1. Design a course management system (Like Canvas);

// While there are plenty of fancy functions of Canvas,
// I mainly concentrate on the usage of turning homework ,
// releasing grades and publishing announcements.

- Student:
Data: loginCredentials, name, registeredCourses
Behavior: login, logout, chooseClass, finishAssignment, turnInAssignment, readAnnouncement, checkGrades

- Instructor:
Data: loginCredentials, name, teachingCourseList,
Behavior: publishAnnouncement, releaseAssignment

- TeachingAssistant:
Data: loginCredentials, name, studentList
Behavior: correctingAssignment, releaseGrades

- Assignments:
Data: questions, requirements

-Course:
Data: courseName

Instructor siva,
Course info5100,
Student phoebe,
Assignments a1,
TeachingAssistant ta,

siva.login(siva.loginCredentials)
//after siva login, all the courses he taught of current semester should pop up.
Array courses = siva.teachingCourseList;
Info5100 = siva.chooseCourse(courses)
siva.publishAnnouncement("Welcome come to Info5100...");

phoebe.login(phoebe.loginCredentials);
Array courses = phoebe.registeredCourses;
phoebe.chooseClass(info5100);
phoebe.readAnnouncement("Welcome come to Info5100...");

siva.releaseAssignment(a1);
a1_finished = phoebe.finishAssignment(a1_finished);
phoebe.turnInAssignment(a1_finished);

ta.login(ta.loginCredentials);

```
Array students = ta.studentList;
for (student in students) {
        ta.correctingAssignment( student_a1);
        ta.releaseGrades(student_a1);
}

phoebe.checkGrades(a1_finished);
phoebe.logout();
siva.logout();
ta.logout()
```

2. Design an app to book airline ticket.

- Customer:
Data: emailAddress, legalName, legalId, departureDate, oneWay(boolean), returnDate, departure, destination, bankAccount
Behavior: login, logout, searchFlights, bookFlights, cancelOrder, requestRefund

- Carrier：
Data: airRoute
Behavior: cancel, countAvailableSeats, sendReceipt

- AirRoute:
Data: flightId, carrier, destination, departure, flightTime, price, availableSeatsNum

```
Customer phoebe,
phoebe.login(loginCredentials);
AirRoute departure_airRoute = phoebe.searchFlights(phoebe.departureDate,
                        phoebe.departure, phoebe.destination)

AirRoute return_airRoute = null;
if(!phoebe.oneWay) {
   return_airRoute = phoebe.searchFlights(phoebe.returnDate,
                        phoebe.destination, phoebe.departure)
}

// for now, just consider phoebe will book a one-way ticket.
phoebe.bookTickets(phoebe.legalName, phoebe.legalId, phoebe.bankAccount,
    departure_airRoute);
Carrier departure_C = departure_airRoute.carrier;
Int seatNum = departure_C.countAvailableSeats(departure_airRoute.flightId);
if(seatNum > 0 && phoebe.bankAccount > departure_airRoute.price) {
        //booking successful;
        departure_C.sendRecipt(phoebe.email)
} else {
        System.out.println("Error! Cannot to book the current flight");
```

```
            return;
}

   if(departure_C.cancel(departure_C.flightId)) {
     departure_C.refund(departure_C.price, phoebe.bankAccount)
   }

   if(phoebe changes mind) {
     phoebe.cancel(departure_C.flightId);
     phoebe.requestRefund(departure_C, departure_C.price)
   }
 phoebe.logout();
```

3. Design a pet adoption platform

```
// Assume all the animals available for adoption are from shelter.
- Customer:
Data: loginCredentials, email, address, petSpecies, adoptionRelatedDocuments
Behavior: searchPets, requestInformation, scheduleMeeting, requestAdopt

- Pet:
Data: species, sex, age, shelterAddress

- Shelter:
Data: address
Behavior: offerInformation, checkPaperWorks, refuseAdoptRequest


Customer phoebe;
Pet desirePet = phoebe.searchPets(species, age, sex, phoebe.address);
Shelter s1 = desirePet.shelteAddressr;
phoebe.requestInformation(desirePet, s1);
phoebe.scheduleMeeting(s1);
if (Phoebe satisfy with desirePet) {
            phoebe.requestAdopt(phoebe.adoptionRelatedDocuments);
            //after check all the documentations, return true if Phoebe is qualified for adopting the pet
            //otherwise return false;
            Boolean decision = shelter.checkPaperWorks(phoebe.adoptionRelatedDocuments);
            if (decision) {
                        phoebe.pickup(desirePet, s1. address);
            } else {
                        shelter.refuseAdoptRequest()
            }
} else {
            //repeat the action, search for another pet.
            phoebe.searchPets(species, age, sex, location);
}
```

4. Design a course registration platform.

- Student:
Data: student_Id, name, loginCredentials, program, transcript, registeredSchedule
Behavior: login, registerSection, planSchedule, checkSectionTime, checkProgramCourses, logout

-Program:
Data: programName, coreCourses

- Course:
Data: sectionList, prerequisites, credits, courseID
Behavior: checkPrerequisites, presentAllSections, unableToRegister

-Sections:
Data: sectionID, courseID, instructor, time, date, availableSeatsNum;
Behavior: countAvailableSeats, checkAvailableTime, registerSuccess, registerFail

```
Student phoebe,
phoebe.login(loginCredentials);
Array courseList  = phoebe.checkProgramCourses(phoebe.program)
Course selectedCourse = phoebe.planSchedule(courseList);

//return true if student fulfills all the prerequisites; Otherwise, return false.
Boolean preRe = selectedCourse.checkPrerequisites(phoebe.transcript);
if(preRe) {
  Array sections = selectedCourse.presentAllSections(courseID);
  Section chosenSection = sections.forEach((sec) ->
       { phoebe.checkSectionTime(sec.time, sec.date, sec.availableSeatsNum)}
      );
  phoebe.registerSection(chosenSection.sectionID, phoebe.student_Id);
  Int availableSeatsNum = chosenSection.countAvailableSeats();
 // return true if there's a time conflict with current schedule
  Boolean timeConflict = chosenSection.checkAvailableTime(phoebe.registeredSchedule)
  if(availableSeatsNum  > 0 && !timeConflict) {
    chosenSection.registerSuccess(phoebe.student_Id);
    chosenSection.availableSeatsNum--;
    phoebe.registeredSchedule.add(chosenSection);
  } else {
    chosenSection.registerFail();
  }
} else {
   selectedCourse.unableToRegister(phoebe.student_Id);
}
```

5. Order food in a food delivery app.(Like Uber Eats)

- Customer:
Data: emailAddress, telePhone, name, loginCredentials, Address, creditCard,
favoirteRestaurantList
Behavior: login, orderFood, writeReview, searchRestaurant, cancelOrder

- Restaurant:
Data: name, menu, price, order_Id, address
Behavior: checkOut, sendToShipper, sendReceipt, notifyFoodReady,
giveOrderToDriver, prepareFood,
generateOrder, cancelOrder

- Order:
Data: order_Id, entries, customer.name

- Driver:
Data: Name, telePhone,
Behavior: searchAvailableOrders, chooseOrders, deliveryOrder, pickUpfromRestaurant,
contactCustomer

```
Customer phoebe,
Restaurant chipotle = phoebe.searchRestaurant(phoebe.favoriteRestaurantList);
Array entries = phoebe.orderFood(chipotle.menu);
phoebe.checkOut(entries, phoebe.creditCard);
Order phoebeOrder = chipotle.generateOrder(order_Id, phoebe.name, entries);
chipotle.sendReceipt(phoebe.emailAddress, phoebeOrder);

if(phoebe.cancelOrder(phoebeOrder)) {
   chipotle.cancelOrder(phoebeOrder.order_Id);
   chipotle.refund(phoebe.creditCard, phoebeOrder.order_Id);
   return;
}

chipotle.prepareFood(entries);
Driver zico = Driver;
Array orders = zico.searchAvailableOrders();
phoebeOrder = zico.chooseOrders(orders);

chipotle.notifyFoodReady(zico, phoebeOrder.order_Id);
zico.pickUpfromRestaurant(phoebeOrder.order_Id, chipotle.name, chipotle.address);
zico.deliveryOrder(phoebe.address, entries);
zico.contactCustomer(phoebe.tel, "I've arrived");

if(phoebe satisfied with driver and restaurant) {
  phoebe.writeReview("good experience, worth to try!")
```

```
}
else {
  phoebe.writeReview("terrible restaurant and driver. Never waste your money!!")
}
```