import scala.collection.mutable.Map

import scala.List

import java.util.\_

import java.time.\_

import java.text.SimpleDateFormat

object HelloWorld

{

var busSeat:Int=0

// var slSeat=100

// var ac3Seat=50

// var ac2Seat=50

var ecSeat=60

var bcSeat=30

var fcSeat=30

var tr:Map[Int,List[Any]] = Map(10021 -> List("Shatabdi","12:01","16:25",452,100,50,50),10045->List("Deccan","14:01","18:25",600,50,25,25))

var bus:Map[Int,List[Any]] = Map(2014 -> List("Shivneri","10:01","12:25",500,60),2015->List("Shivshahi","11:01","13:25",400,60))

def main(args: Array[String])

{

do

{

println("Transport Booking App")

println("1:Bus \t 2: Train \t 3: Flight \t 4: Exit")

var choice = readInt()

choice match

{

case 1 => busBooking()

case 2 => trainBooking()

case 3 => flightBooking()

case 4 => System.exit(0)

case \_ => println("Invalid Input")

}

}while(true)

}

def busBooking()

{

val DATE\_FORMAT = "yyyy-MM-dd"

println("Enter date")

var dateOfJourney=readLine()

val dateFormat = new SimpleDateFormat(DATE\_FORMAT)

dateFormat.parse(dateOfJourney)

//dateFormat.format(dateOfJourney)

println("Enter source")

var source=readLine()

println("Enter destination")

var destination=readLine()

println("Available Buses : ")

println("Bus No.\t Bus Name \t Depature Time \t Arrival Time \t Seats Available")

bus.keys.foreach{ i =>

var details=bus(i)

printf("%d\t %s\t %s\t %s\t %d\t",i,details(0),details(1),details(2),details(4))

println()

}

println("Enter bus number")

var busNum=readInt()

var details=bus(busNum)

var busSeat:Int=details(4).asInstanceOf[Int]

var cost:Int=details(3).asInstanceOf[Int]

println("Enter no of passengers")

var n=readInt()

if(busSeat-n<0)

{

println("Insufficient Availability")

}

else

{

//details(4).asInstanceOf[Int]=details(4).asInstanceOf[Int]-n

println("Enter name")

var name=readLine()

println("Enter contact number ")

var contact=readLine()

println("Booking Details:")

println("Source: "+source)

println("Destination: "+destination)

println("Name : "+name)

println("No of passengers :"+ n)

println("Total cost : "+(n\*cost))

println("Seat Number Allocated :")

var s=details(4).asInstanceOf[Int]-n

bus(busNum)=details.updated(4,s)

for(x<-1 to n)

{

println(s+x)

}

}

}

def trainBooking()

{

println("Enter name")

var name=readLine()

println("Enter contact number ")

var contact=readLine()

val DATE\_FORMAT = "yyyy-MM-dd"

println("Enter date")

var dateOfJourney=readLine()

val dateFormat = new SimpleDateFormat(DATE\_FORMAT)

dateFormat.parse(dateOfJourney)

println("Enter source")

var source=readLine()

println("Enter destination")

var destination=readLine()

println("Available Trains : ")

println("Train No.\t Train Name \t Depature Time \t Arrival Time \t Sleeper \t 3Ac \t 2AC")

tr.keys.foreach{ i =>

var details=tr(i)

printf("%d\t %s\t %s\t %s\t %d\t %d\t %d\t",i,details(0),details(1),details(2),details(4),details(5),details(6))

println()

}

println()

println("Enter train no")

var train:Int=readInt()

if(tr.contains( train ))

{

println("Train Coach")

println("1:sleeper\t2:3AC\t3:2AC")

println("Enter your choice")

var c=readInt()

println("Enter no of passengers")

var n=readInt()

var avail=0

var details=tr(train)

c match

{

case 1=> var a:Int=details(4).asInstanceOf[Int];if(a-n>=0){ a=a-n;avail=1;a=a-n;tr(train)=details.updated(4,a) }

case 2=> var a:Int=details(5).asInstanceOf[Int];if(a-n>=0){ a=a-n;avail=1;a=a-n;tr(train)=details.updated(5,a) }

case 3=> var a:Int=details(6).asInstanceOf[Int];if(a-n>=0){ a=a-n;avail=1;a=a-n;tr(train)=details.updated(6,a) }

case \_=>println("Invalid Coach")

}

if(avail==1)

{

println("Booking Details:")

println("Source: "+source)

println("Destination: "+destination)

println("Name : "+name)

println("No of passengers :"+ n)

var cost:Int=details(3).asInstanceOf[Int]

println("Total cost : "+(cost\*n))

println("Seat Number Allocated :")

for(x<-1 to n)

{

if(c==1)

{

var sl:Int=details(4).asInstanceOf[Int]

println("SL"+(sl+x))

}

else if(c==2)

{

var ac3:Int=details(5).asInstanceOf[Int]

println("A3 "+(ac3+x))

}

else

{

var ac2:Int=details(6).asInstanceOf[Int]

println("A3 "+(ac2+x))

}

}

}

else

println("Not Available")

}

else

println("Invalid train number")

}

def flightBooking()

{

println("Enter name")

var name=readLine()

println("Enter contact number ")

var contact=readLine()

val DATE\_FORMAT = "yyyy-MM-dd"

println("Enter date")

var dateOfJourney=readLine()

val dateFormat = new SimpleDateFormat(DATE\_FORMAT)

dateFormat.parse(dateOfJourney)

println("Enter source")

var source=readLine()

println("Enter destination")

var destination=readLine()

println("Available Flights : ")

println("Flight No.\t Flight Name \t Depature Time \t Arrival Time \t Economy Class \t First Class \t Business Class")

printf("SG-703\t SpiceJet \t 12:06 \t 16:00\t %d \t %d \t %d",ecSeat,fcSeat,bcSeat)

println()

println("Enter flight no")

var flight:String=readLine()

if(flight.equals("SG-703"))

{

println("Flight Class")

println("1:Economy Class\t2:Business Class\t3:First Class")

println("Enter your choice")

var c=readInt()

println("Enter no of passengers")

var n=readInt()

var avail=0

c match

{

case 1=> if(ecSeat-n>=0){ ecSeat-=n;avail=1 }

case 2=> if(bcSeat-n>=0){ bcSeat-=n;avail=1 }

case 3=> if(fcSeat-n>=0){ fcSeat-=n; avail=1}

case \_=>println("Invalid Class")

}

if(avail==1)

{

println("Booking Details:")

println("Source: "+source)

println("Destination: "+destination)

println("Name : "+name)

println("No of passengers :"+ n)

println("Total cost : "+(n\*500))

println("Seat Number Allocated :")

for(x<-1 to n)

{

if(c==1)

{

println("EC"+(ecSeat+x))

}

else if(c==2)

println("BC "+(bcSeat+x))

else

println("FC "+(fcSeat+x))

}

}

else

println("Not Available")

}

else

println("Invalid flight number")

}

}