CLASS DIAGRAMS

CONCEPT OF CLASS DIAGRAMS

A class is a blueprint for creating objects (a particular data structure or entity) with attributes(properties) and methods (operations or behavior).

A class diagram is a static structure diagram in UML that describes the structure of a system by showing its classes, their properties and behavior, and the relationships between the classes.

CLASS STRUCTURE

- **Class properties** are variables that belong to the class. They can hold values like strings, numbers, and Booleans. For example, a class named STUDENT will have properties such as name, age and grade.
- **Class methods** are functions that a class can carry out or can be carried out on. The define the behavior of classes. For example, a student can be admitted, promoted or dismissed.

CLASS RELATIONS

- **Inheritance** is a mechanism where a class derives (inherits) some or all of its properties and methods from another class. The class which is being inherited from is called the base or super class and the one which inherits is the derived or sub class. For example, two classes, STUDENT and LECTURER can inherit properties such as name, date of birth and contact from another class named PERSON. Relationship is indicated in the UML by an arrow with a triangular arrowhead pointing towards the generalized class.
- **Association** indicates two classes with are linked. The link (relation) is form when one class has a property of another. For example, a STUDENT class and a COURSE class can be linked when the student has courses as one of its properties. This relation is shown by a line joining the classes. Sometimes, the relation is direction which is represent with as an arrow pointing from one class to the other.

- **Aggregation** occurs when one class is made up of another (or an array of another class). For example, a class named DEPARTMENT is made up of an array of Students. In this case, DEPARTMENT aggregates STUDENT, as STUDENT is part of DEPARTMENT. The relation is represented in the class diagram by an arrow with a white diamond head, where the arrow points to class which aggregates (DEPARTMENT).
- **Composition** is an aggregation where if the class which aggregates is removed, the other class ceases to exist. That is the life time of the class which is being aggregated depends on the class which aggregates. For example, if the STUDENT ceases to exist once DEPARTMENT is removed, the DEPARTMENT is composed of STUDENT. In simple terms, DEPARTMENT controls the life time of STUDENT. This is represented in the same way as an aggregation but with a black diamond head.

MULTIPLICITY

The multiplicity of a relationship is indicated by a number (or *) placed at the end of an association.

- **One-to-one:** For example, for a class named HOD (which also inherits from PERSON), one HOD will be associated to one department. That is, one HOD heads one department and one department is headed by one HOD.
- **One-to-many/many-to-one:** For example, DEPARTMENT is associated with STUDENT by one-to-many. That is, one department can have many students, but one student can be only under one department. Or a man can marry many women but a woman can be married to only one man.
- **Many-to-many:** This is a **one-to-one and a many-to-one** association at the same time. For example, the association between STUDENT and COURSES where many students can register many courses. That is, one student can register many courses and many students can register one course, thus a one-to-many and many-to-one relationship.

These relations are shown on the class diagram below.

CLASS DIAGRAM FOR THE BUS RESERVATION SYSTEM

CLASSES:

1. **USER (superclass):** The system is made up users, so this class holds attributes and behavior for users such as employees and admins.
   Properties: user_id, user_name, user_address, user_contact, user_email, user_gender, user_password.
   Methods: *create_account(), update_account(), log_in(), log_out(), delete_account()*

2. **ADMIN (subclass of USER):** This class is for the administration.
   Properties: admin_id, admin_name, admin_address, admin_contact, admin_email, admin_gender, admin_password. These properties are inherited from the USER class (an admin is a user).
   Methods: the inherit all the methods from USER and have additional methods such as; *add_manager(), update_manager(), remove_manager(), pay_manager(), _create_station(), update_station(), remove_station().*

3. **EMPLOYEE (subclass of USER and super class)**
   Properties: It inherits all the properties from USER and has other properties such as; employee_station (working station of employee), monthly_salary and employee_status (job type and position).
   Methods: It inherits all methods from USER.

4. **CUSTOMER (subclass of USER)**
   Properties: It inherits all the properties from USER and has other properties such as; customer_status (sick, old, injured, pregnant, child, VIP, non-VIP, adult), ticket (customer's tickets)
   Methods: It inherits all the methods from USER and has other methods such as; *add_ticket(), update_ticket(), check_ticket(), delete_ticket().*

5. **VENDOR (subclass of EMPLOYEE):** Employees who sell tickets at the station.
   Properties: It inherits all the properties from EMPLOYEE.
   Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *create_ticket(), search_ticket(), sign_ticket().*

6. **CASHIER (subclass of EMPLOYEE)**
   Properties: It inherits all the properties from EMPLOYEE.
   Methods: It inherits all the methods from EMPLOYEE and has other method; *deposit_money().*

7. **TICKET**
   Properties: ticket_no, booking_date, booking_station, departure_date, destination_station, duration, ticket_type, bus_no, seat_no, amount, ticket_status (upcoming, ongoing, expired), payment_details.
   Methods: *modify_ticket(), remove_ticket()*

8. **NON_CUSTOMER_TICKETS.** Holds tickets for those passengers who are not customers. That is, those passengers who buy tickets at the station rather than book them online.
   Properties: tickets, vendors
   Methods: *add_ticket(), search_ticket(), delete_ticket().*

9. **DRIVER (subclass of EMPLOYEE)**
   Properties: It inherits all the properties from EMPLOYEE and an additional property; bus_number..
   Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *assign_bus(), change_bus(), remove_bus(), check_depaarture().*

10. **ATTENDANT (subclass of EMPLOYEE)**
    Properties: It inherits all the properties from EMPLOYEE and has bus_number..
    Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *assign_bus(), change_bus(), remove_bus(), check_depaarture().*

11. **BUS**

    Properties: bus_no, bus_routes, no_seats, available_seats, bus_type (VIP, non-VIP), bus_status(good, on repairs, damaged), travel_route, bus_description, bus_station.

    Methods: *add_bus(), remove_bus(), modify_bus(), check_bus()*

12. **STATION**

    Properties: station_id, station_location, station_town, departure_time (departure times), destination_stations, open_time, employees.

    Methods: *add_station(), remove_station(), update_station().*

13. **ROUTE:**

    Properties: route_id, town1, town2_id, duration, departure_times.

    Methods: create_route(), update_route(), delete_route().

14. **EMPLOYEE_MANAGER (subclass of EMPLOYEE):**

    Properties: It inherits all the properties from EMPLOYEE and an additional property; employees (all the employees been managed).

    Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *add_employee, update_employee, search_employee, pay_employee, remove_employee.*

15. **STATION_MANAGER (subclass of EMPLOYEE)**

    Properties: It inherits all the properties from EMPLOYEE and an additional property; station.

    Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *update_station, remove_station.*

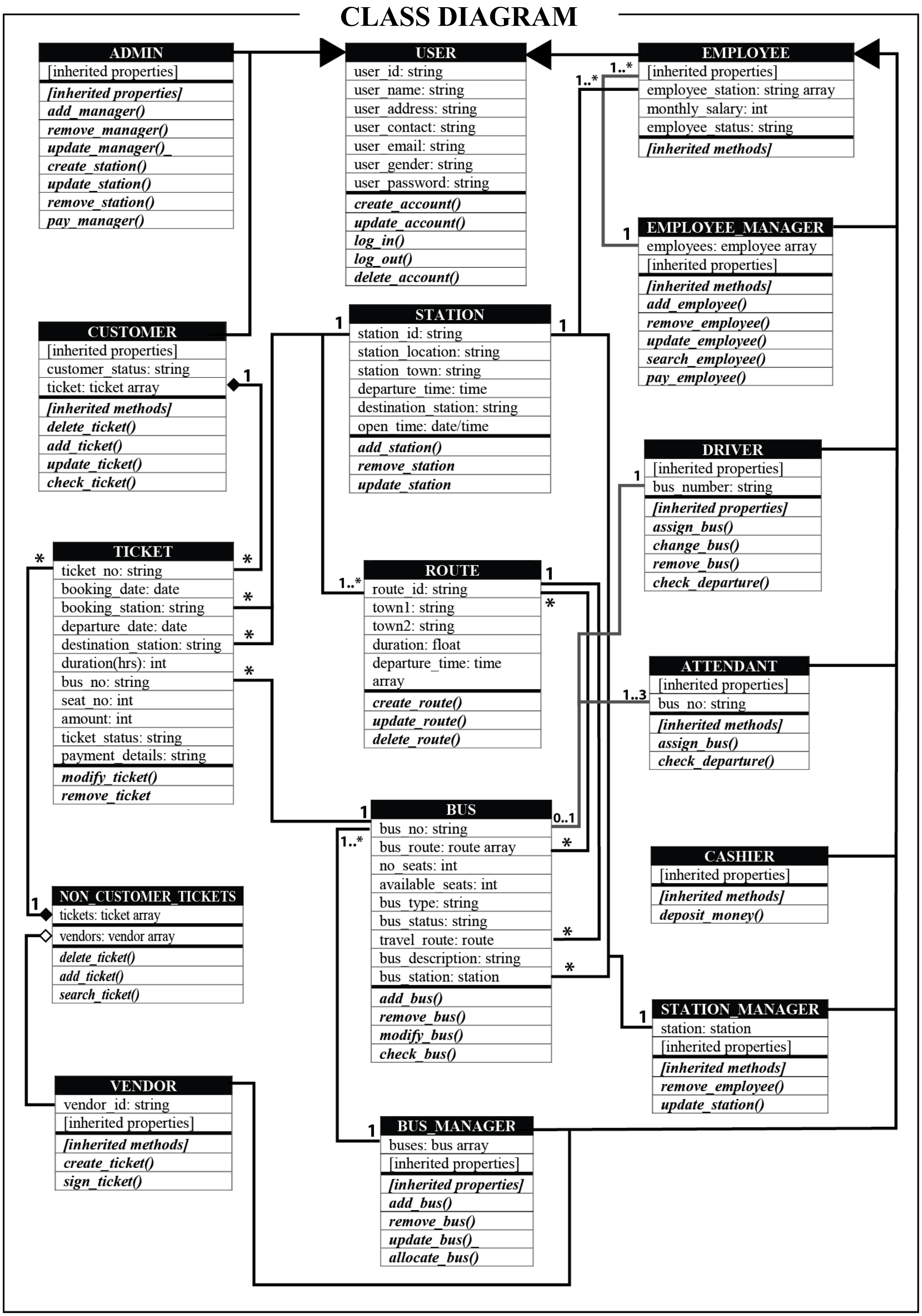
16. **BUS_MANAGER (subclass of EMPLOYEE)**

    Properties: It inherits all the properties from EMPLOYEE and an additional property; bus_array.

    Methods: It inherits all the methods from EMPLOYEE and has other methods such as; *add_bus(), update_bus(), remove_bus(), allocate_bus().*

These relations are shown on the class diagram below.

CLASS DIAGRAM



RELATIONSHIPS BETWEEN CLASSES

- **ADMIN, EMPLOYEE** and **CUSTOMER** (subclasses) all inherit from **USER** (superclass)
- **EMPLOYEE_MANAGER, DRIVER, ATTENDANT, CASHIER, STATION_MANAGER, BUS_MANAGER** and **VENDOR** (subclasses) inherit their properties and methods from **EMPLOYEE** (super class)
- **CUSTOMER** is composed of **TICKET.** That is, a customer can have an array of (many tickets) and a tickets life time depends on the customer. One customer can have many tickets, but one ticket can only belong to one customer.
- **NON_CUSTOMER_TICKETS** is also composed of **TICKET.** This class, has many tickets bought at the station by the passengers which are not customers. It is also composed of **VENDOR** as it has many vendors. But unlike TICKET, VENDOR's lifetime does not depend on this class.1 non_customer_ticket object can hold many tickets but one ticket can be found in only one of them. Many vendors can sell tickets in many of these objects.
- The association between **TICKET** and **STATION** (2 associations) is one to many as many tickets can have many booked or sold in one station and but one ticket is booked only in one station. For the second association, many tickets can have one destination station, but one ticket has only one destination station.
- **TICKET** associates with **BUS** by many to one. Many tickets can be booked to one bus but one ticket can be booked only to one bus.
- **STATION** and **ROUTE** have a 1 to many relationship where one station can have many routes. A route is between two towns.
- **STATION** has a 1 to many relationship with **EMPLOYEE.** One station must have at least one employee.
- **STATION_MANAGER** and **STATION** is one to one because one station manager manages only one station and one station is managed by only one station manager.
- **STATION** associates with **BUS** by many to one where many bus can be assign to one station but one bus belongs only to one station.

- **BUS_MANAGER** associates **BUS** by one to many as one bus manager can manager at least one bus but a bus is managed by just one bus manager.
- **BUS** and **ROUTE** have a many to many association, where many buses can use many routes. That is, one bus can use many routes and one route can be used by many buses. Another association between BUS and ROUTE is many to one. One bus can travel only through one road at a time but many buses can use one road at a time.
- **BUS** and **DRIVER** are zero/one to one. That is, at moat on bus can be assigned to a driver and a bus can be assigned to only one driver.
- **BUS** and **ATTENDANT** are associated by zero/one to one/three attendants. A bus must have at least one and at most three attendants and an attendant can be assigned to at most one bus.
- **EMPLOYEE_MANAGER** and **EMPLOYEE** is one to many. An employee manager manages at least one employee and one employee is managed by just one employee manager.

References

- abouml. (n.d.). *A Brief Overview of UML*, 407-418.
- Bell, D. (2003). UML basics: An introduction to the Unified Modeling Language. *UMLBasics*, 1-11.
- Padmanabhan, B. (2012). UNIFIED MODELING LANGUAGE (UML) OVERVIEW. *UML Diagrams*, 1-10.
- Reddy, C. M., Geetha, D. E., K. S., Kumar, T. S., & Kanth, K. R. (2011). General Methology for developing UML modells from UI. 1-16.