

Seat Reservation Problem

Group 20

Name	Index No	Registration No
Dumindu Kavishka	19000693	2019/CS/069
Mohammed Ashfaq	19000081	2019/CS/008
Thisara Dilshan	19000359	2019/CS/035
Udesh Maduranga	19000871	2019/CS/087

Problem

Flight management system to manage various flights with their relent traveling paths. Passenger shall have the facility to book seats from anywhere in the flight. passengers shall have the facility to book seats from any location to any location.

Algorithm

flight considered as an **object** (for easy handling of multiple flights).
In side a flight object a **3D array** is used to store relevant seat data, (hence a seat can be booked by multiple passengers, given that the seat is free upon previous passenger's arrival on the destination)
main menu is called through a infinite loop

ex:-

flight with 8 rows and 10 columns with a traveling path starting from

→ Sri_lanka , India , china , USA , England

passenger 1 can book seat 2,1 from sri_lanka to china

passenger 2 can book the same seat from china to USA

passenger 3 can book the same seat from USA to England

→ implimentation

```
--- Main Menu ---
1.) create flight
2.) view flight list and path
3.) remove flight --not implemented
4.) book a seat
5.) remove booked seat --not implemented
6.) view passenger
7.) view seat layaout(booked)

0.) exit
option -> 1
```

→

```
option -> 1
flight name -> f1
rows - 8
columns - 9
traveling path - sri_lanka,india,china,usa,england
```

```
flight name -> f1
name -> dumindu
from -> sri_lanka
to -> china
specify a seat : y/ny
seat row -> 3
seat col -> 4
booked seat -> row 3 column 4 dumindu sri_lanka
booked seat -> row 3 column 4 dumindu india
```

→

```
option -> 4
flight name -> f1
name -> kavishka
from -> china
to -> usa
specify a seat : y/ny
seat row -> 3
seat col -> 4
booked seat -> row 3 column 4 kavishka china
```

```
option -> 4
flight name -> f1
name -> kankanamage
from -> usa
to -> england
specify a seat : y/ny
seat row -> 3
seat col -> 4
booked seat -> row 3 column 4 kankanamage usa
```

→

```
option -> 7
flight name -> f1
from -> sri_lanka
to ->

- - - - -
- - - - -
- - X - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
```

```
- - - - -
- - - X - - -
- - - - -
- - - - -
- - - - -
- - - - -
dumindu sri_lanka -> china
```

```
kavishka china -> usa
```

```
- - - - -  
- - - X - - -  
- - - - -  
- - - - -  
- - - - -  
kankanamage usa -> england
```

solution

```
import scala.io.StdIn.{readLine, readInt}
import scala.math._
import scala.Array._
import util.control.Breaks._
import scala.collection.mutable._

object seat_reservation{
  def main(args: Array[String]): Unit = {
    // variables
    var exit: Boolean = false; var user_in: String = ""; val fly_list = ArrayBuffer[flight]()
    //
    while(!exit){// main menu loop
      println(s"\n--- Main Menu ---");
      println(f"1.) create flight\n2.) view flight list and path\n3.) remove flight --not
implemented")
      println(f"4.) book a seat\n5.) remove booked seat --not implemented\n6.) view passenger\
n7.) view seat layout(booked)")
      println(s"\n0.) exit")
      user_in = readLine("option -> ")
      //
      if(user_in == "1"){// create a flight
        //variables
        var dupli: Boolean = false; var f_name: String = ""; var rows,cols: Int = -1; var path =
ArrayBuffer[String]();
        //
        breakable{
          while(true){
            print("flight name -> "); f_name = readLine()
            print("rows - "); rows = readInt()
            print("columns - "); cols = readInt()
            print("traveling path - "); path = readLine().split(",").to(ArrayBuffer);
            // check for duplication
            breakable{
              for(x <- fly_list){
                if(x.fly_name() == f_name){
                  println(s"flight already exist")
                  user_in = readLine("enter again -> y/n : ")
                  if(user_in == "y"){ dupli = true; break() }
                }
              }
            }
            if(!dupli){ break() }
          }
        }
        fly_list += new flight(f_name,rows,cols,path)// create the new object
      }
      else if(user_in == "2"){// view flight list and path
        if(fly_list.length <= 0){
```

```

        println("no flight available")
    }
    else{
        for(x <- 0 to fly_list.length-1){
            println(fly_list(x).fly_name()+"\t"+fly_list(x).fly_path()); // print flight name
        }
    }
}
else if(user_in == "3"){// remove flight --not implemented
    println("sorry...")
}
else if(user_in == "4"){// book a seat
    //variables
    var fly_check: Boolean = false; var f_no: Int = -1;
    breakable{// gather information through a loop
        while(true){
            print("flight name -> "); var f_name = readLine()
            // check for flight
            breakable{
                for(x <- 0 to fly_list.length-1){ if(fly_list(x).fly_name() == f_name){ fly_check =
true; f_no = x; break() } }
            }
            if(!fly_check){ println("no such flight"); break() }
            //
            print("name -> "); var name = readLine()
            print("from -> "); var from = readLine()
            print("to -> "); var to = readLine()
            print("specify a seat : y/n"); user_in = readLine()
            if(user_in == "y"){// seat
                print("seat row -> "); var row = readInt()
                print("seat col -> "); var col = readInt()
                // book the flight
                fly_list(f_no).book(name,from,to,row,col); break()
            }
            else{// no seat
                // book the flight
                fly_list(f_no).book(name,from,to); break()
            }
        }
    }
}
else if(user_in == "5"){// remove booked seat --not implemented
    println("sorry...")
}
else if(user_in == "6"){// view passenger
    var fly_no: Int = -1; var found: Boolean = false; var name: String = "";

    print("flight name -> "); var fly_name = readLine("n if no idea : ")
    print("name -> "); name = readLine()
    if(fly_name == "n"){// flight not given
        breakable{

```



```

def book(name: String, from: String, to: String): Unit = {
  // check for name in pass.array
  var stat: Boolean = false; var count: Int = 0;
  breakable{
    for(x <- pass){if(x == name){stat = true}}
  }
  //
  if(stat == true){
    println("already in"); return;
  }
  else{
    // where to look
    var i_from, i_to : Int = -1
    i_from = path.indexOf(from); i_to = path.indexOf(to)

    //traverse the seat 3D array
    breakable{
      for(row <- 0 to rows-1; col <- 0 to cols-1){// traverse the rows and cols
        for(x <- i_from to i_to-1){
          if(seat(row)(col)(x) == "***empty**"){count+=1} // checking for availability
        }
        if((i_to-i_from) == (count)){// space available
          for(x <- i_from to i_to-1){
            seat(row)(col)(x) = name
            println(f"booked seat -> row ${row+1} column ${col+1} " + name + " "+path(x))
          }
          breakable{// add name to the pass array
            for(x <- 0 to pass.length-1){
              if(pass(x) == "***empty**"){
                pass(x) = name
                break()
              }
            }
          }
          stat = true
        }
        else{// space not available
          println("seat not available"); return;
        }
      }
    }
  }
}

```

```

def book(name: String, from: String, to: String, s_row: Int, s_col: Int): Unit = {
  // variables
  var stat: Boolean = false; var count: Int = 0;
  // check for name in pass.array
  breakable{
    for(x <- pass){if(x == name){stat = true; break()}}
  }
}

```

```

    }
    //
    if(stat == true){
        println("already booked"); return;
    }
    else{
        // where to look
        var i_from, i_to : Int = -1
        i_from = path.indexOf(from); i_to = path.indexOf(to)
        // traverse the path in seat
        breakable{// may be usefull
            for(x <- i_from to i_to-1){// checking for availability
                if(seat(s_row-1)(s_col-1)(x) == "***empty***"){count+=1}
            }
            if((i_to-i_from) == (count)){// space available
                for(x <- i_from to i_to-1){
                    seat(s_row-1)(s_col-1)(x) = name
                    println(f"booked seat -> row ${s_row} column ${s_col} " + name + " " + path(x))
                }
                breakable{// add name to the pass array
                    for(x <- 0 to pass.length-1){
                        if(pass(x) == "***empty***"){
                            pass(x) = name
                            break()
                        }
                    }
                }
            }
        }
        else{// space not available
            println("seat not available"); return;
        }
    }
}
}

```

```

def view(from: String, to: String): Unit = {// print the seat layout
    // where to look
    var i_from, i_to : Int = -1
    i_from = path.indexOf(from); i_to = path.indexOf(to)
    var occu: Boolean = false
    //
    for(row <- 0 to rows-1){// traverse rows
        for(col <- 0 to cols-1){// traverse cols
            occu = false
            for(x <- i_from to i_to-1){// traverse the path
                if(seat(row)(col)(x) != "***empty***"){occu = true}
            }
            if(occu) { print(" X ") } else { print(" _ ") }
        }
        print("\n")
    }
}

```



```

def view(from: String): Unit = { // print the seat layout
    // where to look
    var i_from, i_to : Int = -1
    i_from = path.indexOf(from); i_to = path.length
    var occu: Boolean = false;
    //
    for(row <- 0 to rows-1){ // traverse rows
        for(col <- 0 to cols-1){ // traverse cols
            occu = false
            for(x <- i_from to i_to-1){ // traverse the path
                if(seat(row)(col)(x) != "***empty**"){ occu = true }
            }
            if(occu) { print(" X ") } else { print(" _ ") }
        }
        print("\n")
    }
}

```

```

def view_pass_seat(name: String): Unit = {
    // where to look
    var occu: Boolean = false; var from,to,count: Int = 0;
    //
    for(row <- 0 to rows-1){ // traverse rows
        for(col <- 0 to cols-1){ // traverse cols
            occu = false
            for(x <- 0 to path.length-1){ // traverse the path
                if(seat(row)(col)(x) == name){
                    occu = true; count+=1; to = x+1
                    if(count == 1){ from = x };
                }
            }
            if(occu) { print(" X ") } else { print(" _ ") }
        }
        print("\n")
    }
    println(name+" "+path(from)+" -> "+path(to))
}

```

```

def view_pass_path(name: String): Unit = {
    // where to look
    var found: Boolean = false; var from,to,count: Int = 0;
    //
    breakable{
        for(row <- 0 to rows-1){ // traverse rows
            breakable{
                for(col <- 0 to cols-1){ // traverse cols
                    breakable{
                        for(x <- 0 to path.length-1){ // traverse the path
                            if(seat(row)(col)(x) == name){
                                found = true; count+=1; to = x+1
                                if(count == 1){ from = x };
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    if(found){break()}
}
}
if(found){break()}
}
}
if(found){break()}
}
println(name+" "+path(from)+" -> "+path(to))
}

```

```

def look_pass(name: String): Boolean = {
    breakable{
        for(x <- pass){
            if(x == name){ return true; break() }
        }
    }
    return false;
}

```

```

def fly_name(): String = {cls_name;} // return flight name

```

```

def fly_path(): ArrayBuffer[String] = {path} // return flight path

```

```

def fly_cap(): Int = {rows*cols} // return flight passenger capacity
}

```