```
INFO 5100 NAN XIE NUID: 001066682
// 1. Buy a Camera from Amazon.com (assume buyer has shipping address and
creditcard info available in amazon)
Objects: buyer
Behavior: loginToAmazon, searchProduct, selectProduct, addtoCart, checkout,
          selectShippingAddress, selectPmtMethod, selectDeliveryMethod,
          placeOrder, checkLoginInfo, changeSearchCriteria
Data: userid, password, camera, address, creditCardNumber, deliveryMethod

Objects: amazon
Behavior: exhibitProducts, exhibitShippingAddress, confirmShippingAddress,
          exhibitPmtMethod,confirmPmtMethod, exhibitDeliveryMethod,
          confirmDeliveryMethod,oderConfirmation
Data: products, listofShippingAddress, listofCreditCard, listofDeliveryMethod

Boolean authenticated = buyer.loginToAmazon(userid, password)
Loop
if authenticated is valid
    productSearch = buy.searchProduct(cameras)
    products = amazon.exhibitProducts(productSearch)

Loop
    if products is not None
         productChoose = buyer.selectProduct(products)

        if productChoose is not None
            buyer.addtoCart()
            buyer.checkout()
            listofShippingAddress = amazon.exhibitShippingAddress()
            address = buyer.selectShippingAddress(listofShippingAddress)
            amazon.confirmShippingAddress(address)
            listofCreditCard = amazon.exhibitPmtMethod()
            creditCardNumber = buyer.selectPmtMethod(listofCreditCard)
            amazon.cnofirmPmtMethod(creditCardNumber)
            listofDeliveryMethod = amazon.exbihitDeliveryMethod()
            deliveryMethod = buyer.selectDeliveryMethod(listofDeliveryMethod)
            amazon.confirmDeliveryMethod(deliveryMethod)
            buyer.placeOrder()
            amazon.orderConfirmation()
            END
        else
            END
```

```
    else
        buyer.changeSearchCriteria()

else
     buyer.checkLoginInfo()

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

// 2. Design a platform for buying tickets of local events.
Objects: platform
Data: buyerLoginInfo, eventOrganizerLoginInfo, events, locations,dates, tickets
Behavior: Behavior of platform is in buyer and event organizers Behavior flow
--------------------------------------------------------------------------------
Objects: Buyer
Data: username, password, location, eventticketWanttoBuy,name, phoneNumber,
      creditCardNumber, zipcode
Behavior: see below
loginPlatform(){
    boolean signin = platform.login(username, password)
    if signin is valid
        platform.matchLocation(location)
        System.out.print(localEventsList)
    else
        System.out.print(errorMessage)
}

searchEvent(){
    platform.searchEvent(event)
    if event is not None
        System.out.print(eventInfo)
    else
        System.out.print("No such event")
}

selectTicket(){
    if tickets are available
        System.out.print(ticketChoices)
    else
        System.out.print("Sold out")
}

checkout(){
    System.out.print(pmtRequest)
}
```

```
fillPMT(name, phoneNumber, creditcardNumber,zipcode)

placeOrder(){
    Boolean authorize = platform.connectCreditCardCarrier()
    if authorize is True
        System.out.print(orderConfirmation)
    else
        System.out.print(errorMessage)
}

--------------------------------------------------------------------------------
Objects: Local Events Organizers
Data: username, password, eventsDetails, ticketPrice, ticketAmt
Behavior: see below
loginPlatform(){
    boolean signin = platform.login(username, password)
    if signin is valid
        platform.matchLocation(location)
        System.out.print(localEventsList)
    else
        System.out.print(errorMessage)
}

creatEvent(){
    platform.requestEventDetails()
    fillinEventDetails(eventTitle, location, dateRange, eventImage)
}

createTickets(){
    platform.provideTicketType()
    fillinTicketType(ticketName, priceRange, amt, currency, refundChoice)
}

confirmEventCreation(){
    boolean confirmation = platform.confirmEventCreation()
    if confirmation is valid
        System.out.print("Success!")
    else
        System.out.print(errorMessage)
}

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

// 3. Design an app to book a doctor's appointment using your medical insurance
provider.
```

```
Objects: app
Data: userLoginInfo, medicalInsuranceProvidersList,categoryList
Behavior: Behavior of app is in patient booking Behavior
-------------------------------------------------------------------------------
Objects: medicalInsuranceProviders
Data:doctorsName, doctorAvailability, doctorLocation,doctorCategory
Behavior: Behavior of app is in patient booking Behavior
-------------------------------------------------------------------------------
Objects: patient
Data: username, password, medicalInsuranceProvider, insuranceID, name, ssn,
      address, symptom, dateAvailable
Behavior: see below
loginApp(){
    boolean signin = app.login(username, password)
    if signin is valid
        System.out.print(medicalInsuranceProviderList)
    else
        System.out.print(errorMessage)
}

selectMedicalInsuranceProvider(){
    app.requestMedicalInfo()
    fillinMedicalInfo(insuranceID, name, ssn, address)
    Boolean authorize = app.connectMedicalInsuranceProder()
    if authorize is True
        System.out.print(categoryChoice)
        category = selectCategory(symptom)
        medicalInsuranceProviders.searchDoctors(address, category)
        System.out.print(doctorsList)
    else
        System.out.print(errorMessage)
}

Loop doctor in doctorslist
selectDoctor(){
    selectAvailableTime()
    Boolean availbility = medicalInsuranceProviders.confirmAvailability()
    if availability is True
        System.out.print(availbility)
    else:
        System.out.print("NotAvailable")
}
END

confirmAppoitment(){
    medicalInsuranceProviders.sendAppointmenttoDoctor()
```

```
    medicalInsuranceProviders.confirmonAppointment()
    System.out.print(confirmation)
}

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

// 4. Design a job searching platform
Objects: platform
Data: employers, applicantsProfile, jobPositons
Behavior: Behavior of app is in applicant and employers Behavior
--------------------------------------------------------------------------------
Objects: applicant
Data: username, password, name, contactInfo, degree, workExperience, skillset,
      resume, availability, employmentType, locations
Behavior: see below
loginPlatform(){
    boolean signin = platform.login(username, password)
    if signin is valid
        platform.requestBasicInfo
    else
        System.out.print(errorMessage)
}

fillinBasicInfo(name, contactInfo, degree, workExperience, skillset){
    platform.requestResume()
    uploadResume(resume)
    Boolean resumeConfirmation = platform.confirmResumeUpload()
    if resumeConfirmation is True
        profiles = platform.createProfile()
        positionList = platform.matchEmployers(degree, workExperience,
skillset)
        System.out.print(positionList)
    else
        System.out.print(errorMessage)
}

searchJobs(){
    platform.provideSearchCriteria()
    criteria = filterCriteria(jobFunction, employmentType, Location)
    positionsList = platform.matchEmployers(profiles, criteria)
    if positionsList is not None
        System.out.print(positionsList)
    else
        System.out.print("No position available")
}
```

```
applyJobs(){
    platform.sendApplicationtoEmployers()
    platform.confirmApplication
}

--------------------------------------------------------------------------------
Objects: employers
Data: companyBasicInfo, jobFunction, jobDescription, qualifications,
employmentType,
      desiredAvailability
Behavior: see below
postJobposition(){
    platform.requestPositionInfo()
}

fillinPositionInfo(companyBasicInfo, jobFunction, jobDescription,
qualifications, employmentType, esiredAvailability)

receiveApplication(){
    platform.sendApplication()
}

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

// 5. Order Pizza from Dominos (assume order pizza from Domino website for
delivery pay with credit card)
Objects: pizzaLover
Behavior: openDominosWebsite, selectServiceMethod, enterDeliveryAddress,
          chooseFood,chooseSize, chooseCrust,
          chooseCheeseSauce, chooseToppings, addtoOrder,
          checkout, fillinPMTInfo, placeOrder
Data: foodWanttoEat,deliveryAddress, creditCardInfo

Objects: dominos
Behavior: exhibitServiceMethod, searchLocations,
          exhibitMenu, createOrderList,
          requestforPMTMethod, confirmOrder
Data: menu, listofServiceMethod, storelocations, listofPMTInfo

pizzaLover.openDominosWebsit()
listofServiceMethod = dominos.exhibitServiceMethod()
pizzaLover.selectServiceMethod(listofServiceMethod)
deliveryAddress = pizzaLover.enterDeliveryAddress()
confirmAddress = dominos.searchLocations(deliveryAddress)
```

```
if confirmAddress is not None:
    dominos.exhibitMenu()
    foodWanttoEat = pizzaLover.chooseFood()
    Loop choice in foodWanttoEat
       domino.exhibitMenu(choice)
       pizzaLover.chooseSize()
       pizzaLover.chooseCrust()
       pizzaLover.chooseCheeseSauce()
       pizzaLover.chooseToppings()
       pizzalover.addtoOrder()
       dominos.creatOrderList()
    END
    pizzaLover.checkout()
    dominos.requestforPMTMethod()
    pizzaLover.fillinPMTInfo(creditCardInfo)
    pizzaLover.placeOrder()
    dominos.confirmOrder()
else
    END
```