Software Engineering Coursework II

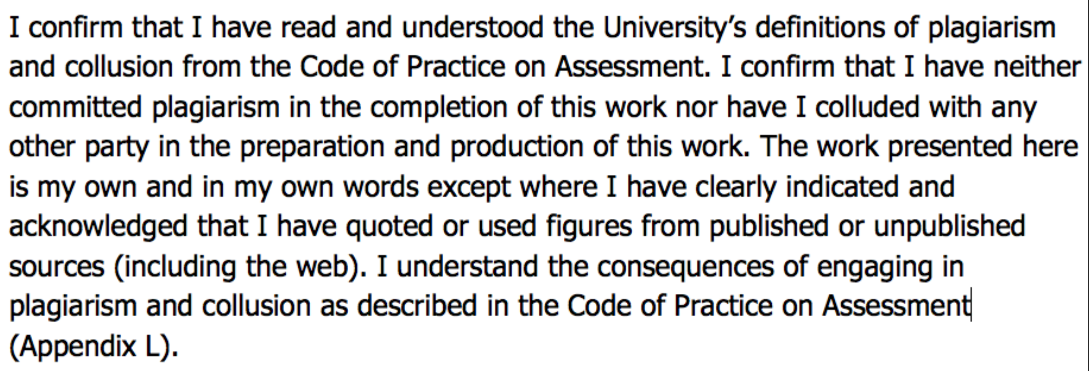
Dequn Teng

December 12, 2019

Abstract

In this coursework, I am been approached by a garage to design a computer system to log jobs. Additionally, The garage will perform inspection tasks, repair tasks, and maintenance tasks on different types of vehicles (car, bus, van). In terms of staffs in the garage, the shop has multiple staff members working each day: the receptionist, the manager, and several mechanics, serving different functionalities. Finally, the workflow is represented by the change of tickets from waiting, progress, to check, signed-Off, and completed achieved by different staffs. There are four questions in this scenario, illustrated as follows. of vehicle: cars, vans, and busses.

Declaration



I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

1 Notations, assumptions, and Justifications

1. One customer can have many vehicle in garage, for different services, based on task sheet.
2. One ticket only record one service, based on instructor's email feedback.
3. One Bill contains all tickets of this customer.

2 Task 1

2.1 Problem Restatement

You are to create 4 separate lists, each with added details if required

(a) List all candidate classes, their candidate attributes, and their candidate operations.

(b) List all potential inheritance relationships.

(c) List those candidate classes that are to be eliminated, and give justification as to why.

(d) Give the final list of candidate classes, along with their attributes and their candidate operations. You should ensure minimal data duplication (e.g. if a customer has multiple cars in for repair).

2.2 Answers to questions

2.2.1 (a)

According to Thomas, the noun identification technique can be used to identify candidate class [1]. There are fifty two nouns which in this requirement document. The candidate class name, candidate attributes, and candidate operations are listed as follows.

Table 1: Candidate Class, attributes, and operations

candicate class	candidate attributes	candicate operations

garage	garageID
	garageName
	manager
		openGarage(garageID)
		closeGarage(garageID)

computer	(NULL too general)	(NULL too general)
system	systemID systemName	logInSystem(staffID, staffpassword)

log	logID, logName	writeLog(logID, logValue)
		readLog(logID)
		updateLog(logID, logValue)
		deleteLog(logID)

job	jobID
	jobType
	jobOwner
	jobStatus
		startJob(jobID)
		endJob(jobID)

task	taskID
	taskName
	taskType
	mechanicsID
	TicketID
	taskStatus
		startTask(taskID)
		endTask(taskID)
		findWaitingTask()
		findCheckTask()
		findProgressTask()
		findSignedOffTask()
		findCompleteTask()

inspection task	InspectionID
	InspectionName
	InspectionType
	mechanicsID
	InspectionStatus
		startInspectionTask(taskID)
		endInspectionTask(taskID)
		findInspectionWaitingTask()
		findInspectonCheckTask()
		findInspectionProgressTask()
		findInspectionSignedOffTask()
		findInspectionCompleteTask()

repair task	repairID
	repairName
	repairType
	mechanicsID
	repairStatus
		startRepairTask(taskID)
		endRepairTask(taskID)
		findRepairWaitingTask()
		findInspectonCheckTask()
		findRepairProgressTask()
		findRepairSignedOffTask()
		findRepairCompleteTask()

maintaincetask	maintainanceID
	maintainanceName
	maintainanceType
	mechanicsID
	maintainanceStatus
		startMaintainanceTask(taskID)
		endMaintainanceTask(taskID)
		findMaintainanceWaitingTask()
		findInspectonCheckTask()
		findMaintainanceProgressTask()
		findMaintainanceSignedOffTask()
		findMaintainanceCompleteTask()

type	(NULL too general)	(NULL too general)

vehicle	vehicleID
	vehicleName
	vehicleStatus
	customerID
	vehicleType
	vehicleNote
		vehicleInGarage(vehicleID)
		vehicleInProgress(vehicleID mechanicsID)
		vehicleInCheck(vehicleID managerID)
		vehicleInSignedOff(vehicleID receptionistID)
		vehicleIn

car	(enum type for vehicleType)	(enum type for vehicleType)
van	(enum type for vehicleType)	(enum type for vehicleType)
bus	(enum type for vehicleType)	((enum type for vehicleType)
MOT test	(enum type for inspectionType)	(enum type for inspectionType)
general diagonstic test	(enum type for inspectionType)	(enum type for inspectionType)

customer	customerID customerName phoneNumber vehicleID paymentStatus ticketID	requireService(customerID vehicleID seriviceType) payBill(customerID vehicleID)
problem	(NULL too general)	(NULL too general)
body repair	(enum type for repairType)	(enum type for repairType)
engine repair	(enum type for repairType)	(enum type for repairType)
window replacement	(enum type for repairType)	(enum type for repairType)
insurance mandated repair	(enum type for repairType)	(enum type for repairType)
air conditioning to-up	(enum type for maintainanceType	(enum type for maintainanceType
body respray	(enum type for maintainanceType	(enum type for maintainanceType
typre change	(enum type for maintainanceType	(enum type for maintainanceType
shop	(NULL too general)	(NULL too general)
member	(should be combined with staff)	(should be combined with staff)

staff	#staffID:Int
	#staffType:enum
	#staffName:String
	#staffpassword:Int
		+getstaffID(): Int
		+setstaffID(staffID: Int)
		+getstaffType(): enum
		+setstaffType(staffType: enum)
		+getstaffName():string
		+setStaffName(staffName:string)
		+getStaffPassword():int
		+setStaffPassword(password:int)

receptionist		+discussWithCustomer(customerRequirement: string domainKnowldge: string) :string +openTicket(customerID : int vehicleID: int work : enum deadline: date quotedPrice: double) findSettledBill():ticket findCustomerPhoneNum(Ticket: Ticket): phoneNum telephoneCustomer(phoneNumber: int): string findBill(Ticket: Ticket): double completeTicket(Ticket: Ticket)

manager		+viewCheckTickets()
		+checkIfGoodStandard(TicketID: int):
		+updatePrice(TicketID: int price: double)
		+signedOffTicket(TicketID: int)

mechanics		+viewWaitingTickets(): Ticket
		+getFirstAvailableOneToProgress(TicketID:int)
		+getFirstAvailableOneToCheck(TicketID: int)
		+addTicketNote(TicketID: int)

charge	(NULL too general)	(NULL too general)
day	(NULL it is the build-in type)	(NULL it is the build-in type)
workflow	(NULL not in system)	(NULL not in system)
reception office	(NULL not in system)	(NULL not in system)
need	(NULL not in system)	(NULL not in system)
domain	(NULL not in system)	(NULL not in system)
knowledge	(NULL not in system)	(NULL not in system)
advice	(NULL not in system)	(NULL not in system)
ticket	customerID : int vehicleID: int work : enum deadline: date quotedPrice: double ticketNote: string	
work	workID workName workType	
deadline	(NULL it is the build-in type date)	(NULL it is the build-in type date)
quoted price	(NULL it is the build-in type double)	(NULL it is the build-in type double)
status	waiting progress check signedOff complete	
start	(NULL too general)	(NULL too general)

progess	(NULL too general)	(NULL too general)
unexpected cost	(NULL it is the build-in type double)	(NULL it is the build-in type double)
ticket notes	(NULL it is the build-in type string)	(NULL it is the build-in type string)
standard	(NULL too general)	(NULL too general)

bill	billID
	billName
	billBalance
		getBillBalance(billID)
		getCustomerBill(CustomerID)
		getVehicleBill(vehicleID)
		setCustomerBill(customerID

person	personID, personName	

2.2.2 (b)

According to Thomas's lecture 19, the inheritance is defined as the sharing of attributes and operations among classes based upon a hierarchical relationship [2]. Therefore, the potential inheritance class is as follows.

1. Inspection inherits from Task
2. Repair inherits from Task
3. Maintenance inherits from Task
4. Receptionist inherits from staff
5. Manager inherits from staff
6. Mechanics inherits from staff
7. Customer inherits from person
8. Staff inherits from person

2.2.3 (c)

Based on the object orientation design, the class to be eliminated as listed as follows.

Table 2: The eliminated classes and reasons

class name	reason
garage	too general
computer	too general
system	too general
log	too general
job	too general
type	too general
car	enum type for vehicleType
van	enum type for vehicleType
bus	enum type for vehicleType
MOT test	enum type for inspectionType
general diagonstic test	enum type for inspectionType
problem	too general
body repair	enum type for repairType
engine repair	enum type for repairType
window replacement	enum type for repairType
insurance mandated repair	enum type for repairType
air conditioning to-up	enum type for maintainanceType
body respray	enum type for maintainanceType
typre change	enum type for maintainanceType
shop	too general
member	should be combined with staff
charge	too general
day	it is the build-in type
workflow	not in system
reception office	not in system
need	not in system
domain	not in system
knowledge	not in system
advice	not in system
work	too general
deadline	it is the build-in type date
quoted price	it is the build-in type double
status	it is an attribute of ticket
start	too general
progess	too general
unexpected cost	it is the build-in type double
ticket notes	it is the build-in type string, as attribute of ticket
standard	too general

2.2.4 (d)

The candidate class, attributes and operations are as follows.

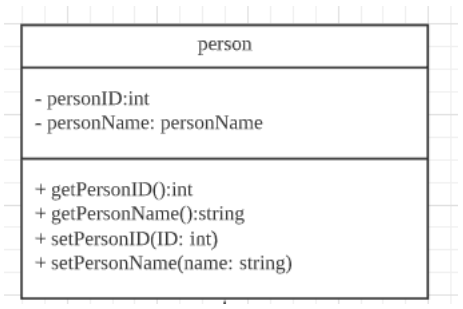


Figure 1: candidate class 1

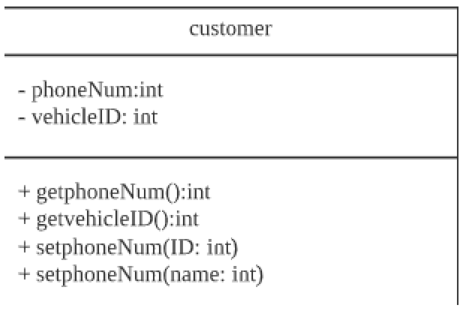


Figure 2: candidate class 2

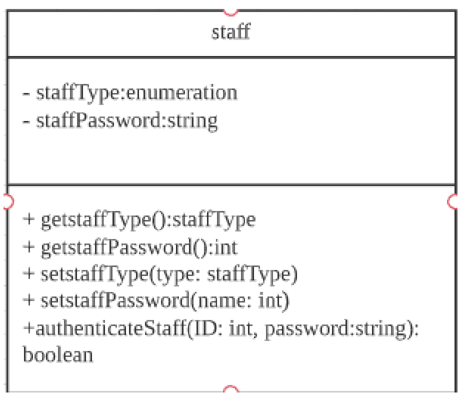

staff
- staffType:enumeration
- staffPassword:string
+ getstaffType():staffType
+ getstaffPassword():int
+ setstaffType(type: staffType)
+ setstaffPassword(name: int)
+authenticateStaff(ID: int, password:string): boolean

Figure 3: candidate class 3

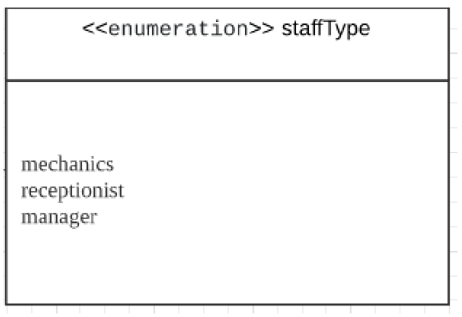


Figure 4: candidate class 4

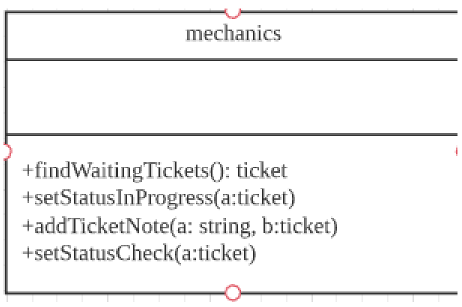

mechanics
+findWaitingTickets(): ticket
+setStatusInProgress(a:ticket)
+addTicketNote(a: string, b:ticket)
+setStatusCheck(a:ticket)

Figure 5: candidate class 5

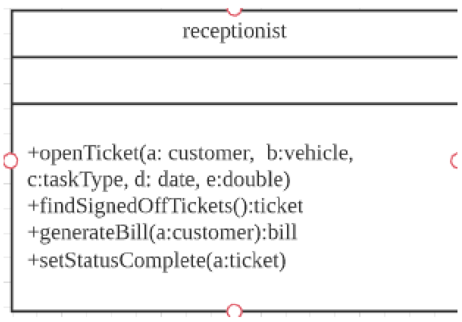

receptionist
+openTicket(a: customer, b:vehicle, c:taskType, d: date, e:double)
+findSignedOffTickets():ticket
+generateBill(a:customer):bill
+setStatusComplete(a:ticket)

Figure 6: candidate class 6

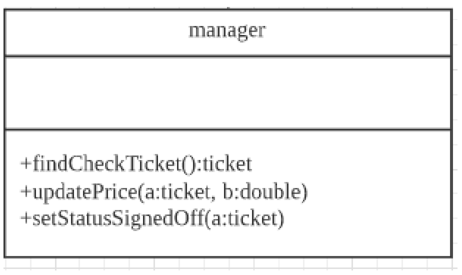

manager
+findCheckTicket():ticket
+updatePrice(a:ticket, b:double)
+setStatusSignedOff(a:ticket)

Figure 7: candidate class 7

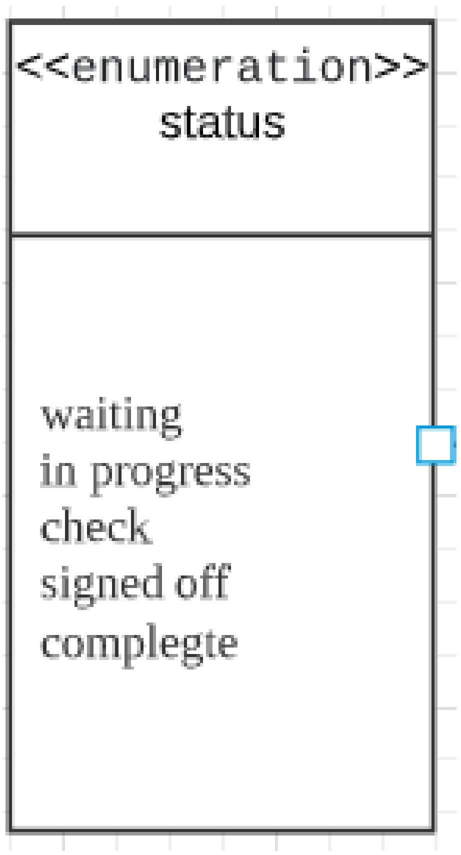


Figure 8: candidate class 8

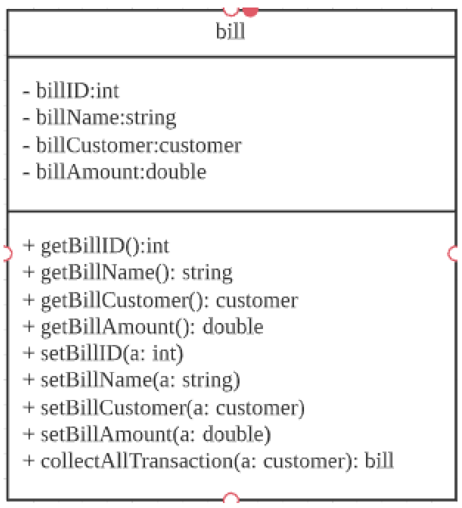


Figure 9: candidate class 9

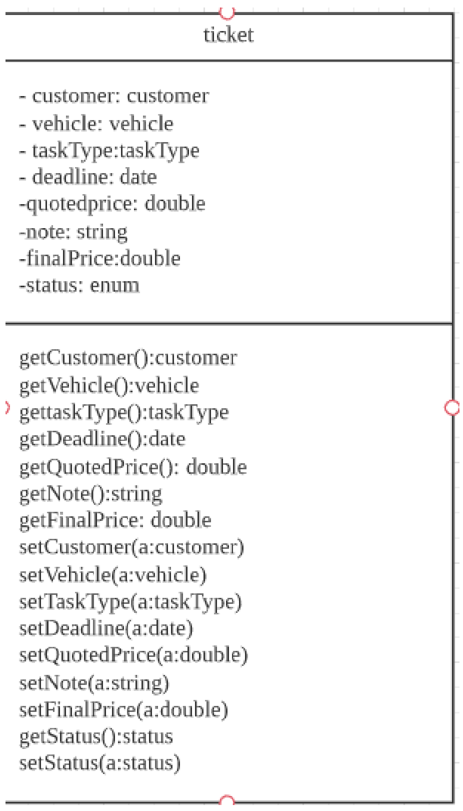


Figure 10: candidate class 10

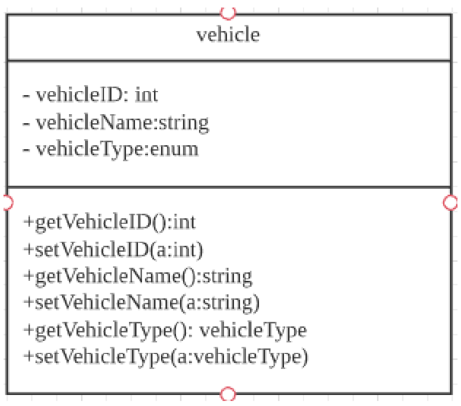


Figure 11: candidate class 11

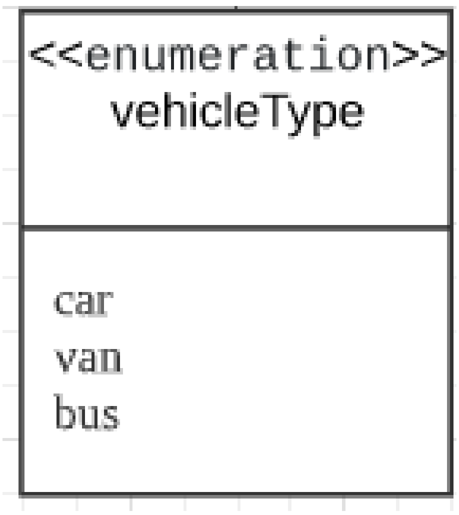


Figure 12: candidate class 12

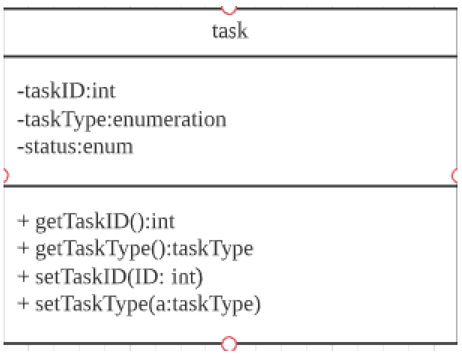


Figure 13: candidate class 13

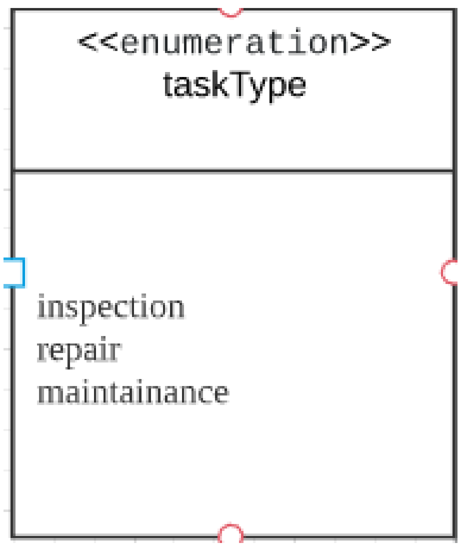


Figure 14: candidate class 14

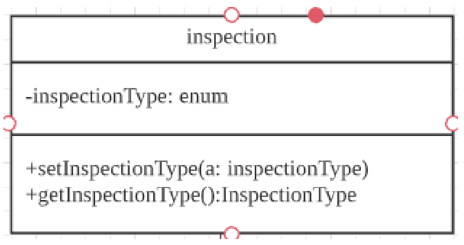

inspection
-inspectionType: enum
+setInspectionType(a: inspectionType)
+getInspectionType():InspectionType

Figure 15: candidate class 15

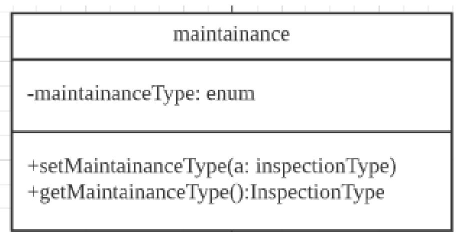

maintainance
-maintainanceType: enum
+setMaintainanceType(a: inspectionType)
+getMaintainanceType():InspectionType

Figure 16: candidate class 16

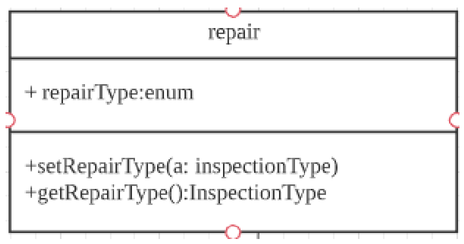

repair
+ repairType:enum
+setRepairType(a: inspectionType)
+getRepairType():InspectionType

Figure 17: candidate class 17

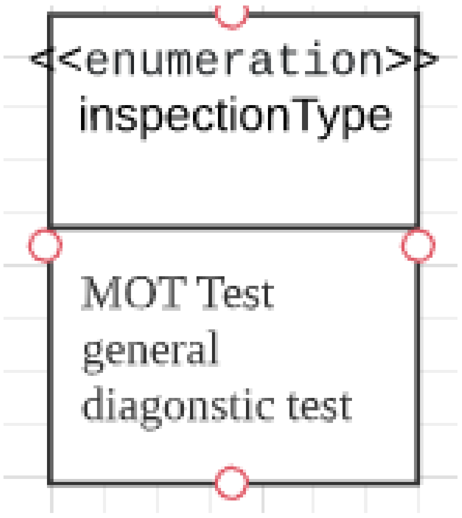


Figure 18: candidate class 18

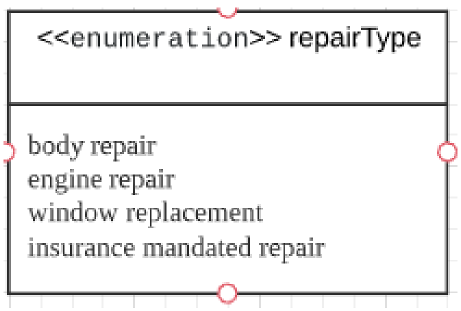


Figure 19: candidate class 19

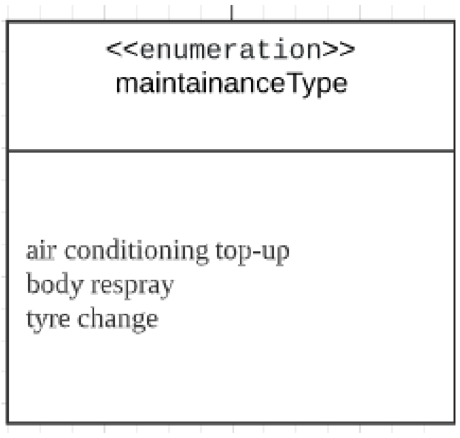


Figure 20: candidate class 20

3 Task 2

3.1 Problem Restatement

Produce CRC Cards for each class. For each CRC Card, comment on whether the class is "Good" or "Bad", and give justification for your reasoning. If it is "Bad" then you should state how it may be improved, but do not implement this improvement.

3.2 Answers to questions

person

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility less than 3 and collaborator less than 3)	
Responsibility	**Collaborators**
1. stores information for person 2. serves to the super class for all human	1. customer 2. staff

customer

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility less than 3 and collaborator less than 3)	
Responsibility	**Collaborators**
1. stores information for customer 2. communicate with receptionist	1. receptionist 2. ticket

staff

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility less than 3 and collaborator equal to 4 (still acceptable))	
Responsibility	**Collaborators**
1. stores information for staff 2. serves to the super class for manager, receptionist, mechanics	1. manager 2. receptionist 3. mechanics 4. person

staffType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of staff	1. staff

mechanics

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 2 and collaborator of 3)	
Responsibility	**Collaborators**
1. finish task 2. update ticket	1. staff 2. task 3. ticket

receptionist

Comments: This is a medium CRC card with high cohesion and medium coupling (Responsibility of 4 and collaborator of 3), this is due to the working property of receptionist.	
Responsibility	**Collaborators**
1. meet customer and offer advice 2. update ticket information 3. telephone customer 4. collect payment	1. staff 2. customer 3. ticket

manager

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 4 and collaborator of 3) due to the working property of manager.	
Responsibility	**Collaborators**
1. update price if necessary 2. become mechanics if necessary 3. check ticket 4. update ticket	1. staff 2. ticket 3. mechanics

staffType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of staff	1. staff

task

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 2 and collaborator of 4)	
Responsibility	**Collaborators**
1. stores info for three type of tasks (superclass) 2. carried out by mechanics	1. inspection 2. repair 3. maintenance 4. mechanics

taskType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of task	1. task

inspection

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 3)	
Responsibility	**Collaborators**
1. inspect the vehicle condition	1. inspectionType 2. mechanics 3. task

inspectionType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of inspection	1. inspection

repair

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 3)	
Responsibility	**Collaborators**
1. stores info for repair task	1. repairType 2. mechanics 3. task

repairType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of repair	1. repair

maintenance

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 3)	
Responsibility	**Collaborators**
1. stores info for maintainance task	1. maintainanceType 2. mechanics 3. task

maintainanceType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for three type of maintenance	1. maintenance

ticket

Comments: This is a medium CRC card with high cohesion and high coupling (Responsibility of 1 and collaborator of 5)	
Responsibility	**Collaborators**
1. stores info for one service of a customer	1. customer 2. mechanics 3. task 4. receptionist 5. manager

status

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for ticket status	1. ticket

bill

Comments: This is a good CRC card with high cohesion and relatively low coupling (Responsibility of 1 and collaborator of 4)	
Responsibility	**Collaborators**
1. stores information for all services ordered by one customer	1. ticket 2. customer 3. receptionist 4. vehicle

vehicle

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 4)	
Responsibility	**Collaborators**
1. stores info for a vehicle	1. ticket 2. bill 3. mechanics 4. customer

vehicleType

Comments: This is a good CRC card with high cohesion and low coupling (Responsibility of 1 and collaborator of 1)	
Responsibility	**Collaborators**
1. stores info for vehicleType	1. vehicle

4 Task 3

4.1 Problem Restatement

Produce a UML Class Diagram showing the classes, attributes, operations, and associations of the system (use answers from Task 1 to guide you). You should be sure to use the correct type of association, navigability, and multiplicity.

4.2 Answers to questions

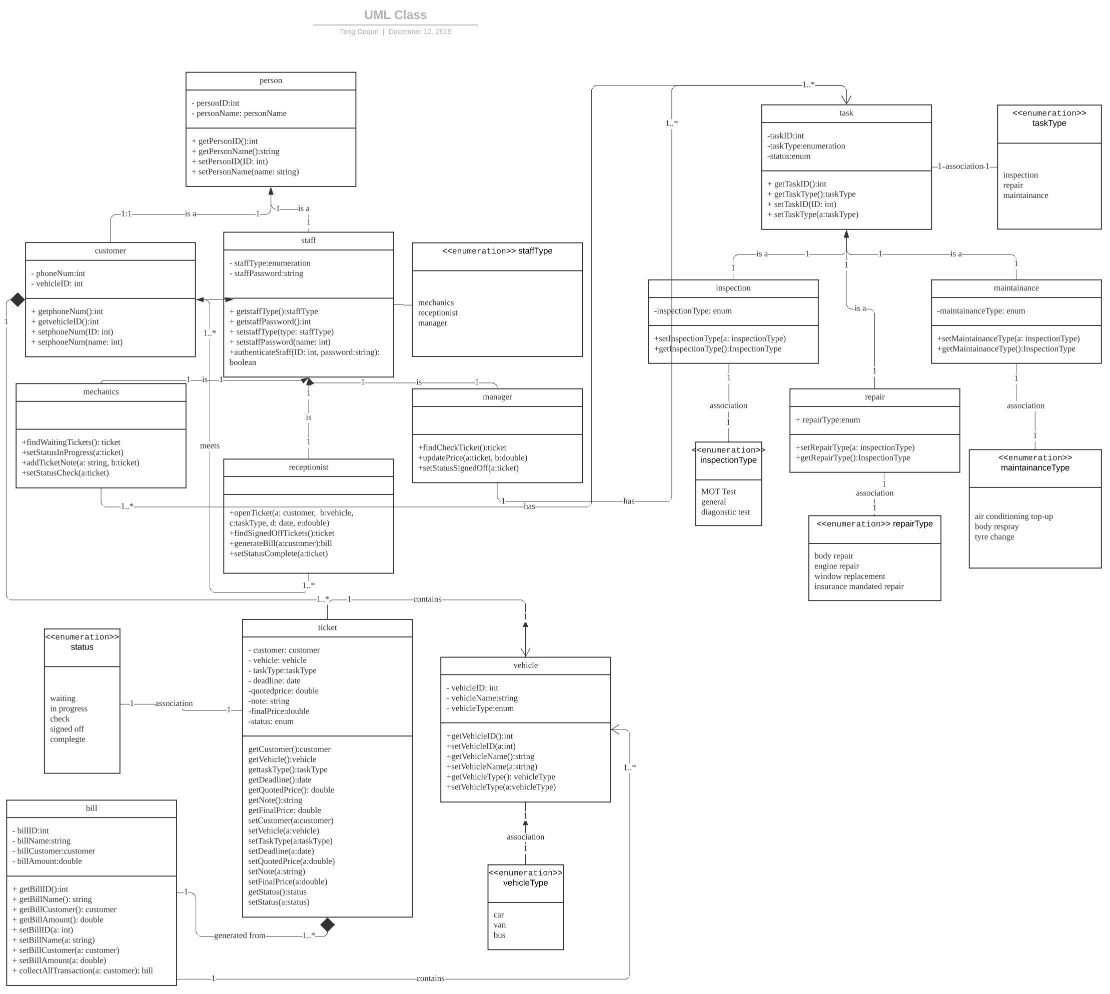


Figure 21: Overview of the UML class diagram

The simplified version for UML class diagram

The task 5 is continued as follows.

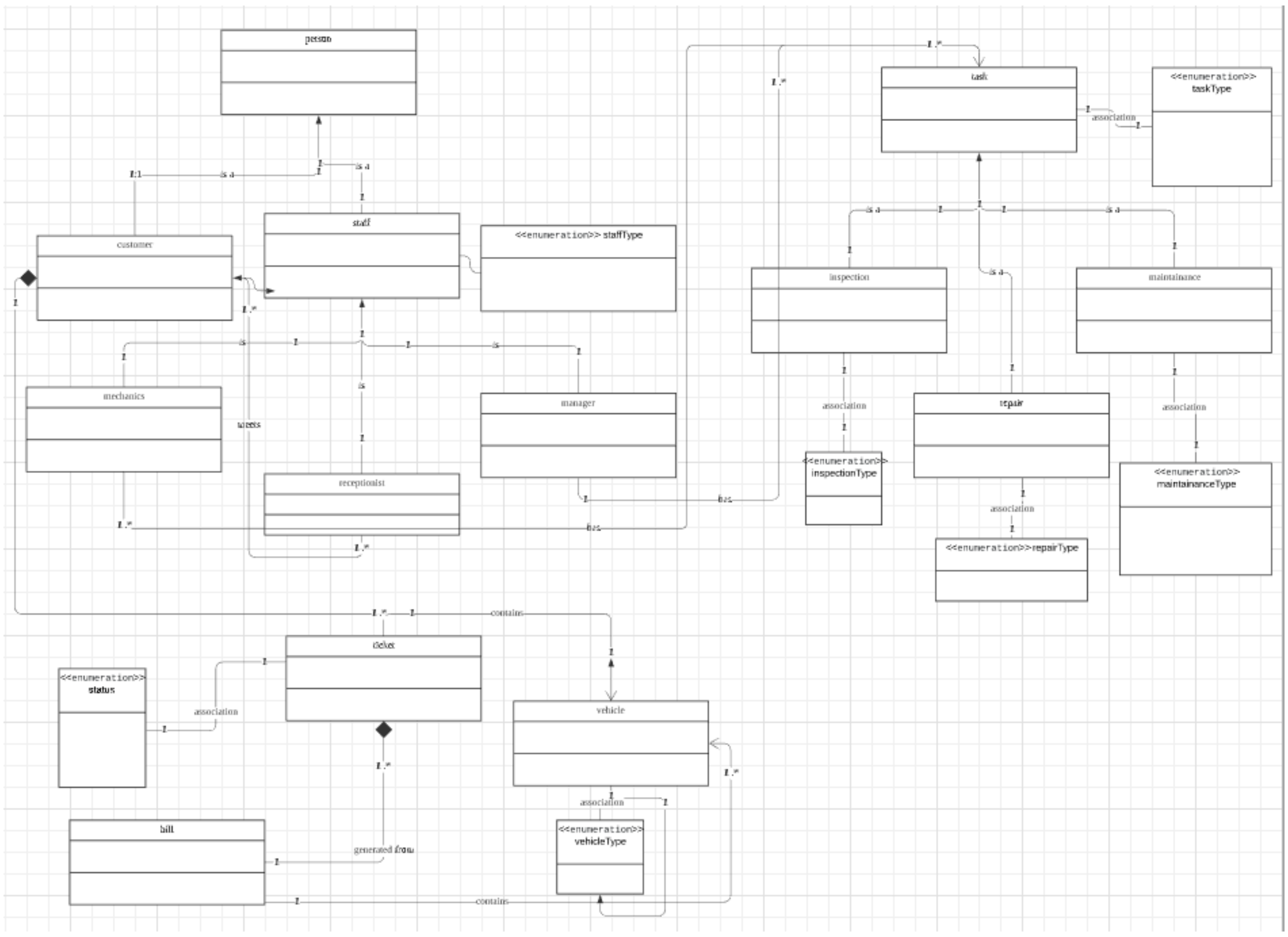


Figure 22: simplified version for UML class diagram

5 Task 5

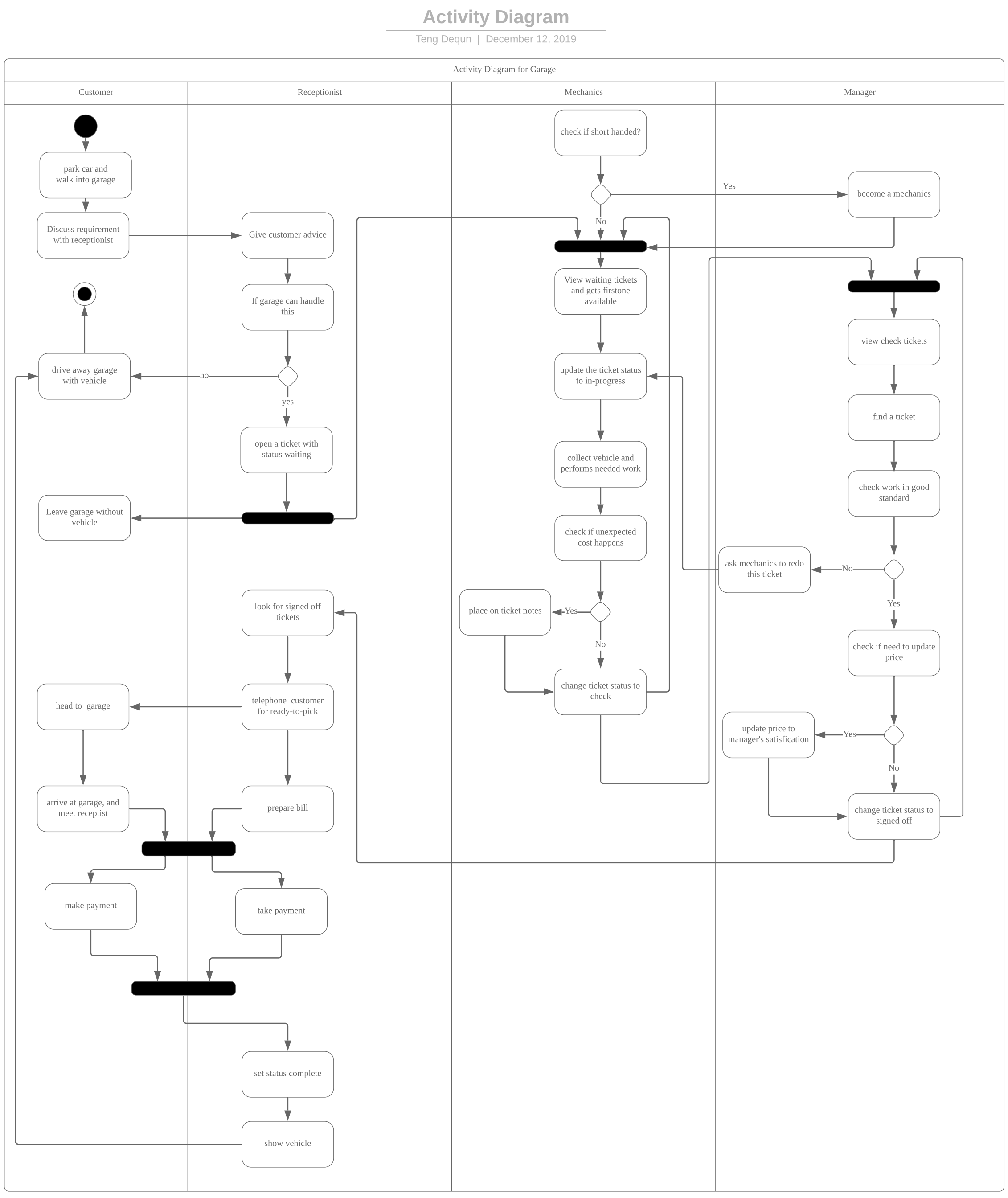

Activity Diagram
Teng Dequn | December 12, 2019
Activity Diagram for Garage
Customer
Receptionist
Mechanics
Manager
park car and walk into garage
Discuss requirement with receptionist
Give customer advice
If garage can handle this
no
yes
drive away garage with vehicle
open a ticket with status waiting
Leave garage without vehicle
check if short handed?
Yes
No
become a mechanics
View waiting tickets and gets firstone available
update the ticket status to in-progress
collect vehicle and performs needed work
check if unexpected cost happens
place on ticket notes
Yes
No
change ticket status to check
view check tickets
find a ticket
check work in good standard
ask mechanics to redo this ticket
No
Yes
check if need to update price
update price to manager's satisfication
Yes
No
change ticket status to signed off
look for signed off tickets
telephone customer for ready-to-pick
head to garage
prepare bill
arrive at garage, and meet receptist
make payment
take payment
set status complete
show vehicle

Figure 23: Activity diagram

References

[1] Comp201 – software engineering 1 lecture 20 – oo design & uml. `https://vital.liv.ac.uk/bbcswebdav/pid-2051345dtcontentrid-120962501/courses/COMP201-201920/201-20_prev.pdf`. (Accessed on 12/12/2019).

[2] Comp201 – software engineering i lecture 19. `https://vital.liv.ac.uk/bbcswebdav/pid-2049855-dt-content-rid-12061928_1/courses/COMP201-201920/201-19.pdf`. (Accessed on 12/12/2019).