Question 1:

Problem : Order a hotel online before a trip

Objects and Behaviors:

Consumer

Data:	name, CellphoneNumber, EmailAddress
Behavior:	ReserveOne, EnterCertainCondition, CompareSeveralHotels, InputCreditCard, ChoosePeriod

Hotel

Data:	HotelName, PriceRange, StarRate, location, FloorPlan, MoveInOutDate, Price
Behavior:	ConfirmReservation, Boolean CreditCard
Data:	CardNumber, CVV, ExpireDate, NameOnCard

CreditCardCompany

Behavior:	MatchTheInformation, AuthorizeTransaction, RejectTransaction

HotelSearchEngine

Data:	HotelList[] hotellist
Behavior:	Boolean MeetWithMatch, PleaseResetCondition, ReturnSearchResults

Sequence of invoking behaviors on Objects:

Consumer consumer;
Hotel hilton
CreditCard mycreditcard
CreditCard mastercard
HotelSearchEngine booking
OrderConfirmation response

```
Consumer.EnterCertainCondition(); ->  PriceRange, StarRate, Location, FloorPlan
if( booking.MeetWithMatch() == True){
    booking.ReturnSearchResults(); -> hotellist
    consumer.CompareSeveralHotels(); -> HotelName
    HotelName = hilton;
    consumer.ChoosePeriod(); -> MoveInOutDate
    if( hilton.CheckAvaliability() == True){
        consumer.ReserveOne(); -> Price, name, FloorPlan, MoveInOutDate;
        consumer.InputCreditCard(); -> CardNumber, CVV, ExpireDate, NameOnCard
        if (mastercard.MatchInformation == True){
            mastercard.AuthorizeTransaction();
            hilton.ConfirmReservation(); -> MoveInOutDate, Price, Location, FloorPlan
            consumer.OrderSuccessfully();
        }else
        {
            mastercard.RejectTransaction();
            consumer.CantOrderAHotel();
        }
    }else
    {
        hilton.NoAvaliableRoom();
        hilton.PleaseResetPeriod();
        consumer.CantOrderAHotel();
    }
}else
{
    booking.PleaseResetCondition();
    consumer.CantOrderAHotel();
}
```

Question 2:

Problem : Design an app for calling taxis

Objects and behaviors:

Traveller:
Data: Destination
Behavior: TypeInDestination; Boolean ConfirmArrived, ChooseCarType, SendMoney, Boolean ConfirmTravelInfo, GetCurrentLocation, CancelTrip

ServiceProvider
Data:
Behavior: SearchNearestDriver, SendRequestToDriver, EndTrip, RequestMoney, NumberOfDrivers, ChooseDriver, ShowTravelSummary

Driver
Data: Name, CarNumber, PhoneNumber
Behavior: AcceptRequestFromTraveller, ConfirmArrived, ConfirmPickUp, GetCurrentLocation

MapProvider:
Data: Location, route, EstimateArrival
Behavior: GetCurrentLocation, SearchDestination, CalculateRoute (Start, End), Navigation

CarProvider:
Data: CarType, CarPrice, MaxCarCapacity
Behavior: ShowCarTypes

Timer:
Data: time
Behavior: TimeKeepRunning, GetTime

Sequence of invoking behaviors on Objects:

```
Traveller Gfamily
ServiceProvider Uber
Driver [] driverlist
Driver Tony
MapProvider GoogleMap
CarProvider UberProvider
Timer timer

GoogleMap.GetCurrentLocation();
Gfamily.TypeInDestination(); -> GoogleMap: Location
GoogleMap.CalculateRoute();
Gfamily.ChooseCarType(); -> CarProvider: CarType
Uber.ShowTravelSummary(); -> UberProvider: MaxCarCapacity, CarPrice; GoogleMap: EstimateArrival
if (Gfamily.ConfirmTravelInfo() is true)
{
    driverlist = Uber.SearchNearestDriver(); -> driverlist
    while (timer.GetTime() :-> time <= 3min)
    {
        if (driverlist.length >= 1){
            Uber.SendRequestToDriver();
            if (Tony.AcceptRequestFromTraveller is true && Gfamily.CancelTrip is False)
            {
                GoogleMap.CalculateRoute(Tony.GetCurrentLocation, Gfamily.GetCurrentLocation);
                GoogleMap.Navigation();
                Tony.ConfirmPickUp();
                GoogleMap.CalculateRoute(Tony.GetCurrentLocation, destination)
                GoogleMap.Navigation()
                if (Gfamily.ConfirmArrived || (Tony.GetCurrentLcation == destination && Tony.ConfirmArrived))
                {
                    Uber.RequestMoney();
                    Gfamily.SendMoney();
                    Uber.EndTrip();
                }
            }elseif( Gfamily.CancelTrip is ture)
                Uber.EndTrip();
            else
                break;
        }else
        {
            timer.TimeKeepRunning(); ->time
            driverlist = Uber.SearchNearestDriver(); -> driverlist
        }
    }
    Uber.EndTrip();
}else
    Uber.EndTrip();
```

Question 3:
Problem : Design a job searching and posting platform

Objects and behaviors:

CompanyHR

Data:	CompanyName, HRName, HREmail
Behavior:	AddNewJobDescription; DeleteJobDescription; EditJobDescription

Job

Data:	JobDescription, Salary, WorkType, WeeklyWorkHours, Prerequisite
Behavior:	IsExpired; NeedToEdit; IsExist; GetHours; GetSalary; GetPrerequisite

Interviewee

Data:	age, skills
behavior:	GetPreferedWorkHour, GetPreferedSalary, GetSkills

JobWebsite
    behavior: ReturnSearchInfo, HRLoginSystem, IntervieweeLogin, UserAccount, LoginError, exit, IsMeetWithRequirement

Sequence of invoking behaviors on Objects:

```
    CompanyHR HRStaff
    Job job
    Job joblist[]
    Interviewee Gavin
    JobWebsite jobweb

    if (Internet.isAvaliable)
    {
        if( jobweb.UserAccount == CompanyHR)
        {
            jobweb.HRLoginSystem(); -> job[]
            if (job.IsExist)
            {
                if (job.IsExpired)
                {
                    HRStaff.DeleteJobDescription();
                }else{
                    HRStaff.EditJobDescription(); -> JobDescription, Salary, WorkType, WeeklyWorkHours, Prerequisite
                }
            }else{
                HRStaff.AddNewJobDescription(); ->JobDescription, Salary, WorkType, WeeklyWorkHours, Prerequisite
            }
            jobweb.exit()
        }elseif ( jobweb.UserAccount == Gavin){
            jobweb.IntervieweeLogin();
            if (job.GetHours() <= Gavin.GetPreferedWorkHours() && job.GetSalary() >= Gavin.GetPreferedSalary())
```

```
                {
                    joblist[] = jobweb.ReturnSearchInfo(); -> JobDescription, Salary, WorkType, WeeklyWorkHours, Prerequisite
                    for (int i = 0, i <= joblist.length-1, I++){
                        if ( jobweb.IsMeetWithRequiremet(job.GetPrerequisite(), Gavin.GetSkills())){
                            jobweb.ReturnMatchedInfo();
                        }else{
                            jobweb.NoMatchedInfo()
                            jobweb.exit()
                        }
                    }
                }else{
                    jobweb.exit()
                }
            }else{
                jobweb.LoginError();
                jobweb.exit()
            }
        }
```

Question 4:
Problem : Order food in a restaurant

Objects and behaviors:

Customer:

Data:	Membership
Behavior:	KeepWaiting, AskToOrder, Order, ConfirmOrder

Menu:

Data:	FoodList, Style,
Behavior:	ListAllTheFood, ChooseTaste

Waiter:
Data: Name
Behavior: IsAvaliable, ResponseToRequest

OrderService:
Data:
Behavior: AddToCart, DeleteItem, ChangeAmount, Finished

FoodCart:
Data: Food, Taste
Behavior: ShowSummary

Sequence of invoking behaviors on Objects:

```
Customer customer;
Menu menu;
Waiter waiter;
OrderService orderservice;
ShoppingCart cart

if (waiter.IsAvaliable){
    customer.AskToOrder();          ->   Waiter: Name
    waiter.ResponseToRequest();
    FoodList[] = menu.ListAllTheFood();
    for(int i = 0; i <= Food.length-1; i ++){
        if (customer.Order(Food[i]))
            orderservice.AddToCart(Food[i]);
    }
    menu.ChooseTaste(); -> Taste
    cart.Show();             -> cart: Food, Taste
    if (customer.ConfirmOrder(cart) == true){
```

```
                orderservice.Finished();
            }else{
                customer.KeepLooking();
            }
        }else{
            customer.KeepWaiting();
        }
```

Question 5:
Problem : Design a course registration platform

Objects and behaviors:

Student:
Data: StudentID, Password, Credit, Major
Behavior: LoginToServer, GetCurrentCredit, GetMajor, AddToSchedule, DeteleCourse, ToRegistCourseList, GetPreviousCourse

SchoolServer:
Data:
Behavior: AuthorizeLogin, RejectLogin, Validate(StudentID, Password)

Course:
Data: Credit, Prerequisite, Major, Capacity, Name, CurrentEnrollNumber
Behavior: GetCourseInfo, Search, GetTimeInfo

CoursePlatform:
Data: MaxCredit
Behavior: GetTermInfo, GetCurrentRegistrationNumber, ConfirmRegisteration, AddCredit, RegisterationFailed

Sequence of invoking behaviors on Objects:

```
        Student Gavin
        SchoolServer NEU
        Course course
        CoursePlatform NEUCourse

        course_list [] = Gavin.ToRegistCourseList();  -> course: Name[]
if (internet.IsAvaliabe){
        StudentID, Password = Gavin.LoginToServer();  -> StudentID, Password
        if (NEU.Validate( StudentID, Password) is true){
                NEU.GetTermInfo();  -> MaxCredit
                currentCredit = Gavin.GetCurrentCredit();  -> int
                int CreditToAdd = 0;
                for (int j = 0; j <= course_list.length -1; j++){
                        int CreditToAdd = CreditToAdd + course_list[j].GetCredit();
                }
                PreviousCourse = Gavin.GetPreviousCourse();  -> Course
                if ( (CreditToAdd + currentCredit) <= MaxCredit){
                        for (int i = 0; i <= course_list.length -1; i++){
                                course_list[i].GetCourseInfo();   ->  Credit, Prerequisite, Major, MaxCapacity, Name, CurrentEnrollNumber
                                if (PreviousCourse.IsHave(Prerequisite) && CurrentEnrollNumber + 1 <= MaxCapacity &&
Gavin.HasNoClass(course_list[i].GetTimeInfo))
                                        NEUCourse.ConfirmRegisteration();
                                else
                                        NEUCourse.RegisterationFailed();
                        }
                }else{
                        NEUCourse.RegisterationFailed();
                }
        }else{
                NEU.RejectLogin();  ->MsgInfo
        }
}
```