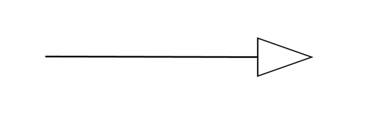
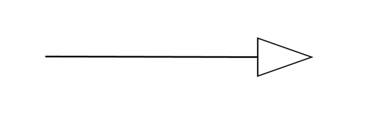
**UML class diagram:**

|  |
| --- |
| Employee |
| -department: str  -jobTitle: str  -basicSalary: int |
| Beth yw Diagram UML: Gan gynnwys Saeth a Symbol Diagram UML+displayDetails(): str |

|  |
| --- |
| Person |
| -id: int  -name: str  -dob: datetime  -age: int  -passportDetails: str |
| +displayDetails(): str |



0..\*

|  |
| --- |
| Manager |
| -salespersons: List[Salesperson] |
| +addSalesperson(salesperson: Salesperson)  +removeSalesperson(salesperson: Salesperson)  +displayDetails(): str |

|  |
| --- |
| Salesperson |
| -sales: List[Sale] |
| +addSale(sale: Sale)  +calculateCommission(): int  +displayDetails(): str |

1

0..\*



1



0..\*

|  |
| --- |
| Sale |
| -salesperson: Salesperson  -car: Car  -salePrice: int  -saleDate: datetime.date |
| +\_\_str\_\_(): str |

|  |
| --- |
| Car |
| -id: int  -name: str  -price: int  -type: str  -fuelCapacity: int  -maxSpeed: int  -color: str |
| +displayDetails(): str |

1

|  |
| --- |
| Application |
| -employees  -cars |
| +showEmployeeDetails  +showCarDetails  +showSalesDetails |



1

1

1

0..\*

**Description:**

* Application has a 0-to-many relationship with Employee, as an application instance can manage multiple employees (which includes Managers and Salespersons).
* Application has a 0-to-many relationship with Car, as an application instance can manage multiple cars.
* Manager has a 0-to-many aggregation relationship with Salesperson, as one manager can supervise multiple salespersons, but a salesperson might not have a manager assigned.
* Salesperson has a 0-to-many relationship with Sale, as one salesperson can have multiple sales, but it's also possible that a salesperson hasn't made any sales.
* Sale has a 1-to-1 relationship with Car, as each sale is linked to a single car.

Inheritance relationships:

* Employee inherits from the Person class.
* Manager and Salesperson inherit from the Employee class. This indicates that these classes share common attributes and behaviors with the Employee class.

Aggregation relationship:

* The Manager class has an aggregation relationship with the Salesperson class. This means that a Manager object can have multiple Salesperson objects associated with it. If a Manager object is destroyed, the associated Salesperson objects are not destroyed but can exist independently.

Association relationships:

* The Sale class is associated with both the Salesperson and Car classes. This means that a Sale object is connected to a Salesperson object (who made the sale) and a Car object (which was sold).
* The Application class is associated with both the Employee and Car classes. This means that the Application object interacts with multiple instances of Employee and Car objects.

**Assumptions:**

* An Employee object can be a Manager or a Salesperson.
* A Manager can have multiple Salesperson objects associated with them.
* A Salesperson can have multiple Sale objects associated with them.
* Each Sale object is associated with exactly one Salesperson and one Car.
* The Application object interacts with multiple Employee and Car objects but does not own them.