COMP201 Assignment

**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 | -People: Vector<Staff>  -TicketQueue: LinkedList<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 |  |
| Customer | -name: String  -telephone: String  -vehicle: String [] | +get\_telephone(): String  +get\_vehicles():String [] |
| Staff | -name: String  -garage: Garage | +update\_status(): void |
| Receptionist | -name: String | +openTicket(Ticket new\_ticket): void  +get\_next\_ticket(): Ticket  +call\_customer(): void  +update\_status(): void |
| Mechanic | -name: String | +get\_next\_ticket(): Ticket  +update\_status(): void  +perform\_task(): void  +update\_price(): void |
| Manager | -name: String | +get\_next\_ticket(): Ticket  +update\_status(): void  +perform\_task(): void  +update\_price(): void  +check\_work(): void |
| Ticket | -customer: Customer  -vehicle: Vehicle  -work: Work  -deadline: Date  -price: double  -status: String | +get\_status(): String  +get\_price(): double  +set\_status(): void  +set\_price(): void |
| Work | - workTypes: String[] | +get\_ workTypes(): String[] |
| Vehicle | - vehicleTypes: String[] | +get\_ vehicleTypes(): String[] |

**1b) List all potential inheritance relationships.**

|  |
| --- |
| Person |
| -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 |

|  |
| --- |
| Receptionist |
| ^-name: String |
| ^+update\_status():void  ^+get\_next\_ticket(): Ticket  +openTicket(Ticket new\_ticket): void  +call\_customer(): void |

|  |
| --- |
| Mechanic |
| ^-name: String |
| ^+update\_status(): void  ^+get\_next\_ticket(): Ticket  +perform\_task(): void  +update\_price(): void |

|  |
| --- |
| Manager |
| -name: String |
| ^+update\_status(): void  ^+update\_price(): void  ^+perform\_task(): void  ^+get\_next\_ticket(): Ticket  +check\_work(): void |

**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.

**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>  -TicketQueue: LinkedList<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) |
| Receptionist | -name: String | +openTicket(Ticket new\_ticket): void  +get\_next\_ticket(): Ticket  +call\_customer(): void  +update\_status(): void |
| Mechanic | -name: String | +update\_status(): void  +perform\_task(): void  +update\_price(): void |
| Manager | -name: String | +update\_status(): void  +perform\_task(): void  +update\_price(): void  +check\_work(): boolean |
| Ticket | -customer: Customer  -vehicle: Vehicle  -work: Work  -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 |

|  |  |
| --- | --- |
| Person | |
| Responsibilities | Collaborators |
| 1.Provide his/her name |  |

|  |  |
| --- | --- |
| Customer | |
| Responsibilities | Collaborators |
| 1.Provide his/her personal phone number  2.Provide his/her vehicle information | Person |

|  |  |
| --- | --- |
| Staff | |
| Responsibilities | Collaborators |
| 1.Access the data of Garage  2.Update ticket status | Person  Garage |

|  |  |
| --- | --- |
| Receptionist | |
| Responsibilities | Collaborators |
| 1.Open tickets for customer  2.View ‘signoff’ tickets  3.Call customers | Garage  Staff  Ticket  Customer |

|  |  |
| --- | --- |
| Mechanic | |
| Responsibilities | Collaborators |
| 1.View ‘waiting’ tickets  2.Perform the work required  3.Update cost on the ticket | Garage  Staff  Ticket |

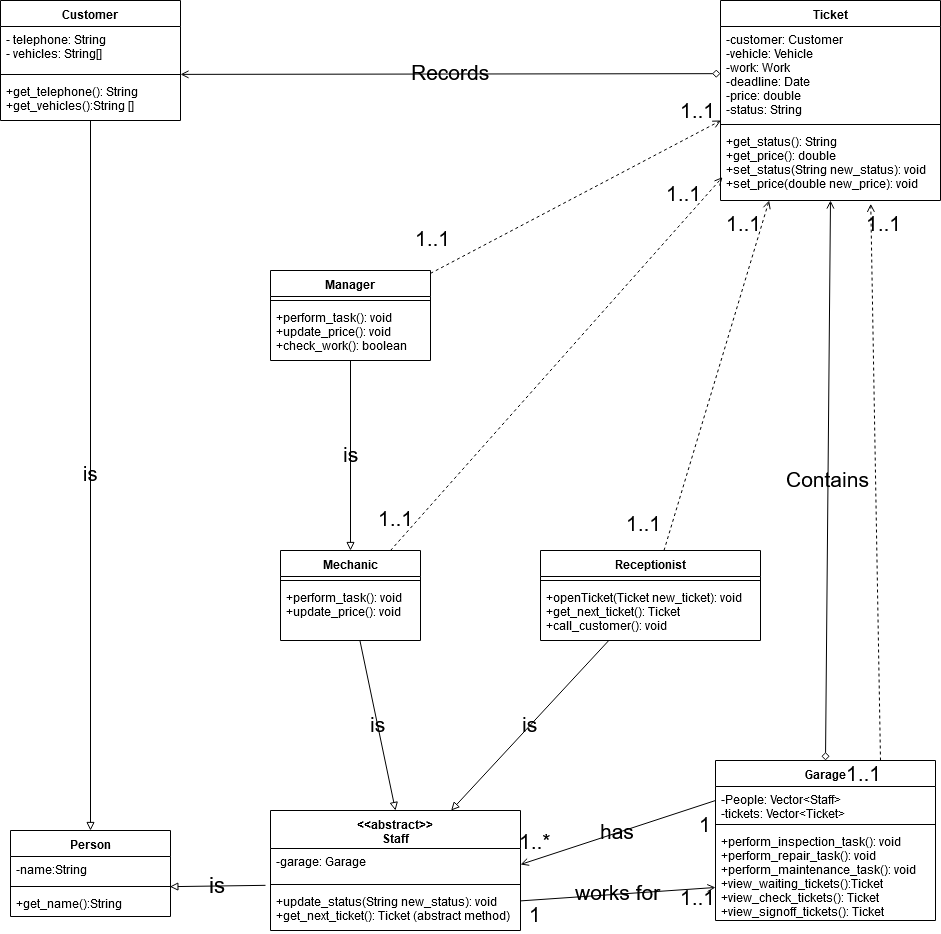
|  |  |
| --- | --- |
| Manager | |
| Responsibilities | Collaborators |
| 1.View ‘check’ tickets  2.Perform the work required as a mechanic when short-staffed  3.Update cost on the ticket  4.Check whether the work has been carried out  to a good standard | Garage  Mechanic  Ticket |

|  |  |
| --- | --- |
| Ticket | |
| Responsibilities | Collaborators |
| 1.Record all the information links the customer, their vehicle, the work required, the deadline, and the quoted price. |  |

**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.**

