Design a Traffic Controller System for a Junction

Class: Lamp

State: red,green,status

Behaviour:

Light(){

          this.lighted = true;

          if(opposite != null){

              Lamp.valueOf(opposite).light();

          }

}

Class: lampControl

State:

Behaviour:

Change(){

this.status = !(this.status);

}

Class:road

State:

Behaviour:

Run(){

                  for(int i=1;i<1000;i++){

                      try {

                         Thread.sleep((new Random().nextInt(10) + 1)

                      }

                      vechicles.add(Road.this.name + "\_" + i);

                  }

              }

LetPass(){

 if(vechicles.size()>0){

boolean lighted = Lamp.valueOf(Road.this.name).isLighted();

                              if(lighted){

                                 System.out.println(vechicles.remove(0) + " is traversing !");

                              }

                          }  }

2Design a Car Rental System

Class: Customer

State: name,phone,order

Behaviour:

loginSystem() {

System.login(name);

}

SearchCars(SearchParams) {

List<Cars> cars = CarService.search(searchParams);

}

RentCars() {

if(CreditCard.cardAvailable())

System.rent;

else

Error;

}

CancelOrder(bookId){

System.cancelorder(bookingId)

}

Class:Car

State: carId,priced

Behavior:

CheckAvailablity(Date){ }

Class: System

Behaviour:

Login(userName) {

List<String> nameList;

foreach(String name in nameList) {

if(userName == name)

status = true;

}

status = false;

}

SearchCars(SearchParams){ }

Rent(){ }

getCars(){

return car }

CancelOrder(){

fetch traveller car and fare

}

Class: CreditCard

Data: type, bankName, cardNumber,securityCode, expireDate

Behaviour:

cardAvailable() {

Date currentDate = new Date();

return currentDate.compareTo(expireDate);

}

Design a Course Registration Platform

Class: Student

State: userName, studentID

Behaviour:

loginSystem() {

System.login(name);

}

RearchCourse(){

}

RegisterCourse(){

if(studentID.idAvailable())

System.registerCourse(this.Data);

else

Error;

}

showCourse {

if(courses.size()>0)

{

for (Course course : courses) {

System.out.println(course);

}

}

else

{

System.out.println("还没有选课");

}

}

DropCourse(){deletemanage(int id);

}

Class:Administrator

State: runName,courseName,teacherName,StudentName,

Behaviour:

addStudent (student stu){

}

addCourse(student stu ){

}

Class: System

State:

Behavior:

loginSystem() { }

searchCourse(){}

registerCourse(){ }

addStudent(){ student stu }

dropCourse(){Administrator.course=null

Student.course=null }j

Class: Service

Data:

Behaviour:

idAvailable() {

id currentId = new id();

return currentId.compareTo(presentDate);

}

Design a vending Machine (will accept coins of different denominations and vends out candies, soda etc)

Class:autoSalemachine

State: result,drink,food,money

Behaviour:ship(){ Order order=new order;)

}

takeChange(){ if (needSum > userMoneySum) {

JOptionPane.showMessageDialog(null, "投入的金额小于需要购买商品的总价，请再投入"

+ (needSum - userMoneySum) + "元或取消一些商品。");

reinitGoodCount();

return;}

Class:costumer

State:orderId

Behavior:

Reviewprice(){machineService.showPrice }

Reviewstock(){

gp\_coke.showRemain();

gp\_fanta.showRemain();

gp\_rtea.showRemain();

gp\_gtea.showRemain();}

Order(double order ){this.order=order }

Class:administrator

State:

Behavior:

Reviewstock(){machineService.showStock}

Addstock(double stock){this.stock=stock }

TakeawayMoney(){}

Class:machineService

State: stock, money, contactInformation,price

Behavior:

showStock(){

System.out.print(Stock) }

showPrice(){

System.out print(price)}

takeChange(){ }

Addstock(double dring,doble food){this.food=food,this.drink=drink }

TakeawayMoney(){}

Design a Parking lot

Class:Administrator

State:ownerId, name, address, list of parkingLot

Behavior:

Class: User

State: name,car,

Behavior:

getInto(){BookingDetails=HotelService.book(checkinDate,checkOutDate,numberOfpark)}

getOut(){ parkingService.Leave(orderId) }

Review(){System.out.print(this.order);}

Contactadmin(){ parkingService.Contactadmin(number)}

Class:parkingLot

State:parkingTime,name,carId, parkingPrice, totalspace, reviews, rating

Behavior: checkAvailability(date, numberOflot) { }

Class:parkingService

Behavior:

Review(){}

getInto(){isAvailable() {

for (int i = 0; i < this.status.length; i++) {

if (this.status[i] == null) {

return i;

} }

Leave(bookingId){

calculate time and charge

PaymentService.initiatefare(parkingTime, price)

}

Contactadmin(number) {phoneCallAPI.call(number);}