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

(1) objects (with data and behaviors)

Student:

Data: Name; StudentID； LoginCredentials; Account

Behaviors: LogIn; Submit; Check; ReportITIssue; Select; Contact

Professor:

Data: Name; FacultyID; LoginCredentials; Email

Behaviors: LogIn; CreateCourse; Update

CourseContent:

Data: CourseNumber, Assignments; Announcements; Grade; Syllabus; ZoomMeetings

Behaviors: BeAvailable; gradeAllWork

IT Support:

Data: StaffName; StaffID; IssueID; IssueDescription

Behaviors: FixIssue

(2) sequence of invoking behaviors on objects.

```
Student ss;
Professor pp;
IT Support ii;
CourseContent info5100;

ss.login(ss.LoginCredentials);
pp.login(pp.LoginCredentials);
pp.createCourse (info5100);
info5100.beAvailable;

pp.update (info5100.announcements, info5100.syllabus, info5100.zoomMeetings);
ss.select(info5100);
if ss has IT issue;
   ss.reportITIssue(ii);
   ii.fixissue(ss.acount);
else
ss.check(info5100.announcements, info5100.syllabus, info5100.zoomMeetings);

if ss need finish new assignments
pp.update(info5100.assignment);
ss.check(info5100.assignment);
   if ss has problems about the assignemt
      ss.contact(pp.email);
```

```
   else
ss.submit(info5100.assignment, ss.account);
pp.update(info5100.grade, ss.account);
ss.check(info5100.grade, ss.acount);
   if ss has problems about the grade
      ss.contact(pp.email);
         if get approved
            pp.update(info5100, ss.account);
         else nothing updates
     else
info5100.gradeAllwork(info5100.grade, ss.acount);
```

2. Design a pet adoption platform.

(1) objects (with data and behaviors)

Adopter:

Data: Name; LoginCredentials

Behaviors: LogIn; ChoosePet; ContactCompany; Adopt; ProvideInformation

Company:

Data: Name

Behaviors: Update; Serve; ProcessAdoption; TakeBack

Pet:

Data: Name; PetID; Species; Color; Breed; Size; Character; Age; Pictures; Gender; Healthy

Behaviors: BeingAdopted; BeingAvailable

(2) sequence of invoking behaviors on objects.

```
Adopter aa;
Company cc;

aa.LogIn(LoginCredentials);
Pet pp = aa.ChoosePet(species, breed, color, age, gender, size, character);

if pp isAvailable
     aa.contactCompany(cc);
     cc.serve(aa);

   if aa decides adoption
        aa.provideInformation;
        if cc deny adoption;
            pp.beingAvailable;
```

```
        else cc approve adoption;
                cc.processAdoption(aa, pp);
                aa.adopt(pp);
                cc.update(pp);
                pp.beingAdopted;

                if aa has problems
                    aa.ContactCompany(cc);
                    cc.Serve(aa);
                    if aa stops adoption
                    cc.takeBack(pp);
                    cc.update(pp);
                    pp.beingAvailable;
                    else pp.beingAdopted;

else pp is not available;
```

3. Design an app to book airline ticket.

(1) objects (with data and behaviors)

Customer:

Data: Name; CustomerID； Account; LoginCredentials; Payment

Behaviors: LogIn; BookTicket; Search; WriteReview; RequestCancel; WaitFlight; ContactCompany

Company:

Data: Name;

Behaviors: SendReceipt; CheckOut; Refund; CustomerServe; Update; NotifyCustomers

Ticket:

Data: FlightNumber; FlightTime; Terminal; BaggageRequirement

Behaviors:

(2) sequence of invoking behaviors on objects.

```
Customer cus;
Company com;

cus.logIn(LoginCredentials);
Ticket ticket= cus.search(FlightNumber; FlightTime; Terminal; BaggageRequirement)

if ticket isAvailable
   if cus has problems
```

```
        cus.contactConpany(com);
        com.customerServe(cus);
    else
   cus.bookTicket(ticket);
   com.checkOut(cus.payment, cus.account);
   com.sendReceipt(com.account);
   cus.waitFlight;
   cus.writeReview("####");

   if cus wants to cancle
        cus.requestCancleOrder(ticket);
        com.refund(ticket, cus);
        cus.writeReview("****");
   else
      if ticket information change
      com.update(ticket);
      com.notifyCustomers(cus);
         if cus wants to cancle
              cus.requestCancleOrder(ticket);
              com.refund(ticket, cus);
              cus.writeReview("****");
         else
               cus.waitFlight
               cus.writeReview("####");
      else
         cus.waitFlight;
         cus.writeReview("####");
else ticket is notAvailable;
```

4. Design a course registration platform.

(1) objects (with data and behaviors)

Student:

Data: Name; StudentID； Account; LoginCredentials; Email

Behaviors: LogIn; Search; Register; WaitInList; ChooseSemester; Drop; Contact

Professor:

Data: Name; FacultyID; LoginCredentials; Email

Behaviors: LogIn; PostCourses; UpdateStudent

Course:

Data: CourseNumber; Description; Requirement; Schedule; Section

Behaviors: UpdateRegister; UpdateWaitingList; IsnotAvailable;

Advisor

Data: Name; StaffID; Email

Behaviors: SolveProblem

(2) sequence of invoking behaviors on objects.

```
Student ss;
Professor pp;
Advisor aa;

pp.login(pp.LoginCredentials);
pp.postCourses;
ss.login(LoginCredentials);
ss.chooseSemester;
Course cc=ss.search(courseNumber, section);

if cc is not full
   if ss decides to register
      if ss has problems
           ss.contact(aa.email);
           aa.SolveProblem(ss.email);
     ss.register(cc);
     cc.updateRegister(ss);
     if ss want to drop cc
          ss.drop(cc);
          cc.UpdateRegister(cc);

else cc is full
   if waitinglist is full
         if ss still wants to register
            ss.contact(pp.email);
            if pp approved
                 pp.updateStudent(ss);
                 ss.register(cc);
                 cc.updateRegister(ss);
            else cc.isnotAvailable;

   else waitinglist is not full
      if ss want to wait in the list
      ss. WaitInList(cc);
      cc.updateWaitingList(ss);
          if ss is able to register after waiting
              if ss wants to register
```

```
        ss.register(cc);
        cc.updateRegister(ss);
        else ss gives up registering
        ss.drop(cc);
        cc.UpdateWaitingList(cc);
    else ss can't register even after waiting
        if ss still wants to register
            ss.contact(pp.email);
            if pp approved
                pp.updateStudent(ss);
                ss.register(cc);
                cc.updateRegister(ss);
             else
                cc.isnotAvailable;
                ss.drop(cc);
                cc.UpdateWaitingList(cc);
```

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

(1) objects (with data and behaviors)

Customer:

Data: Name; CustomerID； Account; LoginCredentials; Address; Phone; Payment

Behaviors: LogIn; Search; Order; WriteReview; Contact; RequestRefund; GetFood

App:

Data: Name;

Behaviors: SendReceipt; CheckOut; Refund; ServeCustomer; ContactRestaurant

Courier:

Data: Name; StaffId

Behaviors: Delivery; ContactCustomer; ContactRestaurant; ReceiveOrder

Restaurant:

Data: Name;

Behaviors: PrepareFood; ContactCourier

Order

Data: OrderNumber; OrderFood; Restaurant;

Behaviors: isDone

(2) sequence of invoking behaviors on objects.

```
Customer cus;
App app;
Restaurant res;
Courier cour;

cus.login(loginCredentials);
Order neworder = cus.search(oderFood, restaurant);

if orderFood is available
  cus.order(neworder);
  app.checkout(cus.address, cus.phone, cus.payment);
  app.sendReceipt(cus.account);
  app.contactRestaurant(res);
  res.prepareFood;

  if cus has problems
     cus.contact(app);
     app.serveCustomer(cus);
     app.contactRestaurant(res);
     if cus is not satisfied
     cus.requestRefund(neworder);
     app.refund(neworder, cus);
     app.contactRestaurant(res);
     cus.writeReview("****");

  else
     cour.receiveOrder;
     res.contactCourier(neworder, cour);
     cour.delivery(neworder);
     cour.contactCustomer(cus);
     if food is lost or damage during delivery
          cour.contactCustomer(cus);
          cus.requestRefund(app, neworder);
          app.refund(neworder,cus);
          cus.writeReview("****");
     if customer can't get food finally
          cus.contact(app);
          app.serveCustomer(cus);
          app.refund(neworder,cus);
          cus.writeReview("****");
     else
     cus.getFood(neworder);
     order.isDone;
     cus.writeReview("####");
```

else orderFood is not available