class Location

{

public float lat { get; set; }

public float lon { get; set; }

}

abstract class Person

{

public string Name { get; protected set; }

protected string userName { get; set; }

protected string password { get; set; }

public Location location { get; set; }

protected List<Trip> trips { get; set; }

protected Guid ID { get; set; }

public int Rating { get; set; }

}

abstract class Car

{

public string vin { get; set; }

public CarType type { get; set; }

public string makeModel { get; set; }

}

public enum CarType { Black, Taxi, SUV, Lux};

interface ICustomer

{

bool Login();

void RequestCab(Location start, CarType type);

void RequestCab( Location start, Location end, CarType type);

void Pay(Trip trip);

void RateDriver(Trip trip, int rating);

void CancelRequest(Trip trip);

}interface IDriver

{

CarType carType{ get; set; }

Guid Signup(string Name, CarType carType);

bool Login();

bool AcceptRequest(Trip trip);

void RateCustomer(Trip trip, int rating);

void CancelRequest(Trip trip);

bool occupied { get; set; }

}

Customer class

class Customer : Person, ICustomer

{

public event EventHandler<RequestCabNotificationArgs> requestCabEvent;

private Customer() { }

private string creditCard { get; set; }

public Customer(string Name, string creditCard, string userName, string password)

{

this.Name = Name;

this.creditCard = creditCard;

this.ID = Guid.NewGuid();

this.userName = userName;

this.password = password;

this.Rating = 0;

}

public Guid GetID()

{

return this.ID;

}

public bool Login()

{

return true;

}

public void RequestCab(Location start, CarType type)

{

Trip trip = new Trip(start, DateTime.Now, type);

RequestCabNotificationArgs args = new RequestCabNotificationArgs(trip);

if (requestCabEvent != null)

{

requestCabEvent(this, args);

}

}

public void RequestCab(Location start, Location end, CarType type)

{

Trip trip = new Trip(start, end, DateTime.Now, type);

RequestCabNotificationArgs args = new RequestCabNotificationArgs(trip);

if (requestCabEvent != null)

{

requestCabEvent(this, args);

}

}

public void Pay(Trip trip)

{

}

public void RateDriver(Trip trip, int rating)

{

}

public void CancelRequest(Trip trip)

{

}

public void StartTrip(Trip trip)

{

}

public void EndTrip(Trip trip)

{

}

}

Driver class

class Driver : Person, IDriver

{

private ArrayList listTrips;

public CarType carType { get; set; }

private Driver() { }

public void Subscribe(RequestManager requestManager)

{

requestManager.driverNotifyEvent += requestManager\_driverNotifyEvent;

}

public void Unsubscribe(RequestManager requestManager)

{

requestManager.driverNotifyEvent -= requestManager\_driverNotifyEvent;

}

void requestManager\_driverNotifyEvent(object sender, DriverRequestCabNotificationArgs e)

{

// get the customer and the trip and figure out if you want to trake the trip or not.

listTrips.Add(e.trip);

throw new NotImplementedException();

}

public Driver(string Name, string userName, string password, CarType carType)

{

this.Name = Name;

this.occupied = false;

this.carType = carType;

this.ID = Guid.NewGuid();

this.userName = userName;

this.carType = carType;

this.Rating = 0;

this.listTrips = new ArrayList();

}

public Guid GetID()

{

return this.ID;

}

public Guid Signup(string Name, CarType carType)

{

return Guid.NewGuid();

}

public bool Login()

{

return true;

}

public bool AcceptRequest(Trip trip)

{

return true;

}

public void RateCustomer(Trip trip, int rating)

{

}

public void CancelRequest(Trip trip)

{

}

public bool occupied { get; set; }

public void CheckTrips()

{

if (this.occupied == false)

{

if (listTrips.Count != 0)

{

bool accepted = RequestManager.AcceptRequest(this, (Trip)listTrips[0]);

if (accepted)

{

this.occupied = true;

}

}

}

}

}

class RequestCabNotificationArgs : EventArgs

{

public Trip trip {get; set;}

public RequestCabNotificationArgs( Trip trip)

{

this.trip = trip;

}

}

class DriverRequestCabNotificationArgs : EventArgs

{

public Trip trip { get; set; }

public DriverRequestCabNotificationArgs( Trip trip)

{

this.trip = trip;

}

}

class RequestManager

{

private static object objLock;

private static Dictionary<Guid, Trip> dictTrip;

public static bool AcceptRequest(Driver driver, Trip trip)

{

Trip trp = dictTrip[trip.tripID];

// do double locking here.

lock(objLock)

{

if (trp.tripStatus == TripStatus.Waiting)

{

dictTrip[trip.tripID].tripStatus = TripStatus.InProgress;

Customer cust = (Customer)dictTrip[trip.tripID].customer;

cust.StartTrip(dictTrip[trip.tripID]);

return true;

}

else

{

return false;

}

}

}

public RequestManager()

{

dictTrip = new Dictionary<Guid, Trip>();

objLock = new object();

}

public event EventHandler<DriverRequestCabNotificationArgs> driverNotifyEvent;

public void NotifyDriver( Trip trip)

{

DriverRequestCabNotificationArgs args = new DriverRequestCabNotificationArgs( trip);

if (driverNotifyEvent != null)

{

driverNotifyEvent(this, args);

}

}

public void Subscribe(Customer customer)

{

customer.requestCabEvent += customer\_requestCabEvent;

}

public void Unsubscribe(Customer customer)

{

customer.requestCabEvent -= customer\_requestCabEvent;

}

void customer\_requestCabEvent(object sender, RequestCabNotificationArgs e)

{

// this is the call when the users will call for request,

// now the request manager will call cab drivers and let them know

// there is a request

dictTrip.Add(e.trip.tripID, e.trip);

NotifyDriver(e.trip);

// now raise the event for the

throw new NotImplementedException();

}

}

class Uber

{

Dictionary<Guid, Customer> dictCustomers;

Dictionary<Guid, Driver> dictDrivers;

Thread[] customerThreads = new Thread[20];

Thread[] driverThreads = new Thread[20];

RequestManager requestmanager; public Uber()

{

dictCustomers = new Dictionary<Guid, Customer>();

dictDrivers = new Dictionary<Guid, Driver>();

requestmanager = new RequestManager();

Initialize();

} void CustomerThreadFunction(object obj)

{

Customer cust = (Customer)obj;

while (true)

{

Console.WriteLine(cust.Name);

Thread.Sleep(10000);

}

}

void DriverThreadFunction(object obj)

{

Driver driver = (Driver)obj;

while (true)

{

driver.CheckTrips();

Thread.Sleep(10000);

}

} private void Initialize()

{

// create 20 customers and 10 drivers

for (int i = 0; i < 20; i++)

{

Customer cust = new Customer("Cust" + i.ToString(), "CreditCard" + i.ToString(),

"Cust" + i.ToString(), "Password" + i.ToString());

dictCustomers.Add(cust.GetID(), cust);

customerThreads[i] = new Thread(new ParameterizedThreadStart(CustomerThreadFunction));

customerThreads[i].Start(cust);

requestmanager.Subscribe(cust);

}

for (int i = 0; i < 10; i++)

{

Driver driver = new Driver("Driver" + i.ToString(),"Driver" + i.ToString(), "Password" + i.ToString(),

CarType.Black);

dictDrivers.Add(driver.GetID(), driver);

driverThreads[i] = new Thread(new ParameterizedThreadStart(DriverThreadFunction));

} }

}