1.Design a course management system(Like Canvas)

Student:

- Data: Student id, student email address, password, grade, computer, course
- Behavior: doHomework, takeExam

Professor:

- Data: Professor id, professor email address, password, computer
- Behavior: giveGrade, postComment

Courses:

- Data: Course name, course id, virtualOrOnGround(true if virtual), professorThatTeach, hasSeat, waitingList
- Behavior: addWaitingList

Computer:

- Data: Brand, operatingSystem
- Behavior: logInToCanvas, registerCourse, dropCourse

Sequence of invoking behaviors on objects:

Computer simon's = new Computer(brand, operatingSystem);
Computer siva's = new Computer(brand, operatingSystem);
Professor siva = new Professor(professorId, password, emailAddress, computer);
Student simon = new Student(studentId, password, emailAddress, computer);
Course INFO5100 = new Course(courseName, courseId, virtualOrOnGround, professorThatTeach, hasSeat, waitingList);
simon.computer.logIn(simon.studentId, simon.password);
if Course INFO5100.hasSeat
    simon.computer.registerCourse(Course INFO5100);
else
    INFO5100.addWatingList(simon);
if simon registered INFO5100
    simon.doHomework(INFO5100, homework);
    siva.giveGrade(simon, INFO5100, homework);
    siva.postComment(simon, INFO5100, homework);
    if there is a test
        simon.takeExam(INFO5100, test);
        siva.giveGrade(simon, INFO5100, test);
        siva.postComment(simon, INFO5100, test);
if simon registered INFO5100 but wants to drop
    simon.computer.dropCourse(INFO5100);

2. Design a pet adoption platform

Pet:

- Data: gender, age, breed
- Behavior: makeASound

Customer:

- Data: email, name, password, address
- Behavior: adopt, search, giveOutPet

Platform:

- Data: petList
- Behavior: confirmAdoption, receivePet

Sequence of invoking behaviors on objects:

```
Customer simon;
Platform petShop;
Pet cat = simon.search(gender, age, cat);
if petShop.petList has cat
        simon.adopt(cat);
        petShop.comfirmAdoption(simon.email, simon.name, simon.address);
        petShop.petList.remove(cat);
if simon has a pet and want to give it to the platform
        simon.giveOutPet(cat);
        petShop.receivePet(cat);
        petShop.petList.add(cat);
```

3. Design an app to book airline tickets

Customer:

- Data: name, email, phone
- Behavior: search, book, requestCancel, requestChangeDate, writeReview

App:

- Data: flightList
- Behavior: checkOut, cancelOrder, sendConfirmation, refund, updateWithAirline

Flight:

- Data: date, time, departureFrom, destination, airline, isFull
- Behavior:

Sequence of invoking behaviors on objects:

```
Customer simon;
App ticketMaster;
Flight flight = simon.search(date, time, departureFrom, destination, airline);
if ticketMaster.filighList has flight && !flight.isFull()
        simon.book(flight);
        ticketMaster.checkOut(simon, flight);
        if check out success
                ticketMaster.sendConfirmation(simon, flight);
if simon wants to cancel tickets
        simon.requestCancel(flight);
        ticketMaster.cancelOrder(simon, flight);
        ticketMaster.refund(simon, flight);
        ticketMaster.sendConfirmation(simon, flight);
else
        ticketMaster.updateWithAirline(simon, flight);
if simon wants to change date
        Flight fight = simon.search(date, time, airline)
        simon.requestChangeDate(flight);
        ticketMaster.updateWithAirline(simon, flight);
        ticketMaster.sendConfirmation(simon, flight);
```

4. Design a course registration platform

Student:

- Data: studentId, student email, password, name, phone, computer, course
- Behavior: registerClass, dropClass, search, login

Professor:

- Data: professorId, email, password, name, phone, computer, course
- Behavior: postClass, login

PlatForm:

- Data: allCourses
- Behavior:

Course:

- Data: Course id, course name, virtualOrOnGround, classCapacity, hasSeat, studentList, waitingList
- Behavior:

Sequence of invoking behaviors on objects:

Porfessor siva;
Student simon;
PlatForm canvas;
siva.login(professorId, password);
Course INFO5100 = siva.postClass(courseId, courseName, virtualOrOnGround, classCapacity);
if simon wants to register INFO5100
    simon.login(studentId, password);
    simon.search(INFO5100);
    if (INFO5100.hasSeat())
        simon.registerClass(INFO5100);
    else
        INFO5100.waitingList(simon);
if simon wants to drop INFO 5100
    simon.login(studentid, password);
    simon.dropClass(INFO5100);

5. Order food in a food delivery app.

Customer:

- Data: email, password, phone number, customerLocation, credit card
- Behavior: search, placeOrder, writeReview, requestRefund, orderAgain, makePayment, pickup order, login

Driver:

- Data: currentLocation, phone, email, password
- Behavior: driveToLocation, pickup order, inform customer

App:

- Data: restaurants
- Behavior: receiveOrder, informDriver, completeOrder, makeRefund

Restaurant:

- Data: food
- Behavior:

Order:

- Data: destination, restaurant, food, restaurant address, arriving time
- Behavior:

Sequence of invoking behaviors on objects:

Customer simon;
App app;
Restaurant restaurant;
simon.login(phoneNumber, password);
simon.search(restaurant, food);
Order simonOrder = simon.placeOrder(restaurant, food, restaurantAddress, customerLocation);
simon.makePayment(app, creditcard);
app.receiveOrder(simonOrder);
Driver siva = app.informDriver(simonOrder);
siva.driveToLocation(simonOrder.restaurantAddress);
siva.pickUpOrder(simonOrder);
siva.driveToLocation(simonOrder.destination);
siva.informCustomer(simon);
simon.pickupOrder(siva);
app.completeOrder(simonOrder);
if simon wants refund because the food is not as expected
    simon.requestRefund(“food not as expected”, simonOrder);
    app.makeRefund(simon);
if simon wants to write a review
    simon.wirteReview(“it is good”, simonOrder);
If simon wants to reorder it
    simon.orderAgain(simonOrder);