1. Design a course management system

```
- Course
  Data: Name, Syllabus, Assignment, Document, zoomLink
  Behaviors:
-Student
  Data: Name, loginCredential
  Behaviors: login, search, lookover, completeTask, discuss, watchVideo
-CourseManagementWebsite
  Data: Name;
  Behaviors: publish, grade
```

Sequence of invoking behaviors on objects

```
  Student Rachel;
  Course INFO5100;
  CourseManagementWebsite Canvas;
  Rachel.login(loginCredential);
  Rachel.search(INFO5100);
  if Canvas publish infomation about INFO5100
    Rachel.lookover(INFO5100.Syllabus,  INFO5100.Assignment,  INFO5100.Document,
INFO5100.zoomLink);
    if INFO5100 publish a document
      Rachel.discuss(INFO5100.document)；
      Canvas.grade(Rachel.discuss(INFO5100.document));
    if INFO5100 publish an assignment
      Rachel.completeTask(INFO5100.assignment)；
      Canvas.grade(Rachel.completeTask(INFO5100.assignment));
    if Rachel need take INFO5100 class
      Rachel.watchVideo(INFO5100.zoomLink);
    if Rachel want to learn the course outline
      Rachel.lookover(INFO5100.Syllabus);
   else
    There is no information about INFO5100;
```

2. Design a pet adoption platform

```
 -Adopter
  Data: Name, Background, loginCredential, petPreference, adoptionReason
  Behaviors: login, search, applyForAdopt, submitInfor, adoptPet
 -Pet
  Data: Name, Type, Color, Characteristic
  Behaviors:
 -AdoptionCenter
  Data: Name
  Behaviors: publishPetInfo, processAdopterInfo, waitNextAdopter
```

Sequence of invoking behaviors on objects

```
  Adopter Rachel;
  Adopter Oliver;
  AdoptionCenter PetsHome;
  Rachel.login(loginCredential);
  Pet lucky = Rachel.search(type, Color, Characteristic);
  if lucky is in PetsHome
    Rachel.applyForAdopt(lucky);
    Rachel.submitInfo(background, petPreference, adoptionReason);
    PetsHome.processAdopterInfo(Rachel.background,          Rachel.petPreference,
Rachel.adoptReason);
    if Rachel suit the adopt condition
      Rachel.adoptPet(lucky);
    else
      PetsHome.waitNextAdopter(Oliver);
   else
      lucky can not be adopted;
```

3. Design an app to book airline ticket.

-Customer
 Data: Name, phone, creditCard, emailAddress, loginCredential
 Behaviors: login, search, book, cancelTicket
-Airline
 Data: Time, Price, seatLevel, departurePlace, arrivalPlace
 Behaviors:
-AirlineTicketAgency
 Data: Courier
 Behaviors: allocateCourier, refund, sendReceipt, checkout
-Courier
 Data: Name
 Behaviors: contactCustomer

Sequence of invoking behaviors on objects

```
   AirlineTicketAgency Feizhu;
   Customer Rachel;
   Airline SeaToSfo = Rachel.search(Time, Price, seatLevel, departurePlace, arrivalPlace);
   Courier Oliver = Feizhu.allocateCurier;
   Rachel.login(loginCredential);
   if Feuzhu has suitable ticket
      Rachel.book(SeaToSfo);
      Feizhu.checkout(Rachel.emailAddress, Rachel.creditCard, Rachel.phone);
      Feizhu.sendReceipt(Rachel.emailAddress);
      if Rachel doesn't want this ticket
         Rachel.cancelTicket(SeaToSfo);
```

```
            Oliver.contactCustomer(Rachel.phone);
            Feizhu.refund(SeaToSfo);
      else
         no suitable ticket;
```

4. Design a course registration platform

- Course
  Data: Name, CourseNumber, MeetingTime, MeetingRoom, ClassSize
  Behaviors:
-Student
  Data: Name, loginCredential
  Behaviors: login, search, register, drop, wait
-CourseRegistrationWebsite
  Data: Administrator;
  Behaviors: allocateAdministrator, publishCourseInformation
-Administrator
  Data: Name
  Behaviors: processStudentRegistration, allocateAviaibleSeat, allocateAdministrator, letInWaitingList

Sequence of invoking behaviors on objects

```
   Student Rachel;
   Student Oliver;
   Course INFO5100;
   CourseRegistrationWebsite myNEU;
   Administrator Susan = myNEU.allocateAdministrator;
   Rachel.login(loginCredential);
   Oliver.login(loginCredential);
   oliver.register(INFO5100);
   Rachel.search(INFO5100);
   if INFO5100 can be registered
      Rachel.register(INFO5100);
      if INFO5100 is Full and have waiting seats
            susan.processStudentRegigtration(Rachel);
            susan.letInWaitingList(Rachel);
            if Oliver.drop(INFO5100);
               myNEU.publishCourseInformation(INFOR5100);
               susan.allocateAviaibleSeat(Rachel);
            else
               Rachel.wait(INFO5100);
     else
      INFO5100 can't be registered
```

5. Order food in a food delivery app

- Customer
  Data: Name, foodPreference, Address, Phone, creditCard, loginCredential
  Behaviors: login, buy, writereview, applyCalcelOrder, applyRefund, search
- Take-outRestaurant
  Data: Food
  Behaviors: sendReceipt, refund, allocateDeliver, prepareFood, checkout
- Food
  Data: size, price, type, taste
  Behaviors:
- Deliver
  Data: Name
  Behaviors: contactCustomer, deliverFood

Sequence of invoking behaviors on objects

```
Customer Rachel;
Take-outRestaurant yelp;
Rachel.login(loginCredential);
Food ChineseFood = Rachel.search(size, price, type, taste);
if yelp has suitable Chinesefood
  Rachel.buy(ChineseFood);
  yelp.checkout(Rachel.address, Rachel.phone, Rachel.creditCard);
      if Rachel want to buy another food
          Rachel.applyCancelOrder(ChineseFood);
          yelp.refund(ChineseFood);
      else
          yelp.prepareFood(ChineseFood);
          Deliver Oliver = yelp.allocateDeliver;
          Oliver.contactCustomer(Rachel);
          Oliver.deliverFood(ChineseFood, Rachel.address);
          if Rachel love this food
              Rachel.writeReview("The food is delicious");
          else
              Rachel.writeReview("I will not eat this food again");
              Rachel.applyRefund(ChineseFood, yelp);
              yelp.refund(Rachel);
    else
          yelp hasn't this kind of food;
```