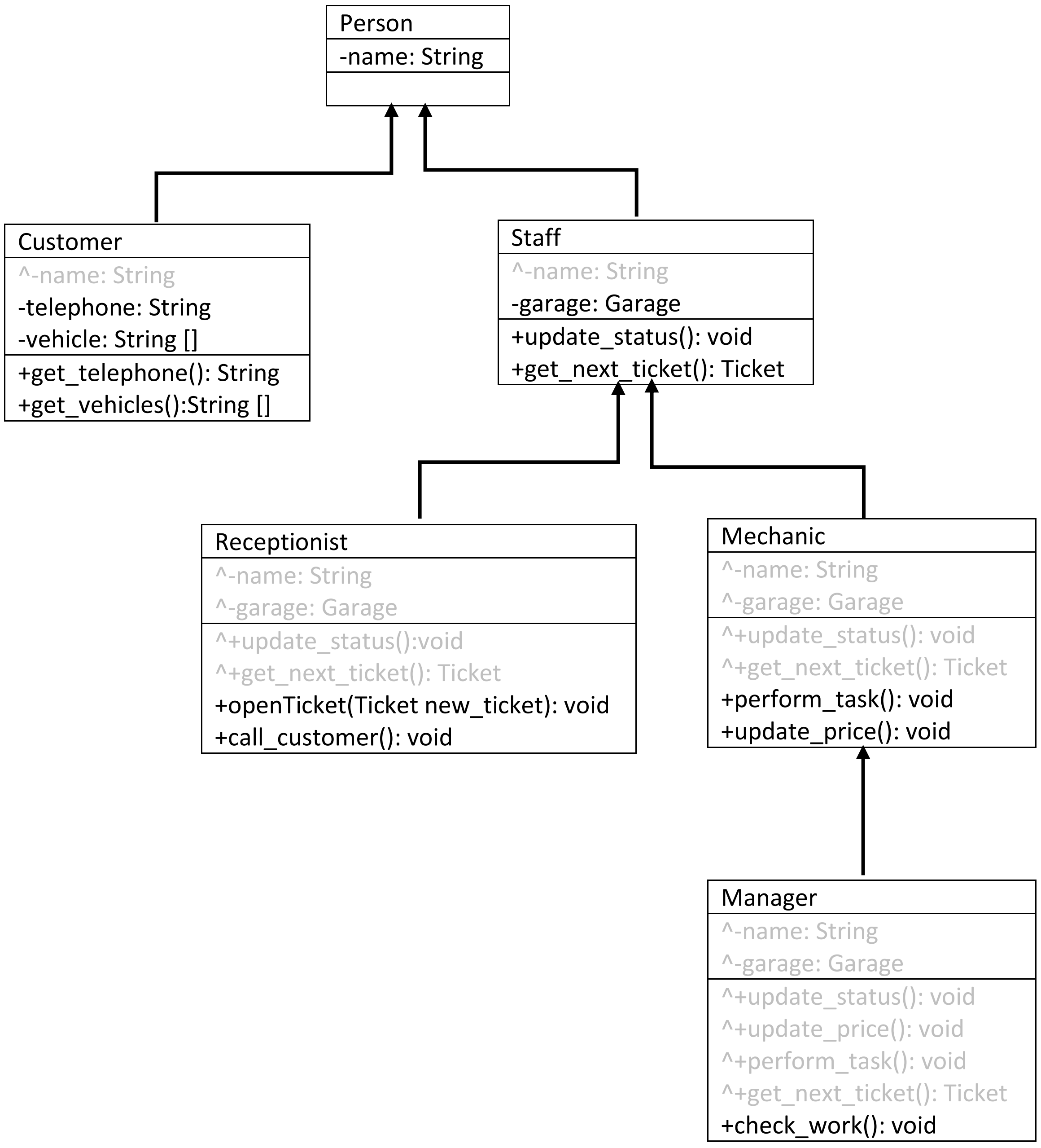
COMP201 - Assignment 2
Name: Jin, Minhao
University Username: sgmjin2
Student ID: 201447766

Task 1 (20%) You are to create 4 separate lists, each with added details if required.

1a) List all candidate classes, their candidate attributes, and their candidate operations.

Candidate class	Candidate attributes	Candidate operations
Garage		Perform inspection task Perform repair task Perform maintenance task
Inspection task	MOT test General diagnostic test	
Repair task	Body repair Engine repair Window replacement	
Maintenance task	air conditioning top-up body resprays tyre change	
Person	name	
Customer	name telephone vehicle	Park vehicle Walk Discuss need
Staff	name	
Receptionist	name	Open tickets Get ‘signed off’ ticket Call customer Set ticket status to ‘waiting’
Mechanic	name	Get ‘waiting’ ticket Set ticket status to ‘in progress’ Set ticket status to ‘check’ Perform work Update price
Manager	name	Get ‘check’ ticket Set ticket status to ‘signed off’ Perform task Update price Check work
Ticket	Customer Vehicle Work deadline price status	
Vehicle	Cars Vans buses	
Bill		
Shop		

1b) List all potential inheritance relationships.



1c) List those candidate classes that are to be eliminated, and give justification as to why.

Candidate classes that needs to be eliminated: ‘Work’ and ‘Vehicle’.

Reason: Class ‘Work’ and ‘Vehicle’ only has only one string attribute which defines their types. Additionally, they have few interactions with the other classes. Thus, it is meaningless to keep these two classes and they can be directly replaced by some strings in further coding.

Candidate classes that needs to be eliminated	reason
Inspection task	It is unnecessary to have these 3 classes because the task is defined on the ticket with the attribute ‘work’. ‘work’ is a string value which can specify the specific work type mechanics should perform.
Repair task	
Maintenance task	
Vehicle	This class can also be replaced by attribute ‘vehicle’ in class ‘customer’ which should be a string array to record what types of vehicles the customer has.
Bill	‘Bill’ is synonymous with the attribute ‘price’ in class ‘ticket’
Shop	‘Shop’ is synonymous with the class ‘garage’

1d) 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).

Candidate class	Candidate attributes	Candidate operations
Garage	-People: Vector<Staff> -Tickets: Vector<Ticket>	+perform_inspection_task(): void +perform_repair_task(): void +perform_maintenance_task(): void +view_waiting_tickets():Ticket +view_check_tickets(): Ticket +view_signoff_tickets(): Ticket
Person	-name: String	+get_name(): String
Customer	-name: String -telephone: String -vehicle: String []	+get_telephone(): String +get_vehicles():String []
Staff	-name: String -garage: Garage	+update_status(): void +get_next_ticket(): Ticket (abstract method)
Receptionist	-name: String -garage: Garage	+update_status(): void +openTicket(Ticket new_ticket): void +get_next_ticket(): Ticket +call_customer(): void
Mechanic	-name: String -garage: Garage	+update_status(): void +perform_task(): void +update_price(): void
Manager	-name: String -garage: Garage	+update_status(): void +perform_task(): void +update_price(): void +check_work(): boolean
Ticket	-customer: Customer -vehicle: String -work: String -deadline: Date -price: double -status: String	+get_status(): String +get_price(): double +set_status(): void +set_price(): void

Task 2 (20%) 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.

Garage	
Responsibilities	Collaborators
1.Record staff members and tickets. 2.Provide methods for staff to view tickets.	Ticket Staff
Comment: Good Cohesion: ‘Garage’ has the attributes staff members and tickets and it also provides the method to inspect these attributes. Coupling: ‘Garage’ has the relation with 2 class: ‘ticket’ and ‘Staff’. Thus, it is strong-cohesion and loose coupling.	

Person	
Responsibilities	Collaborators
1.Provide his/her name	
Comment: Good Cohesion: ‘Person’ has the attribute ‘name’ and the method to inspect this attribute. Coupling: ‘Person’ has no relation with any other classes. Thus, it is strong -cohesion and loose coupling.	

Customer	
Responsibilities	Collaborators
1.Provide his/her personal phone number 2.Provide his/her vehicle information	Person
Comment: Good Cohesion: ‘Customer’ has the attribute ‘telephone’ and ‘vehicles’ and it also provides methods to inspect the attributes. Coupling: ‘Customer’ has the relation with only 1 class ‘person’. Thus, it is strong-cohesion and loose coupling.	

Staff	
Responsibilities	Collaborators
1.Access the data of Garage 2.Update ticket status	Person Garage
Comment: Good Cohesion: ‘Staff’ has the attribute ‘garage’ and it also provides methods to inspect the attribute. Coupling: ‘Staff’ has the relation with 2 classes ‘person’ and ‘Garage’. Thus, it is strong-cohesion and loose coupling.	

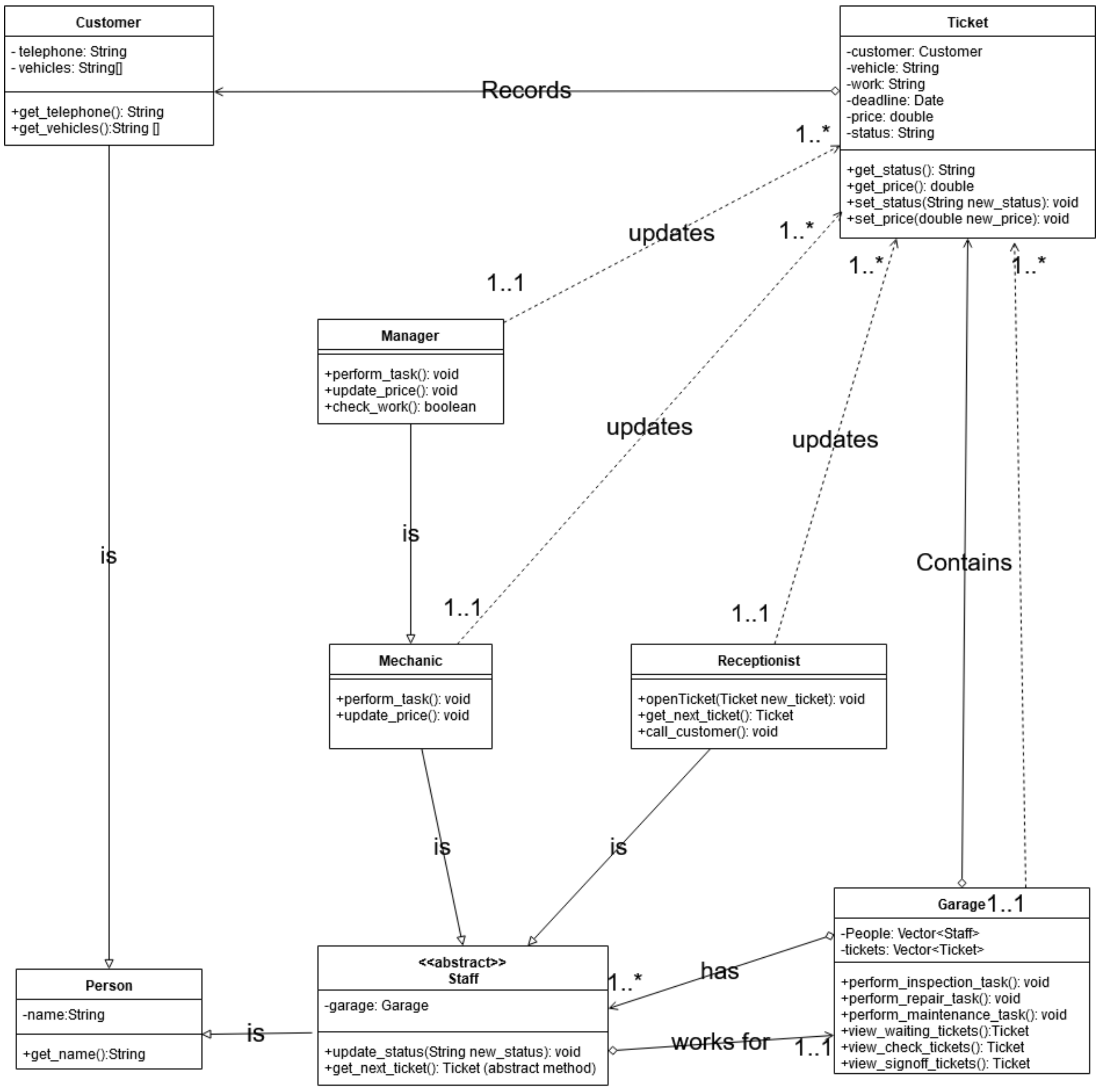
Receptionist	
Responsibilities	Collaborators
1.Open tickets for customer 2.View ‘signoff’ tickets 3.Call customers	Garage Staff Ticket Customer
Comment: Bad Cohesion: ‘Staff’ has the attribute ‘garage’, but its responsibilities are not focused. Coupling: ‘Staff’ has the relation with 4 classes ‘Garage’, ‘Staff’, ‘Ticket’ and ‘Customer’. Thus, it is weak-cohesion and strong coupling.	

Mechanic	
Responsibilities	Collaborators
1.View ‘waiting’ tickets 2.Perform the work required 3.Update cost on the ticket	Garage Staff Ticket
Comment: bad Cohesion: The responsibilities of ‘Mechanic’ are grouped because they perform similar functions Coupling: ‘Staff’ has the relation with 3 classes ‘Garage’, ‘Staff’ and ‘Ticket’. Thus, it is weak-cohesion and strong coupling.	

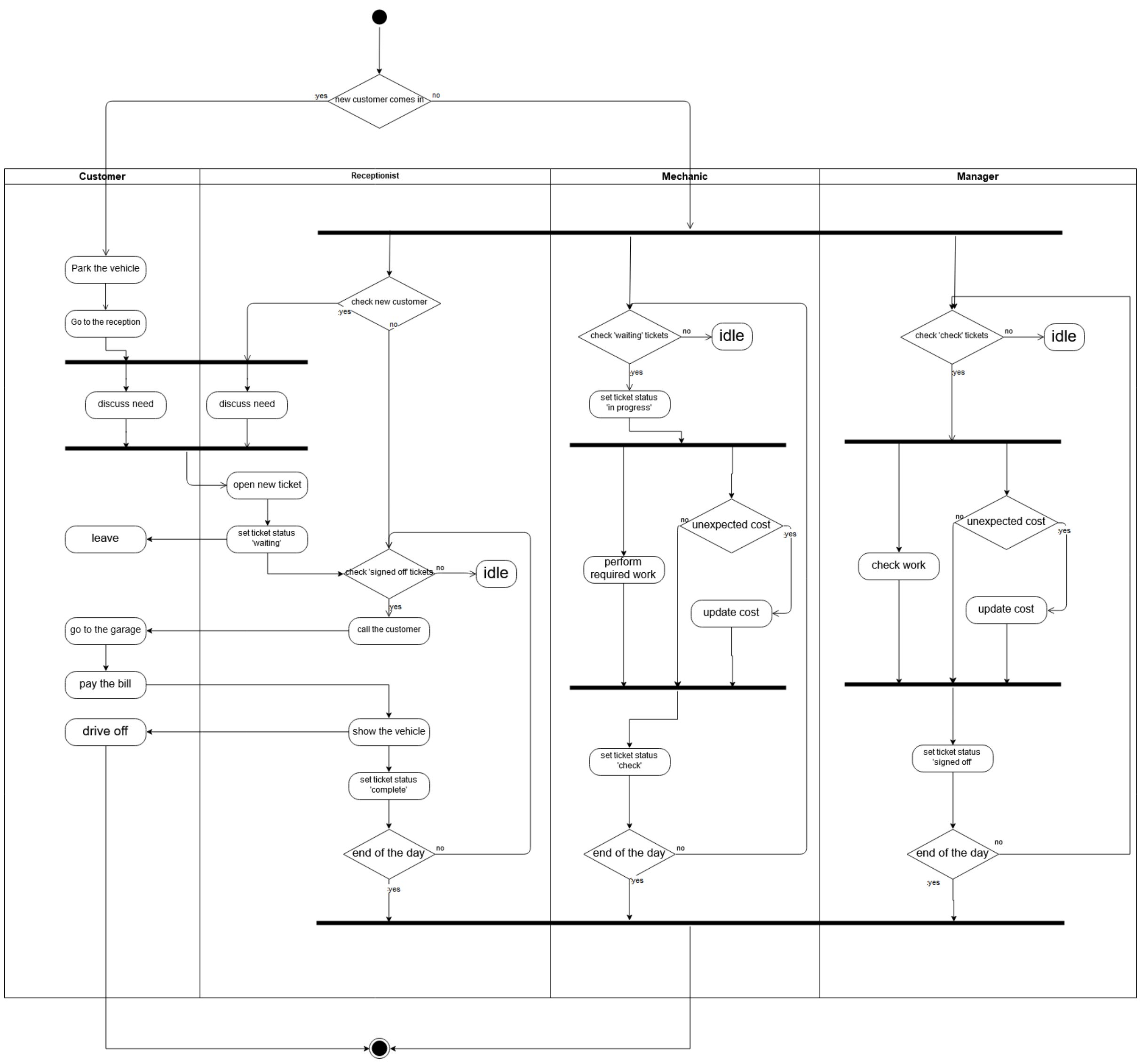
Manager	
Responsibilities	Collaborators
1.View ‘check’ tickets 2.Perform the work required as a mechanic when short-staffed 3.Update cost on the ticket 4.Check whether the work has been carried out to a good standard	Garage Mechanic Ticket
Comment: bad Cohesion: The responsibilities of ‘Manager’ are grouped because they perform similar functions Coupling: ‘Manager’ has the relation with 3 classes ‘Garage’, ‘Staff’ and ‘Ticket’. Thus, it is weak-cohesion and strong coupling.	

Ticket	
Responsibilities	Collaborators
1.Record all the information links the customer, their vehicle, the work required, the deadline, and the quoted price.	
Comment: Good Cohesion: ‘Ticket’ has the attributes and corresponding methods to inspect the attributes. Coupling: ‘Ticket’ has no relation with any other classes. Thus, it is strong-cohesion and loose coupling.	

Task 3 (20%) 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.



Task 5 (20%) Produce a UML activity diagram capturing the workflow of the garage.


new customer comes in
yes
no
Customer
Receptionist
Mechanic
Manager
Park the vehicle
Go to the reception
discuss need
discuss need
check new customer
yes
no
open new ticket
set ticket status 'waiting'
leave
check 'signed off tickets
no
idle
yes
call the customer
go to the garage
pay the bill
show the vehicle
drive off
set ticket status 'complete'
end of the day
no
yes
check 'waiting' tickets
no
idle
yes
set ticket status 'in progress'
unexpected cost
no
yes
perform required work
update cost
set ticket status 'check'
end of the day
no
yes
check 'check' tickets
no
idle
yes
unexpected cost
no
yes
check work
update cost
set ticket status 'signed off'
end of the day
no
yes