INFO 5100 Assignment 1

```
INFO 5100 Assignment 1
Xiaoxi Wang
wang.xiaoxi1@husky.neu.edu



_____1. Order a hotel online before a trip_____

Object:

    Traveler
        Data: name, phone, email
        Behaviors: search, compare, bookHotelRoom, cancel

    Internet
        Data: priceline, holidayInn, marriott
        Behaviors: isAvailable, searchForHotelWebsite

    HotelWebsite
        Data: hotelRoom
        Behaviors: search, sort, display

    HotelRoom
        Data: date, address, distance, price, roomType
        Behaviors: isAvailable

    CreditCard
        Data: cardNumber, bankCompany, holdersName, expireDate, securityCode, billingAddres
s

    BankCompany
        Behaviors: authorizeTransaction

BookHotelOnline:

    Traveler xiaoxi;
    Internet internet;
    HotelWebsite priceline;
    HotelRoom[] hotelRooms;
    CreditCard creditCard;
    BankCompany chase;
    BookingConfirmation response;

    if (internet.isAvailable) {
        // get a list of available hotel rooms
        HotelRoom[] hotelRooms = xiaoxi.search(internet.searchForHotelWebsite(priceline));
        for (HotelRoom hotelRoom : hotelRooms) {
            if (hotelRoom.isAvailable(date, roomType)) {
                xiaoxisList.add(hotelRoom);
            }
```

```
        }

        // select the best room considering price and distance
        sort(xiaoxisList(price));
        HotelRoom myRoom1 = xiaoxisList[0];
        sort(xiaoxisList(distance));
        HotelRoom myRoom2 = xiaoxisList[0];
        myRoom = xiaoxi.compare(myRoom1, myRoom2);

        // booking and payment
        xiaoxi.bookHotelRoom(myRoom);
        priceline.jumpTo(paymentPage);
        xiaoxi.input(cardNumber, holdersName, expireDate, securityCode, billingAddress);
        if (creditCard.bankCompay(authorizeTransaction())) {
            payment_SUCCEEDED;
            BookingConfirmation response;
            priceline.sendConfirmationEmail(xiaoxi);
        } else {
            payment_FAILED;
            backTo(priceline);
        }

        // cancel the booking
        if (exception) {
            xiaoxi.cancel();
            bankCompany.recallTransaction();
            priceline.sendCancellationEmail(xiaoxi);
    } else {
        xiaoxi.searchLater();
    }



_____2. Design an app for calling taxis (e.g. Uber)_____

Object:

    Passenger
        Data: location, destination, phoneNumber, paymentMethod
        Behaviors: login, search, bookTaxi, call, cancel, trip, makePayment

    Driver
        Data: location, phoneNumber, vehicleInfo
        Behaviors: receiveOrder, drive, pickup, receiveMoney, call

    Application
        Data: PassengerInfo, DriverInfo, paymentMethod
        Behaviors: searchDriver, sort, getInfo, display, connectTrafficSystem

    TrafficSystem
        Data: location, route
        Behaviors: getLocation, calculateTime, getRoute

    BankCompany
        Behaviors: authorizeTransaction

CallingTaxis:
```

```
107
108     Passenger passenger;
109     Driver[] driverList;
110     Application app;
111     TrafficSystem trafficSystem;
112
113     if (internet.isAvailable()) {
114         passenger.login();
115         passenger.search();
116         passenger.bookTaxi();
117         if (exception) {
118             passenger.cancel();
119         }
120         passenger.trip();
121         passenger.makePayment();
122     } else {
123         passenger.loginLater();
124     }
125
126     // passenger search a taxi
127     search() {
128         passengersLocation = trafficSystem.getLocation(passenger);
129         passenger.input(destination);
130         Driver[] driverList = app.searchDriver(passengersLocation);
131         for (Driver driver : driverList) {
132             if (!driver.isAvailable()) {
133                 driverList.delete(driver);
134                 continue;
135             }
136             driversLocation = trafficSystem.getLocation(driver);
137             requiredTime = trafficSystem.calculateTime(driversLocation, passengersLocation)
;
138         }
139         app.sort(driverList); // according to time required
140         Driver driver = driverList[0];
141         driversVehicleInfo = app.getInfo(driver);
142         app.display(requiredTime, driversVehicleInfo);
143     }
144
145     // passenger book a taxi
146     bookTaxi() {
147         if (passenger.select(driver)) {
148             driver.receiveOrder(passenger);
149             driversRoute = trafficSystem.getRoute(driversLocation, passengersLocation);
150             driver.drive(driversRoute);
151         } else {
152             passenger.search(anotherTaxi);
153         }
154
155     // passenger on the trip
156     trip() {
157         if (driver.arriveAt(passengersLocation) && driver.pickup(passenger)) {
158                 driver.isAvailable_NO;
159                 app.display(trip_BEGIN);
160                 driversRoute = trafficSystem.getRoute(passengersLocation, destination);
161                 driver.drive(driversRoute);
162                 app.display(trip_FINISH);
```

```
            } else {
                passenger.call(app.getInfo(driver).phoneNumber);
                driver.call(app.getInfo(passenger).phoneNumber);
            }

    // passenger make a payment
    makePayment() {
        app.jumpTo(paymentPage);
        if (app.paymentMethod) {
            passenger.input(securityCode);
        } else {
            passenger.input(cardNumber, holdersName, expireDate, securityCode, billingAddre
ss);
        }
        if (creditCard.bankCompay(authorizeTransaction())) {
            payment_SUCCEEDED;
            app.sendPaymentSummaryEmail(xiaoxi);
        } else {
            payment_FAILED;
            backTo(makePayment);
        }
    }



_____3. Design a job searching and posting platform_____

Object:

    User
        Data: jobSearcher, recruiter, name, emailAddress, password, location, industry, job
Title, company
        Behaviors: createAccount, setupProfile, login, searchJobs, contact, editProfile, po
stJob, deletePostedJob

    Platform
        Data: accounts
        Behaviors: addAccount, getInformation, getJobs, updateProfile, sort, match

JobSearchingAndPostingPlatform:

    User user;
    User jobSearcher;
    User recruiter;
    Platform platform

    if (internet.isAvailable()) {
        if (user == jobSearcher) {
            if (jobSearcher.isNewUser()) {
                jobSearcher.createAccount();
                jobSearcher.setupProfile();
            } else {
                jobSearcher.login();
            }
            jobSearcher.searchJob();
            jobSearcher.contact(user);
            jobSearcher.editProfile();
```

```
        }
        if (user == recruiter) {
            if (recruiter.isNewUser()) {
                recruiter.createAccount();
                recruiter.setupProfile();
            } else {
                recruiter.login();
            }
            recruiter.postJob();
            recruiter.contact(user);
            recruiter.editProfile();
            recruiter.deletePostedJob();
        }
    } else {
        user.comBackLater();
    }

    createAccount() {
        input(userName, emailAddress, password);
        if (verify(email)) {
            create_SUCCEEDED;
            platform.addAccount();
        } else {
            create_FAILED;
        }
    }

    setUpProfile() {
        user.input(name, location, industry, jobTitle, company, education, workExperience,
skills, contactEmail);
        if (resume) {
            upload(resume);
        }
        if (photo) {
            upload(photo);
        }
    }

    searchJobs() {
        input(location, industry, searchingJobTitle);
        Job[] jobs = platform.getJobs(location, industry, searchingJobTitle);
        platform.sort(jobs);
    }

    contact() {
        contactEmail = platform.getContactEmail(searchingUser);
        user.email(contactEmail);
    }

    editProfile() {
        profile = platform.getInformation(user);
        user.edit();
        user.save();
        platform.updateProfile();
    }

    postJob() {
```

```
        recruiter.input(location, industry, postedJobTitle, salaryRange, requirements);
        if (attachment) {
            recruiter.upload(attachment);
        }
    }

    deletePostedJob() {
        Jobs[] postedJobs = platform.getPostedJobs(recruiter);
        postedJobs.get(deleteJob).delete();
    }



_____4. Order food in a restaurant_____

Object:

    Customer
        Data: waitingTime
        Behaviors: getMenu, select, order, leave, pick, makePayment

    Restaurant
        Data: menu
        Behaviors: prepareDish, receiverPayment

    Menu
        Data: dish, prepareTime, price

    Payment
        Data:
        Behaviors:

OrderFoodInRestaurant

    Customer customer;
    Restaurant restaurant;
    Menu[] menu;
    Payment payment;

    Menu[] menu = customer.getMenu(restaurant);
    for (Menu dish : menu) {
        if (customer.select(dish)) {
            orderList.add(dish);
        }
    }
    customer.order(orderList);
    for (Menu dish : orderList) {
        if (dish.isAvailable() && dish.prepareTime <= customer.waitingTime) {
            restaurant.prepare(dish);
            payment += dish.price;
        } else if (customer.keepOrder()) { // customer order without this dish
            orderList.delete(dish);
            continue;
        } else { // customer cannot order without this dish
            break;
            customer.leave();
        }
```

```
    }
    if (orderList.isReady()) {
        customer.pick(orderList);
        customer.makePayment();
    }



_____5. Design a course registration platform_____

Object

    Student
        Data: userName, password, program, semester, maxCreditLimit, minCreditLimit
        Behaviors: login, getCourseList, hasTakenCourse, checkSchedule, add, sort, register
Course, dropCourse

    Course
        Data: prerequisite, time, credit, seat, waitlist

CourseRegistrationPlatform

    Student student;
    Course[] courseList;
    Course[] dropList;

    if (internet.isAvailable()) {
        student.login(userName, password);
        Course[] courseList = student.getCourseList(program, semester);

        // add available courses to studentsList
        for (Course course : courseList) {
            if (student.hasTakenCourse(course) || !student.hasTakenCourse(course.prerequisi
te)) {
                continue;
            }
            if (!student.checkSchedule(course.time)) { // if the course time doesn't fit
                continue;
            }
            studentsList.add(course);
        }

        // register courses
        student.sort(studentsList);
        for (Course course : studentList) {
            creditSum += course.credit;
            if (creditSum > maxCreditLimit) {
                creditSum -= course.credit;
                break;
            }
            student.registerCourse(course);
            if (course.seat >= 0) {
                course.seat--;
            } else {
                course.waitlist++;
            }
            student.checkSchedule(course.time) = 1;
```

```
385         }
386
387         // drop courses
388         for (Course course : dropList) {
389             creditSum -= course.credit;
390             if (creditSum <= minCreditLimit) {
391                 creditSum += course.credit;
392                 break;
393             }
394             student.dropCourse(course);
395             if (course.waitlist > 0) {
396                 course.waitlist--;
397             } else {
398                 course.seat++;
399             }
400             student.checkSchedule(course.time) = 0;
401         }
402     } else {
403         student.loginLater();
404     }
405
406
407
```