }

        return tail.getNext().getItem();
    }

    /**
     * Returns a String listing of the items in the Queue (in order).
     * @return The listing.
     */
    public String toString() //collects and returns a String representation of the
 collection in order from first to last
    {
        Node curr = tail;
        StringBuilder str = new StringBuilder();

        if(curr != null)
        {
            curr = curr.getNext();
            str.append(curr.getItem());
            while(curr != tail)
            {
                curr = curr.getNext();
                str.append("\n");
                str.append(curr.getItem());
            }
        }
        return str.toString();
    }
}
::::::::::::::
QueueException.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Exception for Queues to throw.
 * Status: Complete and thoroughly tested
 * Last update: 11/24/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.11.24
 */

public class QueueException extends RuntimeException {

    /**
     * Creates a new QueueException with a description of what happened.
     * @param s The description of what happened.
     */
    public QueueException(String s) {
        super(s);
    }  // end constructor
}  // end QueueException
::::::::::::::
QueueInterface.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Interface for Queues.
 * Status: Complete and thoroughly tested
 * Last update: 11/24/19
 * Submitted:  12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.11.24
 */

public interface QueueInterface<T> {

    /**
     * Returns whether the Queue is empty or not.
```

```
     * @return True if the queue is empty; False otherwise.
     */
    public boolean isEmpty();
    // Determines whether a queue is empty.
    // Precondition: None.
    // Postcondition: Returns true if the queue is empty;
    // otherwise returns false.

    /**
     * Enqueues an item to the back of the Queue.
     * @param newItem The item to be enqueued.
     * @throws QueueException when the item cannot be enqueued for some reason.
     */
    public void enqueue(T newItem) throws QueueException;
    // Adds an item at the back of a queue.
    // Precondition: newItem is the item to be inserted.
    // Postcondition: If the operation was successful, newItem
    // is at the back of the queue. Some implementations
    // may throw QueueException if newItem cannot be added
    // to the queue.

    /**
     * Dequeues the item from the front of the queue.
     * @throws QueueException when attempting to dequeue from an empty queue.
     * @return The dequeued item.
     */
    public T dequeue() throws QueueException;
    // Retrieves and removes the front of a queue.
    // Precondition: None.
    // Postcondition: If the queue is not empty, the item that
    // was added to the queue earliest is removed. If the queue is
    // empty, the operation is impossible and QueueException is thrown.

    /**
     * Dequeues all items from the queue.
     */
    public void dequeueAll();
    // Removes all items of a queue.
    // Precondition: None.
    // Postcondition: The queue is empty.

    /**
     * Returns the item at the front of the queue.
     * @throws QueueException when attempting to peek at an empty queue.
     * @return The item at the front.
     */
    public T peek() throws QueueException;
    // Retrieves the item at the front of a queue.
    // Precondition: None.
    // Postcondition: If the queue is not empty, the item
    // that was added to the queue earliest is returned.
    // If the queue is empty, the operation is impossible
    // and QueueException is thrown.

    /**
     * Returns a String representation of the Queue.
     * @return The String representation.
     */
    public String toString();
}  // end QueueInterface

::::::::::::::
```

Runway.java

```
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Project: Runway for planes to take off o
f or land on.
 * Status: Complete and thoroughly tested
 * Last update: 12/3/19
 * Submitted: 12/03/19
 * Comment: test suite and sample run attached
 * @author: Devyn Melendez
 * @version: 2019.12.03
 */

public class Runway
{
    private String name;
    private QueueCSLS<Plane> departures;
    private QueueCSLS<Plane> arrivals;

    /**
     * Creates a new runway for planes and sets up the queues for
     * arriving planes and departing planes.
     * @param n The runway's name.
     */
    public Runway(String n)
    {
        name = n;
        departures = new QueueCSLS<Plane>();
        arrivals = new QueueCSLS<Plane>();
    }

    /**
     * Enqueues a new plane that will depart from this runway.
     * @param p The departing plane.
     */
    public void addDeparture(Plane p)
    {
        departures.enqueue(p);
    }

    /**
     * Dequeues the oldest departing plane from this runway.
     * @return The removed plane.
     */
    public Plane removeDeparture()
    {
        return departures.dequeue();
    }

    /**
     * Enqueues a new plane that will arrive on this runway.
     * @param p The arriving plane.
     */
    public void addArrival(Plane p)
    {
        arrivals.enqueue(p);
    }

    /**
     * Dequeues the oldest arriving plane from this runway.
     * @return The removed plane.
     */
```

```
    public Plane removeArrival()
    {
        return arrivals.dequeue();
    }

    /**
     * Returns the oldest plane set to depart from this runway.
     * @return The oldest plane.
     */
    public Plane peekDepartures()
    {
        return departures.peek();
    }

    /**
     * Returns the oldest plane set to arrive on this runway.
     * @return The oldest plane.
     */
    public Plane peekArrivals()
    {
        return arrivals.peek();
    }

    /**
     * Returns true if there are no planes set to depart from
     * this runway; returns false otherwise.
     * @return Boolean stating whether or not there are departing planes.
     */
    public boolean noDepartures()
    {
        return departures.isEmpty();
    }

    /**
     * Returns true if there are no planes set to arrive on
     * this runway; returns true otherwise.
     * @return Boolean stating whether or not there are arriving planes.
     */
    public boolean noArrivals()
    {
        return arrivals.isEmpty();
    }

    /**
     * Returns the runway's name.
     * @return The name.
     */
    public String getName()
    {
        return name;
    }

    /**
     * Sets a new name for the runway.
     * @param n The new name.
     */
    public void setName(String n)
    {
        name = n;
    }

    /**
     * Returns the queue of departing planes.
     * @return The departures.
     */
    public QueueCSLS<Plane> getDepartures()
    {
        return departures;
    }

    /**
     * Sets a new queue for departing planes.
     * @param d The new queue.
     */
    public void setDepartures(QueueCSLS<Plane> d)
    {
        departures = d;
    }

    /**
     * Returns the queue of arriving planes.
     * @return The arrivals.
     */
    public QueueCSLS<Plane> getArrivals()
    {
        return arrivals;
    }

    /**
     * Sets a new queue for arriving planes.
     * @param a The new queue.
     */
    public void setArrivals(QueueCSLS<Plane> a)
    {
        arrivals = a;
    }

    /**
     * Returns a listing of the planes waiting for takeoff on this runway.
     * @return The listing.
     */
    public String listDepartures()
    {
        StringBuilder str = new StringBuilder();
        if(departures.isEmpty())
        {
            str.append("No planes are waiting to takeoff from runway " + name + ".
");
        }
        else
        {
            str.append("These planes are waiting to takeoff from runway " + name +
 ":");
            str.append("\n");
            str.append(departures.toString());
        }
        return str.toString();
    }

    /**
     * Returns a listing of the planes waiting to land on this runway.
     * @return The listing.
     */
    public String listArrivals()
```

```
    {
        StringBuilder str = new StringBuilder();
        if(arrivals.isEmpty())
        {
            str.append("No planes are waiting to land on runway " + name + ".");
        }
        else
        {
            str.append("These planes are waiting to land on runway " + name + ":")
;
            str.append("\n");
            str.append(arrivals.toString());
        }
        return str.toString();
    }

    /**
     * Returns a complete listing of planes waiting to takeoff and land on this ru
nway.
     * @return The listing.
     */
    public String toString()
    {
        StringBuilder str = new StringBuilder();

        str.append(listDepartures());
        str.append("\n");
        str.append(listArrivals());

        return str.toString();
    }
}
::::::::::::::
airport1.output
::::::::::::::
Initializing Airport
Enter number of runways: 3
Enter name for runway #1: NorthEast
Enter name for runway #2: SouthWest
Enter name for runway #3: West
Select from the following menu:
        0. Exit program.
        1. Plane enters the system.
        2. Plane attempts takes off.
        3. Plane is allowed to re-enter a runway.
        4. Runway opens.
        5. Runway closes.
        6. Display info about planes waiting to take off.
        7. Display info about planes waiting to be allowed to re-enter a runway.
        8. Display number of planes who have taken off.
        9. Plane attemps landing.
        10. Display info about planes waiting to land.
        11. Display number of planes who have landed.
Make your selection now: 2
No plane on any runway.
Make your selection now: 3
No planes in wait list.
Make your selection now: 6
No planes are waiting to takeoff from runway NorthEast.

No planes are waiting to takeoff from runway SouthWest.

No planes are waiting to takeoff from runway West.

Make your selection now: 7
No flights are waiting for clearance.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 1
Enter flight number: USAir705
Enter destination: Boston
Enter runway: NorthEast
Flight USAir705 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: AirFrance212
Enter destination: Paris
Enter runway: NorthEast
Flight AirFrance212 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: British909
Enter destination: London
Enter runway: NorthEast
Flight British909 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: United954
Enter destination: Pittsburgh
Enter runway: NorthWest
No such runway.
Enter runway: West
Flight United954 is now waiting for takeoff on runway West.

Make your selection now: 1
Enter flight number: Delta204
Enter destination: Chicago
Enter runway: NorthEast
Flight Delta204 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: USAir305
Enter destination: San Diego
Enter runway: West
Flight USAir305 is now waiting for takeoff on runway West.

Make your selection now: 1
Enter flight number: United572
Enter destination: Fort Lauderdale
Enter runway: SouthWest
Flight United572 is now waiting for takeoff on runway SouthWest.

Make your selection now: 3
No planes in wait list.
Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight USAir705 to Boston.
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

These planes are waiting to takeoff from runway SouthWest:
```

```
Flight United572 to Fort Lauderdale.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.

Make your selection now: 7
No flights are waiting for clearance.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 2
Is Flight USAir705 to Boston. clear for takeoff(Y/N): N
USAir705 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

These planes are waiting to takeoff from runway SouthWest:
Flight United572 to Fort Lauderdale.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 1
Enter flight number: American493
Enter destination: Seattle
Enter runway: West
Flight American493 is now waiting for takeoff on runway West.

Make your selection now: 2
Is Flight United572 to Fort Lauderdale. clear for takeoff(Y/N): Y
United572 has taken off from runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight United954 to Pittsburgh. clear for takeoff(Y/N): N
United954 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight AirFrance212 to Paris. clear for takeoff(Y/N): N
AirFrance212 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight USAir305 to San Diego. clear for takeoff(Y/N): Y
USAir305 has taken off from runway West.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.
```

```
These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 1
Enter flight number: Continental339
Enter destination: Montreal
Enter runway: NorthEast
Flight Continental339 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: jetBlue856
Enter destination: Atlanta
Enter runway: SouthWest
Flight jetBlue856 is now waiting for takeoff on runway SouthWest.

Make your selection now: 1
Enter flight number: AmericaWest691
Enter destination: San Francisco
Enter runway: West
Flight AmericaWest691 is now waiting for takeoff on runway West.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 4
Enter the name of the runway to open: West
Runway already exists.
Enter the name of the runway to open: East
East was added.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

No planes are waiting to takeoff from runway East.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 1
Enter flight number: Lufthansa581
Enter destination: Muenchen
Enter runway: East
Flight Lufthansa581 is now waiting for takeoff on runway East.

Make your selection now: 1
Enter flight number: Alitalia576
Enter destination: Rome
Enter runway: East
Flight Alitalia576 is now waiting for takeoff on runway East.

Make your selection now: 1
Enter flight number: Continental304
Enter destination: Miami
Enter runway: SouthWest
Flight Continental304 is now waiting for takeoff on runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.
```

```
Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 2
Is Flight British909 to London. clear for takeoff(Y/N): Y
British909 has taken off from runway NorthEast.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
3 takeoffs have occurred.

Make your selection now: 3
Enter the the flight number: USAir705

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
3 takeoffs have occurred.

Make your selection now: 2
Is Flight jetBlue856 to Atlanta. clear for takeoff(Y/N): Y
jetBlue856 has taken off from runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.

These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
4 takeoffs have occurred.

Make your selection now: 2
Is Flight American493 to Seattle. clear for takeoff(Y/N): Y
American493 has taken off from runway West.

Make your selection now: 5
Enter the name of the runway to close: North
Runway does not exist.
Enter the name of the runway to close: West
West
Enter new runway for AmericaWest691: West
This is the runway that will close.
Enter new runway for AmericaWest691: North
No such runway.
Enter new runway for AmericaWest691: NorthEast
AmericaWest691 is now assigned to runway NorthEast.
Enter new runway for United954: SouthWest
United954 is now assigned to runway SouthWest
West has been closed.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.
```

```
Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
5 takeoffs have occurred.

Make your selection now: 3
Enter the the flight number: United953
United953 is not waiting for clearance.
Enter the the flight number: United954

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.
Flight United954 to Pittsburgh.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight AirFrance212 to Paris.

Make your selection now: 8
5 takeoffs have occurred.

Make your selection now: 0

::::::::::::::
airport.output
::::::::::::::
Initializing Airport
Enter number of runways: 3
Enter name for runway #1: NorthEast
Enter name for runway #2: SouthWest
Enter name for runway #3: West
Select from the following menu:
        0. Exit program.
        1. Plane enters the system.
        2. Plane attempts takes off.
        3. Plane is allowed to re-enter a runway.
        4. Runway opens.
        5. Runway closes.
        6. Display info about planes waiting to take off.
        7. Display info about planes waiting to be allowed to re-enter a runway.
        8. Display number of planes who have taken off.
        9. Plane attemps landing.
        10. Display info about planes waiting to land.
        11. Display number of planes who have landed.
Make your selection now: 2
No plane on any runway.
Make your selection now: 3
No planes in wait list.
Make your selection now: 6
No planes are waiting to takeoff from runway NorthEast.

No planes are waiting to takeoff from runway SouthWest.

No planes are waiting to takeoff from runway West.

Make your selection now: 7
No flights are waiting for clearance.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 1
Enter flight number: USAir705
Enter destination: Boston
Enter runway: NorthEast
Flight USAir705 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: AirFrance212
Enter destination: Paris
Enter runway: NorthEast
Flight AirFrance212 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: British909
Enter destination: London
Enter runway: NorthEast
Flight British909 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: United954
Enter destination: Pittsburgh
Enter runway: NorthWest
No such runway.
Enter runway: West
Flight United954 is now waiting for takeoff on runway West.

Make your selection now: 1
Enter flight number: Delta204
Enter destination: Chicago
Enter runway: NorthEast
Flight Delta204 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: USAir305
Enter destination: San Diego
Enter runway: West
Flight USAir305 is now waiting for takeoff on runway West.

Make your selection now: 1
Enter flight number: United572
Enter destination: Fort Lauderdale
Enter runway: SouthWest
Flight United572 is now waiting for takeoff on runway SouthWest.

Make your selection now: 3
No planes in wait list.
Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
```

```
Flight USAir705 to Boston.
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

These planes are waiting to takeoff from runway SouthWest:
Flight United572 to Fort Lauderdale.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.

Make your selection now: 7
No flights are waiting for clearance.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 2
Is Flight USAir705 to Boston. clear for takeoff(Y/N): N
USAir705 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

These planes are waiting to takeoff from runway SouthWest:
Flight United572 to Fort Lauderdale.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.

Make your selection now: 8
0 takeoffs have occurred.

Make your selection now: 1
Enter flight number: American493
Enter destination: Seattle
Enter runway: West
Flight American493 is now waiting for takeoff on runway West.

Make your selection now: 2
Is Flight United572 to Fort Lauderdale. clear for takeoff(Y/N): Y
United572 has taken off from runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight United954 to Pittsburgh.
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight United954 to Pittsburgh. clear for takeoff(Y/N): N
United954 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight AirFrance212 to Paris.
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight AirFrance212 to Paris. clear for takeoff(Y/N): N
AirFrance212 has been denied clearance.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight USAir305 to San Diego.
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
1 takeoffs have occurred.

Make your selection now: 2
Is Flight USAir305 to San Diego. clear for takeoff(Y/N): Y
USAir305 has taken off from runway West.
```

```
Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.

No planes are waiting to takeoff from runway SouthWest.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 1
Enter flight number: Continental339
Enter destination: Montreal
Enter runway: NorthEast
Flight Continental339 is now waiting for takeoff on runway NorthEast.

Make your selection now: 1
Enter flight number: jetBlue856
Enter destination: Atlanta
Enter runway: SouthWest
Flight jetBlue856 is now waiting for takeoff on runway SouthWest.

Make your selection now: 1
Enter flight number: AmericaWest691
Enter destination: San Francisco
Enter runway: West
Flight AmericaWest691 is now waiting for takeoff on runway West.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 4
Enter the name of the runway to open: West
Runway already exists.

Enter the name of the runway to open: East
East was added.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

No planes are waiting to takeoff from runway East.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 1
Enter flight number: Lufthansa581
Enter destination: Muenchen
Enter runway: East
Flight Lufthansa581 is now waiting for takeoff on runway East.

Make your selection now: 1
Enter flight number: Alitalia576
Enter destination: Rome
Enter runway: East
Flight Alitalia576 is now waiting for takeoff on runway East.

Make your selection now: 1
Enter flight number: Continental304
Enter destination: Miami
Enter runway: SouthWest
Flight Continental304 is now waiting for takeoff on runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight British909 to London.
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.
```

```
Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
2 takeoffs have occurred.

Make your selection now: 2
Is Flight British909 to London. clear for takeoff(Y/N): Y
British909 has taken off from runway NorthEast.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight USAir705 to Boston.
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
3 takeoffs have occurred.

Make your selection now: 3
Enter the the flight number: USAir705

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.

These planes are waiting to takeoff from runway SouthWest:
Flight jetBlue856 to Atlanta.
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
3 takeoffs have occurred.

Make your selection now: 2
Is Flight jetBlue856 to Atlanta. clear for takeoff(Y/N): Y
jetBlue856 has taken off from runway SouthWest.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.

These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway West:
Flight American493 to Seattle.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
4 takeoffs have occurred.

Make your selection now: 2
Is Flight American493 to Seattle. clear for takeoff(Y/N): Y
American493 has taken off from runway West.

Make your selection now: 5
Enter the name of the runway to close: North
Runway does not exist.
Enter the name of the runway to close: West
West
Enter new runway for AmericaWest691: West
This is the runway that will close.
Enter new runway for AmericaWest691: North
No such runway.
Enter new runway for AmericaWest691: NorthEast
AmericaWest691 is now assigned to runway NorthEast.
Enter new runway for United954: SouthWest
United954 is now assigned to runway SouthWest
West has been closed.

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.
Flight AmericaWest691 to San Francisco.
```

```
These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight United954 to Pittsburgh.
Flight AirFrance212 to Paris.

Make your selection now: 8
5 takeoffs have occurred.

Make your selection now: 3
Enter the the flight number: United953
United953 is not waiting for clearance.
Enter the the flight number: United954

Make your selection now: 6
These planes are waiting to takeoff from runway NorthEast:
Flight Delta204 to Chicago.
Flight Continental339 to Montreal.
Flight USAir705 to Boston.
Flight AmericaWest691 to San Francisco.

These planes are waiting to takeoff from runway SouthWest:
Flight Continental304 to Miami.
Flight United954 to Pittsburgh.

These planes are waiting to takeoff from runway East:
Flight Lufthansa581 to Muenchen.
Flight Alitalia576 to Rome.

Make your selection now: 7
These flights are waiting for clearance:
Flight AirFrance212 to Paris.

Make your selection now: 8
5 takeoffs have occurred.

Make your selection now: 0

::::::::::::::
Rationale.txt
::::::::::::::
ADTs & Justification
We used the Queue ADT in order to store Plane objects within the Runway objects. P
lanes must be accessed in a FIFO order by the AirportSystem, so naturally choosing
 Queue made the most sense. We chose to use a circular singularly-linked structure
 implementation (QueueCSLS) in order to save on memory usage. Queueing and dequeue
ing of Planes is the most frequent operation performed in the program, so choosing
 an efficient implementation was important.
We used the List ADT in order to store Runways in a collection, store Planes that
are waiting to be re-entered into a runway in a collection, and also store Plane f
light numbers (Strings) in a collection. This is because these collections must be
 searched through by user input when choosing to close a Runway, specifying a plan
e to re-enter into a runway, and checking whether a Plane already exists when a ne
w Plane is created. A resizable array-based implementation (ListRAB) is used in or
der to save on memory usage by the internal array, since the majority of operation
s on it will be using direct index access.

Menu Options and Data Flow
0. Exit program.
    Sets loop variable to false.
1. Plane enters the system.
    Collects input to create a Plane, sends new Plane to AirportSystem, which veri
fies flightNumber and Runway against existing values.
    Most frequent operation is expected to be .get().
2. Plane attempts takes off.
    Peeks the next available plane for takeoff, and collects whether it can takeof
f. Send that permission input to a method in AirportSystem that dequeues the plane
 and either moves that plane to the waiting List, or removes it from the Runway an
d the activeFlights List, then increments takeoff tracking int. Most frequent oper
ation is expected to be .get().
3. Plane is allowed to re-enter a runway.
    Takes input of a plane flightNumber, verifies in activeFlights of AirportSyste
m, then moves from waiting List to itâ\200\231s designate Runway. Most frequent op
eration is expected to be .get().
4. Runway opens.
    Takes input of a Runway name, verifies against existing runway, and then adds
to the List of Runway in AirportSystem if able. Most frequent operation is expecte
d to be .get().
5. Runway closes.
    Takes input of runway name, temporarily calls the runway, dequeues all Planes
in both Queues, and also scans through the waiting List to assign new Runways to e
ach. The deletes the Runway. Most frequent operation is expected to be .dequeue().
6. Display info about planes waiting to take off.
    Outputs toString() data from each Runway and Plane. No significant operations.
7. Display info about planes waiting to be allowed to re-enter a runway.
Outputs toString() data from the waiting List and each Plane. No significant opera
tions.
8. Display number of planes who have taken off.
    Outputs toString() of number of planes taken off. No significant operations.
9. Plane attemps landing.
Peeks the next available plane for landingf, and collects whether it can do so. Se
nd that permission input to a method in AirportSystem that dequeues the plane and
either moves that plane to the waiting List, or removes it from the Runway and the
 activeFlights List, then increments landing tracking int. Most frequent operation
 is expected to be .get().
10. Display info about planes waiting to land.
    Outputs toString() data from each Runway and Plane. No significant operations.
11. Display number of planes who have landed.
    Outputs toString() of number of planes that have landed. No significant operat
ions.



::::::::::::::
Devyn_Melendez_Nick_Bovee_submission
::::::::::::::
Script started on 2019-12-03 10:21:42-05:00 [TERM="xterm" TTY="/dev/pts/14" COLUMN
S="207" LINES="59"]
\033]0;melend53@elvix2:~/DSA/Project\007[melend53@elvis Project]$ ls
airport1.input              AirportSystem.java.orig              Driver.class
  ListAB.java.orig                      ListRAB.html        package-summary.htm
l  QueueInterface.class
airport1.output             allclasses-frame.html                Driver.html
  ListException.class                   ListRAB.java        package-tree.html
   QueueInterface.java
AirportException.class      allclasses-noframe.html              Driver.java
  ListException.java                    ListRAB.java.orig   Plane.class
   Rationale.txt
```

```
AirportException.java       constant-values.html                Driver.java.orig
  ListIndexOutOfBoundsException.class      Node.class          Plane.html
   Runway.class
AirportException.java.orig  deprecated-list.html                help-doc.html
  ListIndexOutOfBoundsException.java       Node.html           Plane.java
   Runway.html
airport.input               Devyn_Melendez_Nick_Bovee_submission  index-all.html
  ListIndexOutOfBoundsException.java.orig  Node.java           QueueCSLS.class
   Runway.java
airport.output              Driver1.class                       index.html
  ListInterface.class                    output.pdf           QueueCSLS.html
   script.js
AirportSystem.class         Driver1.html                        ListAB.class
  ListInterface.java                     overview-tree.html   QueueCSLS.java
   stylesheet.css
AirportSystem.html          Driver1.java                        ListAB.html
  ListInterface.java.orig                package-frame.html   QueueException.clas
s
AirportSystem.java          Driver1.java.orig                   ListAB.java
  ListRAB.class                          package-list         QueueException.java
\033]0;melend53@elvix2:~/DSA/Project\007[melend53@elvis Project]$ cd ..
\033]0;melend53@elvix2:~/DSA\007[melend53@elvis DSA]$ cp -r proje\033[K\033[K\033[K
\033[K\033[K\033[KPrp\033[K\033[Koject ~hristescu/DSA/\033[K\033[K\033[K\033[KHDSA/Projec
t/melend53
\033]0;melend53@elvix2:~/DSA\007[melend53@elvis DSA]$ ~hristes\033[Kcu/HDSA/\033[K
\033[KA/Proe\033[K\033[Kject\033[K/Grp\033[Kades/fix_perms.csh

\033[31m1. Fixing permissions for \033(B\033[m melend53 \033[31m... done.

\033[31m2. Check what was submitted:\033(B\033[m
-rw-r--r-- 1 melend53 domain users   682 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport1.input
-rw-r--r-- 1 melend53 domain users 14067 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport1.output
-rw-r--r-- 1 melend53 domain users   239 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportException.class
-rw-r--r-- 1 melend53 domain users   548 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportException.java
-rw-r--r-- 1 melend53 domain users   545 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportException.java.orig
-rw-r--r-- 1 melend53 domain users   682 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport.input
-rw-r--r-- 1 melend53 domain users 14067 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport.output
-rw-r--r-- 1 melend53 domain users  5995 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportSystem.class
-rw-r--r-- 1 melend53 domain users 26815 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportSystem.html
-rw-r--r-- 1 melend53 domain users 16416 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportSystem.java
-rw-r--r-- 1 melend53 domain users 15976 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/AirportSystem.java.orig
-rw-r--r-- 1 melend53 domain users  1407 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/allclasses-frame.html
-rw-r--r-- 1 melend53 domain users  1227 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/allclasses-noframe.html
-rw-r--r-- 1 melend53 domain users  3507 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/constant-values.html
-rw-r--r-- 1 melend53 domain users  3457 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/deprecated-list.html
-rw-r--r-- 1 melend53 domain users     0 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Devyn_Melendez_Nick_Bovee_submission
-rw-r--r-- 1 melend53 domain users  7231 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver1.class
-rw-r--r-- 1 melend53 domain users 15919 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver1.html
-rw-r--r-- 1 melend53 domain users 12896 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver1.java
-rw-r--r-- 1 melend53 domain users 12875 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver1.java.orig
-rw-r--r-- 1 melend53 domain users  6968 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver.class
-rw-r--r-- 1 melend53 domain users 15907 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver.html
-rw-r--r-- 1 melend53 domain users 12379 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver.java
-rw-r--r-- 1 melend53 domain users 12358 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Driver.java.orig
-rw-r--r-- 1 melend53 domain users  7892 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/help-doc.html
-rw-r--r-- 1 melend53 domain users 34450 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/index-all.html
-rw-r--r-- 1 melend53 domain users  2754 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/index.html
-rw-r--r-- 1 melend53 domain users  1512 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListAB.class
-rw-r--r-- 1 melend53 domain users 15273 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListAB.html
-rw-r--r-- 1 melend53 domain users  4070 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListAB.java
-rw-r--r-- 1 melend53 domain users  4056 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListAB.java.orig
-rw-r--r-- 1 melend53 domain users   233 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListException.class
-rw-r--r-- 1 melend53 domain users   420 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListException.java
-rw-r--r-- 1 melend53 domain users   274 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListIndexOutOfBoundsException.class
-rw-r--r-- 1 melend53 domain users   650 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListIndexOutOfBoundsException.java
-rw-r--r-- 1 melend53 domain users   651 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListIndexOutOfBoundsException.java.orig
-rw-r--r-- 1 melend53 domain users   455 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListInterface.class
-rw-r--r-- 1 melend53 domain users  1243 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListInterface.java
-rw-r--r-- 1 melend53 domain users  1241 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListInterface.java.orig
-rw-r--r-- 1 melend53 domain users  1592 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListRAB.class
-rw-r--r-- 1 melend53 domain users 12241 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListRAB.html
-rw-r--r-- 1 melend53 domain users  3440 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListRAB.java
-rw-r--r-- 1 melend53 domain users  3425 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/ListRAB.java.orig
-rw-r--r-- 1 melend53 domain users   888 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Node.class
-rw-r--r-- 1 melend53 domain users 12567 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Node.html
-rw-r--r-- 1 melend53 domain users  1684 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Node.java
-rw-r--r-- 1 melend53 domain users 81620 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/output.pdf
```

```
-rw-r--r-- 1 melend53 domain users  4839 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/overview-tree.html
-rw-r--r-- 1 melend53 domain users  1514 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/package-frame.html
-rw-r--r-- 1 melend53 domain users     1 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/package-list
-rw-r--r-- 1 melend53 domain users  5121 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/package-summary.html
-rw-r--r-- 1 melend53 domain users  4848 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/package-tree.html
-rw-r--r-- 1 melend53 domain users  1058 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Plane.class
-rw-r--r-- 1 melend53 domain users 13306 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Plane.html
-rw-r--r-- 1 melend53 domain users  2084 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Plane.java
-rw-r--r-- 1 melend53 domain users  1677 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueCSLS.class
-rw-r--r-- 1 melend53 domain users 13890 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueCSLS.html
-rw-r--r-- 1 melend53 domain users  2907 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueCSLS.java
-rw-r--r-- 1 melend53 domain users   235 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueException.class
-rw-r--r-- 1 melend53 domain users   581 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueException.java
-rw-r--r-- 1 melend53 domain users   467 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueInterface.class
-rw-r--r-- 1 melend53 domain users  2560 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/QueueInterface.java
-rw-r--r-- 1 melend53 domain users  3545 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Rationale.txt
-rw-r--r-- 1 melend53 domain users  2497 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Runway.class
-rw-r--r-- 1 melend53 domain users 21300 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Runway.html
-rw-r--r-- 1 melend53 domain users  4936 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/Runway.java
-rw-r--r-- 1 melend53 domain users   827 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/script.js
-rw-r--r-- 1 melend53 domain users 12842 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/stylesheet.css

\033[31m3. Let's make sure your code compiles:\033(B\033[m
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

\033[31m4. Check that output files were submitted:\033(B\033[m
-rw-r--r-- 1 melend53 domain users 14067 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport1.output
-rw-r--r-- 1 melend53 domain users 14067 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/airport.output
-rw-r--r-- 1 melend53 domain users 81620 Dec  3 10:22 /home/hristescu/HDSA/Project
/melend53/Project/output.pdf

\033[31mIf anything does not look right or if your code doesn't compile, please fi
x and resubmit.\033(B\033[m
\033]0;melend53@elvix2:~/DSA\007[melend53@elvis DSA]$ exit

Script done on 2019-12-03 10:23:37-05:00 [COMMAND_EXIT_CODE="0"]
```