Software Modelling - UML Use Case and UML Class diagrams

Table

Description automatically generated

1. Identify the use-cases for the software. Draw the \*\*UML use-case diagram \*\*and include supporting use-case descriptions. At-least 3 scenarios must be identified.
2. Identify the objects and their respective classes. Draw the \*\*UML class diagrams \*\*and include supporting descriptions to explain the relationships. At-least 4 classes and respective relationships must be identified.
3. For all the identified classes create \*\*Python classes \*\*with the constructor, attributes, and appropriate setter/getter methods. Each class must include at-least 5 attributes. Create objects of all the identified classes and use the object’s functions to populate and display the details.

Use Cases:

* Generating receipts
* Managing vehicle information
* Managing customer information
* Calculating taxes and discounts
* Repairing vehicles

Use Case Diagram:



Billing Management System

Cashier

Customer

Mechanic

|  |  |
| --- | --- |
| Use Case: | Generating Receipts |
| Trigger: | The costumer wants to get a receipt |
| Precondition: | The customer has received services from the auto repair shop |
| Main Scenario: |  |
| 1. | The employee selects the "Generate Receipt" use-case |
| 2. | The system displays a form to enter the customer information, vehicle information, and services provided |
| 3. | The employee enters the information and clicks "Generate Receipt" |
| 4. | The system generates a receipt and displays it on the screen |
| 5. | The employee prints the receipt and hands it to the customer. |
| Exceptions: |  |
| 3a. | 1. The employee enters invalid data for customer or vehicle information. 2. The system displays an error message and prompts the employee to enter valid data. |
| 3b. | 1. The employee selects a service that is not available. 2. the system displays an error message and prompts the employee to select a valid service. |
| 4a. | 1. There is an error during the receipt generation process. 2. The system displays an error message and prompts the employee to try again. |

Use Case Description 1:

Use Case Description 2:

|  |  |
| --- | --- |
| Use Case: | Manage Customer Information |
| Trigger: | The auto repair shop needs to update or add customer information to maintain accurate records. |
| Precondition: | The auto repair shop has customer information that needs to be managed. |
| Main Scenario: |  |
| 1. | The employee selects the "Manage Customer Information" use-case. |
| 2. | The system displays a list of existing customers. |
| 3. | The employee selects a customer from the list or adds a new customer. |
| 4. | The system displays a form to update or add the customer information. |
| 5. | The employee updates or adds the information and clicks "Save". |
| 6. | The system saves the information and displays a confirmation message. |
| Exceptions: |  |
| 3a. | 1. There are errors during the retrieval of existing customer information. 2. the system displays an error message and prompts the employee to try again. |
| 5a. | 1. There are errors during the update or addition of customer information. 2. The system displays an error message and prompts the employee to try again. |

Use Case Description 3:

|  |  |
| --- | --- |
| Use Case: | Calculate Taxes and Discounts |
| Trigger: | The employee needs to provide the customer with an accurate invoice. |
| Precondition: | The services have been provided to the customer. |
| Main Scenario: |  |
| 1. | The employee selects the "Calculate Taxes and Discounts" use-case. |
| 2. | The system retrieves the services provided and their prices. |
| 3. | The system calculates the total cost of the services. |
| 4. | The system calculates the taxes based on the total cost and the tax rate. |
| 5. | The system applies any applicable discounts to the total cost. |
| 6. | The system displays the total cost, taxes, discounts, and the final amount to the employee. |
| Exceptions: |  |
| 2a. | 1. There are errors during the retrieval of service prices or tax rates. 2. The system displays an error message and prompts the employee to try again. |
| 3a. | 1. There are errors during the calculation of the total cost, taxes, or discounts. 2. The system displays an error message and prompts the employee to try again. |
| 6a. | 1. There are errors during the display of the total cost, taxes, discounts, and final amount. 2. The system displays an error message and prompts the employee to try again. |

The 3 scenarios:

1. James W. Jones received services from the auto repair shop, and an employee generated a receipt for him.
2. An employee added a new customer to the auto repair shop's database and updated their information.
3. An employee calculated the taxes and discounts for the services provided to a customer, and the customer received a discounted final amount.

Classes:

1. Person:

* First Name: String
* Last Name: String
* Gender: ENUM
* Phone Number: String
* Date of Birth: Date

1. Customer:

* Email Address: String
* Billing Address: String
  + Customer1:
    - First Name: James
    - Last Name: W. Jones
    - Gender: Male
    - Phone Number: 816-897-9862
    - Date of Birth: 6-6-1996
    - Email Address: Jamesjones@gmail.com
    - Billing Address: Dubai

1. Cashier:

* Employee ID number: Integer
* Register number: Integer
* Manager name: String
  + Cashier1:
    - First Name: John
    - Last Name: Smith
    - Gender: Male
    - Employee ID number: 01234
    - Register number: 56789
    - Manager name: Tim A.

1. Mechanic:

* Areas of expertise: String
* Years of experience: Integer
  + Mechanic1:
    - First Name: Hans
    - Last Name: K
    - Gender: Male
    - Areas of expertise: Repair
    - Years of experience: 10

1. Vehicle:

* Make: ENUM
* Model: ENUM
* Year: Integer
* Color: ENUM
* Vehicle ID: String
  + Vehicle1:
    - Make: Nissan
    - Model: Altima
    - Year: 2014
    - Color: Silver
    - Vehicle ID: AD-89034

1. Service:

* Vehicle
* Service name: ENUM
* Service price: Integer
* Mechanic name: String
* Service status: ENUM
  + Service1:
    - Vehicle: Vehicle1
    - Service name: Diagnostics
    - Service price: 15
    - Mechanic name: Hans
    - Service status: Complete
  + Service2:
    - Vehicle: Vehicle1
    - Service name: Oil replacement
    - Service price: 120
    - Mechanic name: Hans
    - Service status: Complete
  + Service3:
    - Vehicle: Vehicle1
    - Service name: Oil filter parts
    - Service price: 35
    - Mechanic name: Hans
    - Service status: Complete
  + Service4:
    - Vehicle: Vehicle1
    - Service name: Tire replacement
    - Service price: 100
    - Mechanic name: Hans
    - Service status: Complete
  + Service5:
    - Vehicle: Vehicle1
    - Service name: Tire
    - Service price: 160
    - Mechanic name: Hans
    - Service status: Complete

1. Receipt

* Cashier name: String
* Date: Date
* Total services: Integer
* Taxes: Float
* Discount: Float
* Total cost: Integer
  + Receipt1:
    - Cashier name: John Smith
    - Date: March 13, 2022
    - Total services: 5
    - Taxes: 21.5
    - Discount: 11.5
    - Total cost: 440

UML class diagrams:

|  |
| --- |
| **Person** |
| firstName: String  lastName: String  gender: ENUM  phoneNumbr: String  dateOfBirth: Date |
| +getFirstName():String  +setFirstName(firstName:String)  +getLastName():String  +setLastName(lastName:String)  +setGender(gender:Gender)  +getGender():ENUM  +getPhoneNumber():String  +setPhoneNumbr(phoneNumber:String)  +getDateOfBirth():Date  +setDateOfBirth (dateOfBirth:String) |

|  |
| --- |
| **Gender** |
| male=1  female=2 |

|  |
| --- |
| **Customer** |
| emailAddress: String  billingAddress: String |
| +getEmailAddress():String  +setEmailAddress(emailAddress:String)  +getBillingAddress():String  +setBillingAddress(billingAddress:String) |

|  |
| --- |
| **Customer1: Customer** |
| firstName: “James”  lastName: “W. Jones”  gender: Gender.Male  phoneNumbr: 816-897-9862  dateOfBirth: [1996-6-6]  emailAddress: “Jamesjones@gmail.com”  billingAddress: “Dubai” |

|  |
| --- |
| **Cashier** |
| employeeIDNumber: Integer  registerNumber: Integer  managerName: String |
| +getEmployeeIdNumber():Integer  +setEmployeeIdNumber(employeeIdNumber:Integer)  +getRegisterNumber ():Integer  +setRegisterNumber (registerNumber:Integer)  +getManagerName():String  +setManagerName(ManagerName:String) |

|  |
| --- |
| **Cashier1: Cashier** |
| firstName: “Jane”  lastName: “Smith”  gender: Gender.Female  employeeIDNumber: 01234  registerNumber: 56789  managerName: “Tim A.” |

|  |
| --- |
| **Mechanic** |
| areasOfExpertise: String  yearsOfExperience: Integer |
| +getAreasOfExpertise():String  +setAreasOfExpertise(areasOfExpertise:String)  +getYearsOfExperience():Integer  +setYearsOfExperience(yearsOfExperience :Integer) |

|  |
| --- |
| **Mechanic1: Mechanic** |
| firstName: “Hans”  lastName: “K”  gender: Gender.Male  areasOfExpertise: “Repair”  Years of experience: 10 |

|  |
| --- |
| **Vehicle** |
| make: ENUM  model: ENUM  year: Integer  color: ENUM  vehicleId: String |
| +getMake():ENUM  +setMake(make:ENUM)  +getModel():ENUM  +setModel(model:ENUM)  +getYear():Integer  +setYear(year:Integer)  +getColor():ENUM  +setColor(color:ENUM)  +getVehicleId():String  +setVehicleId(vehicleId:String) |

|  |
| --- |
| **Make** |
| Nissan=1  Toyota=2 |

|  |
| --- |
| **Model** |
| Altima=1  Camry=2 |

|  |
| --- |
| **Color** |
| Silver=1  White=2 |

|  |
| --- |
| **Vehicle1: Vehicle** |
| make: Make.Nissan  model: Model.Altima  year: 2014  color: Color.Silver  vehicleId: “AD-89034” |

|  |
| --- |
| **Service** |
| serviceName: ENUM  servicePrice: Integer  mechanicName: String  serviceStatus: ENUM |
| +getServiceName():ENUM  +setServiceName(serviceName:ENUM)  +getServicePrice():Integer  +setServicePrice(servicePrice:Integer)  +getMechanicName():String  +setMechanicName(mechanicName:String)  +getServiceStatus():ENUM  +setServiceStatus(serviceStatus :ENUM) |

|  |
| --- |
| **ServiceName** |
| Diagnostics=1  OilReplacement=2  OilFilterParts=3  TireReplacement=4  Tire=5 |

|  |
| --- |
| **ServiceStatus** |
| Pending=1  In progress=2  Complete=3 |

|  |
| --- |
| **Service1: Service** |
| vehicleId: “AD-89034”  serviceName: ServiceName.Diagnostics  servicePrice: 15  mechanicName: “Hans K.”  serviceStatus: ServiceStatus.Complete |

|  |
| --- |
| **Service2: Service** |
| vehicleId: “AD-89034”  serviceName: ServiceName.OilReplacement  servicePrice: 120  mechanicName: “Hans K.”  serviceStatus: ServiceStatus.Complete |

|  |
| --- |
| **Service3: Service** |
| vehicleId: “AD-89034”  serviceName: ServiceName.OilFilterParts  servicePrice: 35  mechanicName: “Hans K.”  serviceStatus: ServiceStatus.Complete |

|  |
| --- |
| **Service4: Service** |
| vehicleId: “AD-89034”  serviceName: ServiceName.TireReplacement  servicePrice: 100  mechanicName: “Hans K.”  serviceStatus: ServiceStatus.Complete |

|  |
| --- |
| **Service5: Service** |
| vehicleId: “AD-89034”  serviceName: ServiceName.Tire  servicePrice: 160  mechanicName: “Hans K.”  serviceStatus: ServiceStatus.Complete |

|  |
| --- |
| **Receipt** |
| cashierName: String  date: Date  totalServices: Integer  taxes: Float  discount: Float  totalCost: Integer |
| +getCashierName():String  +setCashierName(cashierName:String)  +getDate():Date  +setDate(date:Date)  +getTotalServices():Integer  +setTotalServices(totalServices:Integer)  +getTaxes():Float  +setTaxes(taxes:Float)  +getDiscount():Float  +setDiscount(discount:Float)  +getTotalCost():Integer  +setTotalCost(totalCost:Integer) |

|  |
| --- |
| **Receipt1: Receipt** |
| Cashier name: “John Smith”  Date: [3- 13-2022]  Total services: 5  Taxes: 21.5  Discount: 11.5  Total cost: 440 |

Relationships:

Person is a superclass of Customer, Cashier, and Mechanic classes. Each of these classes has additional attributes and methods specific to their roles.

Customer is a Person

Cashier is a Person

Mechanic is a Person

Vehicle is associated with Service as each service is performed on a specific vehicle.

The Employee class is associated with the Receipt class through a dependency relationship, as an employee interacts with the system to generate a receipt.

# Enums  
from enum import Enum  
class Gender(Enum):  
 male = 1  
 female = 2  
class Make(Enum):  
 Nissan = 1  
 Toyota = 2  
class Model(Enum):  
 Altima = 1  
 Camry = 2  
class Color(Enum):  
 Silver=1  
 White=2  
class ServiceName(Enum):  
 Diagnostics = 1  
 OilReplacement = 2  
 OilFilterParts = 3  
 TireReplacement = 4  
 Tire = 5  
class ServiceStatus(Enum):  
 Pending = 1  
 Inprogress = 2  
 Complete = 3  
  
  
class Person:  
 def \_\_init\_\_(self, first\_name, last\_name, gender, phone\_number, date\_of\_birth):  
 self.first\_name = first\_name  
 self.last\_name = last\_name  
 self.gender = gender  
 self.phone\_number = phone\_number  
 self.date\_of\_birth = date\_of\_birth  
  
 def get\_first\_name(self):  
 return self.first\_name  
  
 def get\_last\_name(self):  
 return self.last\_name  
  
 def get\_gender(self):  
 return self.gender  
  
 def get\_phone\_number(self):  
 return self.phone\_number  
  
 def get\_date\_of\_birth(self):  
 return self.date\_of\_birth  
  
 def set\_first\_name(self, first\_name):  
 self.first\_name = first\_name  
  
 def set\_last\_name(self, last\_name):  
 self.last\_name = last\_name  
  
 def set\_gender(self, gender):  
 self.gender = gender  
  
 def set\_phone\_number(self, phone\_number):  
 self.phone\_number = phone\_number  
  
 def set\_date\_of\_birth(self, date\_of\_birth):  
 self.date\_of\_birth = date\_of\_birth  
  
 def \_\_str\_\_(self):  
 return f"Name: {self.first\_name} {self.last\_name}, Gender: {self.gender}, Phone: {self.phone\_number}, DOB: {self.date\_of\_birth}"  
  
  
class Customer(Person):  
 def \_\_init\_\_(self, first\_name, last\_name, gender, phone\_number, date\_of\_birth, email\_address, billing\_address):  
 super().\_\_init\_\_(first\_name, last\_name, gender, phone\_number, date\_of\_birth)  
 self.email\_address = email\_address  
 self.billing\_address = billing\_address  
  
 def get\_email\_address(self):  
 return self.email\_address  
  
 def get\_billing\_address(self):  
 return self.billing\_address  
  
 def set\_email\_address(self, email\_address):  
 self.email\_address = email\_address  
  
 def set\_billing\_address(self, billing\_address):  
 self.billing\_address = billing\_address  
  
 def \_\_str\_\_(self):  
 return f"{super().\_\_str\_\_()}, Email: {self.email\_address}, Billing Address: {self.billing\_address}"  
  
class Cashier(Person):  
 def \_\_init\_\_(self, first\_name, last\_name, gender, employee\_id\_number, register\_number, manager\_name):  
 super().\_\_init\_\_(first\_name, last\_name, gender, "",  
 "") # empty string values for phone\_number and date\_of\_birth  
 self.employee\_id\_number = employee\_id\_number  
 self.register\_number = register\_number  
 self.manager\_name = manager\_name  
  
 def get\_employee\_id\_number(self):  
 return self.employee\_id\_number  
  
 def get\_register\_number(self):  
 return self.register\_number  
  
 def get\_manager\_name(self):  
 return self.manager\_name  
  
 def set\_employee\_id\_number(self, employee\_id\_number):  
 self.employee\_id\_number = employee\_id\_number  
  
 def set\_register\_number(self, register\_number):  
 self.register\_number = register\_number  
  
 def set\_manager\_name(self, manager\_name):  
 self.manager\_name = manager\_name  
  
 def \_\_str\_\_(self):  
 return f"Name: {self.first\_name} {self.last\_name}, Gender: {self.gender}, Employee ID: {self.employee\_id\_number}, Register: {self.register\_number}, Manager: {self.manager\_name}"  
  
  
class Mechanic(Person):  
 def \_\_init\_\_(self, first\_name, last\_name, gender, areas\_of\_expertise, years\_of\_experience):  
 super().\_\_init\_\_(first\_name, last\_name, gender, "", "")  
 self.areas\_of\_expertise = areas\_of\_expertise  
 self.years\_of\_experience = years\_of\_experience  
  
 def get\_areas\_of\_expertise(self):  
 return self.areas\_of\_expertise  
  
 def set\_areas\_of\_expertise(self, areas\_of\_expertise):  
 self.areas\_of\_expertise = areas\_of\_expertise  
  
 def get\_years\_of\_experience(self):  
 return self.years\_of\_experience  
  
 def set\_years\_of\_experience(self, years\_of\_experience):  
 self.years\_of\_experience = years\_of\_experience  
  
 def \_\_str\_\_(self):  
 return f"{self.first\_name} {self.last\_name}, Gender: {self.gender}, Areas of Expertise: {self.areas\_of\_expertise}, Years of Experience: {self.years\_of\_experience}"  
  
class Vehicle:  
 def \_\_init\_\_(self, make, model, year, color, vehicle\_id):  
 self.make = make  
 self.model = model  
 self.year = year  
 self.color = color  
 self.vehicle\_id = vehicle\_id  
  
 def get\_make(self):  
 return self.make  
  
 def set\_make(self, make):  
 self.make = make  
  
 def get\_model(self):  
 return self.model  
  
 def set\_model(self, model):  
 self.model = model  
  
 def get\_year(self):  
 return self.year  
  
 def set\_year(self, year):  
 self.year = year  
  
 def get\_color(self):  
 return self.color  
  
 def set\_color(self, color):  
 self.color = color  
  
 def get\_vehicle\_id(self):  
 return self.vehicle\_id  
  
 def set\_vehicle\_id(self, vehicle\_id):  
 self.vehicle\_id = vehicle\_id  
  
 def \_\_str\_\_(self):  
 return f"Make: {self.make}, Model: {self.model}, Year: {self.year}, Color: {self.color}, Vehicle ID: {self.vehicle\_id}"  
  
class Service(Vehicle):  
 def \_\_init\_\_(self, make, model, year, color, vehicle\_id, service\_name, service\_price, mechanic\_name, service\_status):  
 super().\_\_init\_\_(make, model, year, color, vehicle\_id)  
 self.service\_name = service\_name  
 self.service\_price = service\_price  
 self.mechanic\_name = mechanic\_name  
 self.service\_status = service\_status  
  
 def set\_service\_name(self, service\_name):  
 self.service\_name = service\_name  
  
 def get\_service\_name(self):  
 return self.service\_name  
  
 def set\_service\_price(self, service\_price):  
 self.service\_price = service\_price  
  
 def get\_service\_price(self):  
 return self.service\_price  
  
 def set\_mechanic\_name(self, mechanic\_name):  
 self.mechanic\_name = mechanic\_name  
  
 def get\_mechanic\_name(self):  
 return self.mechanic\_name  
  
 def set\_service\_status(self, service\_status):  
 self.service\_status = service\_status  
  
 def get\_service\_status(self):  
 return self.service\_status  
  
 def \_\_str\_\_(self):  
 return f"Service(vehicle\_id={self.vehicle\_id}, service\_name={self.service\_name}, service\_price={self.service\_price}, mechanic\_name={self.mechanic\_name}, service\_status={self.service\_status})"  
  
class Receipt:  
 def \_\_init\_\_(self, cashier\_name, date, total\_services, taxes, discount, total\_cost):  
 self.\_cashier\_name = cashier\_name  
 self.\_date = date  
 self.\_total\_services = total\_services  
 self.\_taxes = taxes  
 self.\_discount = discount  
 self.\_total\_cost = total\_cost  
  
 def set\_cashier\_name(self, name):  
 self.\_cashier\_name = name  
  
 def get\_cashier\_name(self):  
 return self.\_cashier\_name  
  
 def set\_date(self, date):  
 self.\_date = date  
  
 def get\_date(self):  
 return self.\_date  
  
 def set\_total\_services(self, total\_services):  
 self.\_total\_services = total\_services  
  
 def get\_total\_services(self):  
 return self.\_total\_services  
  
 def set\_taxes(self, taxes):  
 self.\_taxes = taxes  
  
 def get\_taxes(self):  
 return self.\_taxes  
  
 def set\_discount(self, discount):  
 self.\_discount = discount  
  
 def get\_discount(self):  
 return self.\_discount  
  
 def set\_total\_cost(self, total\_cost):  
 self.\_total\_cost = total\_cost  
  
 def get\_total\_cost(self):  
 return self.\_total\_cost  
  
 def \_\_str\_\_(self):  
 return f"Cashier: {self.\_cashier\_name}\nDate: {self.\_date}\nTotal Services: {self.\_total\_services}\nTaxes: {self.\_taxes}\nDiscount: {self.\_discount}\nTotal Cost: {self.\_total\_cost}"  
  
# Create a Customer object  
customer1 = Customer("James", "W. Jones", Gender.male.name, "816-897-9862", "1996-06-06", "Jamesjones@gmail.com", "Dubai")  
print(customer1)  
  
# Create a Cashier object  
cashier1 = Cashier("Jane", "Smith", Gender.female.name, "01234", "56789" , "Tim A.")  
print(cashier1)  
  
# Create a Mechanic object  
mechanic1 = Mechanic("Hans", "K", Gender.male.name, "Repair", 10)  
print(mechanic1)  
  
# Create a Vehicle object  
vehicle1 = Vehicle(Make.Nissan.name, Model.Altima.name, 2010, Color.Silver.name, "AD-89034")  
print(vehicle1)  
  
  
  
# Create a Receipt object  
receipt = Receipt("John Smith", "2022-03-13", 3, 21.5, 11.5, 440)  
print(receipt)

Result:

Name: James W. Jones, Gender: male, Phone: 816-897-9862, DOB: 1996-06-06, Email: Jamesjones@gmail.com, Billing Address: Dubai

Name: Jane Smith, Gender: female, Employee ID: 01234, Register: 56789, Manager: Tim A.

Hans K, Gender: male, Areas of Expertise: Repair, Years of Experience: 10

Make: Nissan, Model: Altima, Year: 2010, Color: Silver, Vehicle ID: AD-89034

Cashier: John Smith

Date: 2022-03-13

Total Services: 3

Taxes: 21.5

Discount: 11.5

Total Cost: 440

Summary:

* Use-case diagrams: Created a use-case diagram to represent the different actions and roles involved in a typical garage service.
* Class diagrams: Created a class diagram to represent the relationships between different objects involved in the garage service system. The classes included Customer, Vehicle, Service, Cashier, and Receipt.
* Python classes: Implemented Python classes for each of the identified classes, including their constructor, attributes, and appropriate getter and setter methods.
* Object creation and manipulation: Created objects of the identified classes and used their functions to display their information.