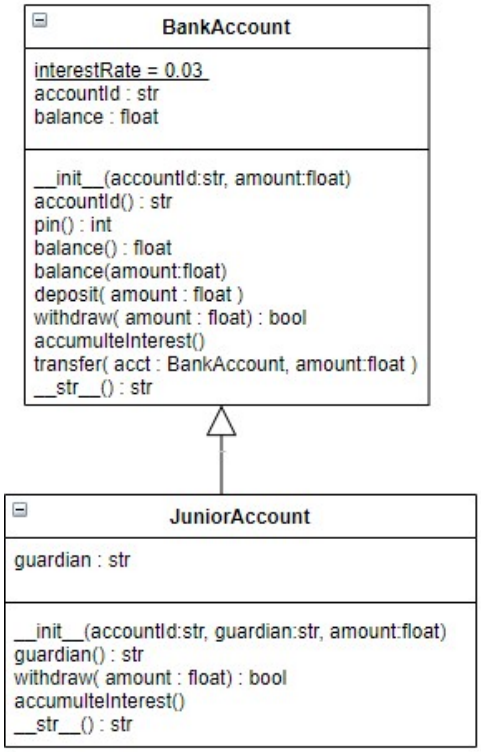
Lab 3 (Inheritance)

1. Given the following BankAccount and JuniorAccount class:



The code for BankAccount class is given as follows:

```
class BankAccount:
    _interestRate = 0.03

    def __init__(self, id,  amount):
        self._accountId = id
        self._balance = amount

    @property
    def accountId(self):
        return self._accountId

    @property
    def balance(self):
        return self._balance

    @balance.setter
    def balance(self,  amt):
        self._balance = amt

    def deposit(self, amount):
        self._balance += amount

    def withdraw(self, amount):
        if  amount <= self._balance:
```

```
            self._balance -= amount
            return True
        return False

    def transfer(self, ba, amount):
        if  self.withdraw(amount):
            ba.deposit(amount)
            return True
        return False

    def accumulateInterest(self):
        self._balance += self._balance  * type(self)._interestRate

    def __str__(self):
        return f'{self._accountId} {self._balance:.2f}'
```

a. A JuniorAccount is a BankAccount. A JuniorAccount has an additional guardian as instance variable. Write the Junior Account class as follows:
   i. The JuniorAccount class as a subclass of BankAccount
   ii. A constructor that has the id, guardian and balance.
   iii. Get property for guardian.
   iv. A withdraw method that limits withdrawal to maximum of 50 dollars. This method also returns true if the withdrawal is successful and false otherwise.
   v. An accumulateInterest method that computes adds the interest amount to the balance. Junior account holders earn 1% more.
   vi. The str method that returns all information of the junior account.

b. Write an application to create a JuniorAccount object. Test the deposit, withdrawal and accumuateInterest methods.

c. Identify, where applicable, methods that exhibit method overriding by replacement/refinement.

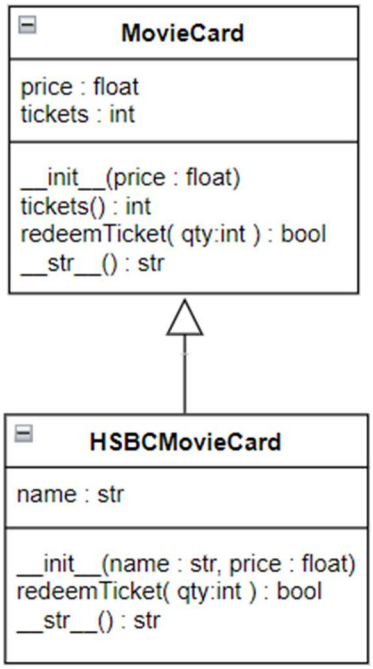
2. Modify the BankAccount and JuniorAccount classes in Question 1.

   a. Modify the constructor, the withdraw and deposit methods in BankAccount such that if the amount is not supplied, a standard amount of 20 dollars is assumed.

   b. Modify the withdraw method in the JuniorAccount so that it does not limit the withdrawal provided the guardian is present with the junior account. The withdraw method has 2 parameters: the guardian's name and the amount. Allow withdraw only if the guardian's name parameter matches the instance variable guardian. It has the same return type as part iv of Question 1. If the guardian is present with the junior account holder, there is no withdrawal limit of 50 dollars.

   c. After modifying the withdraw method in part (b), discuss if any modification is required to the transfer method. In which class should we make the changes, if any.

3. The following UML describes a MovieCard class and a HSBCMovieCard class.



A cinema sells movie cards. For $70, a movie goer can watch 10 movies and for $100, 15 movies. Write a MovieCard class that represents a movie card with the following:

a. A constructor that has price as parameter to initialize the instance variable price. Assume that the parameter value will be 70, or 100. Another instance variable is used to store the number of tickets based on the price, i.e. self_tickets=10 or 15 depending on the price.
b. A property tickets that returns the number of tickets remaining.
c. A redeemTicket method with qty as parameter. Default is 1. Each redemption is for maximum of 2 tickets only. If there are still tickets left, subtract the number of tickets and return True, otherwise return False.
d. A __str__ method, that returns the price and the tickets remaining in this format: Ticket price: $70, tickets remaining: 2

Test this class by creating a movie object and redeem some tickets. Print the tickets remaining.

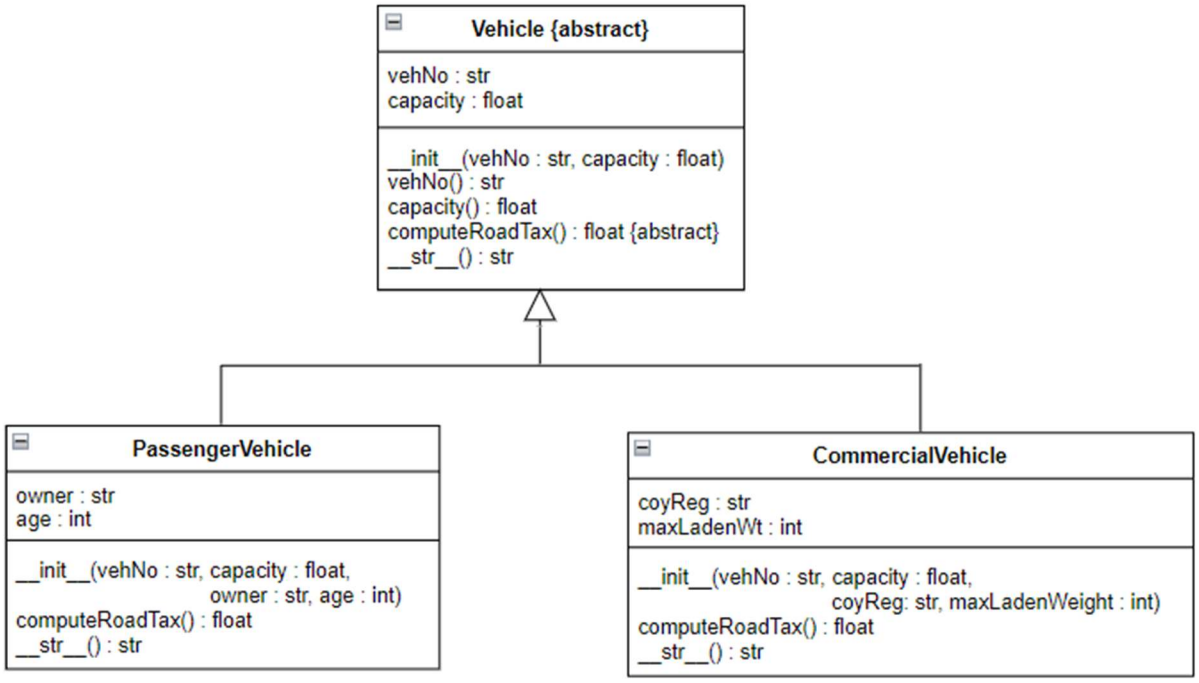
4. A HSBCMovieCard is also a MovieCard, but with more privileges for HSBC credit card holders. Card holders get 2 more tickets for the same prices. Card holders are able to redeem up to a maximum of 4 tickets each time. Write the HSBCMovieCard as a subclass of MovieCard with the following:

a. A constructor that has the credit card holder name and the price. Card holders get 2 more tickets, i.e. 12 for $70 and 17 for $100.

b. A redeemTicket method with qty as parameter. Default is 1. HSBCMovieCard holders can redeem up to 4 tickets at one time as long as there are tickets available in the card. If successful, it returns True, otherwise returns False.
c. A __str__ method that returns the same information as the MovieCard, but with extra information of the name.

Test out this class by creating a HSBCMovieCard object, redeem some tickets and print the status.

5. The Registry of Vehicle classifies all Commercial and Passenger vehicles as Vehicles.



You are required to write a program that computes road tax for such vehicles. Write the following:

a. Vehicle class
It has a vehicle number and engine capacity. Besides the standard constructor and accessor methods, it has a computeRoadTax method that returns the amount of road tax to pay. There is no implementation for this method yet. Include a str method that returns the road tax as well as the vehicle number and engine capacity.

b. PassengerVehicle class
This class inherits from the Vehicle class. It has owner and age of the person who owns the vehicle. Road tax for passenger cars is computed based on $1 per cc of the engine capacity. For owners who are 55 and above, a 10% discount is given. The str method should print the owner's detail before the vehicle's details.

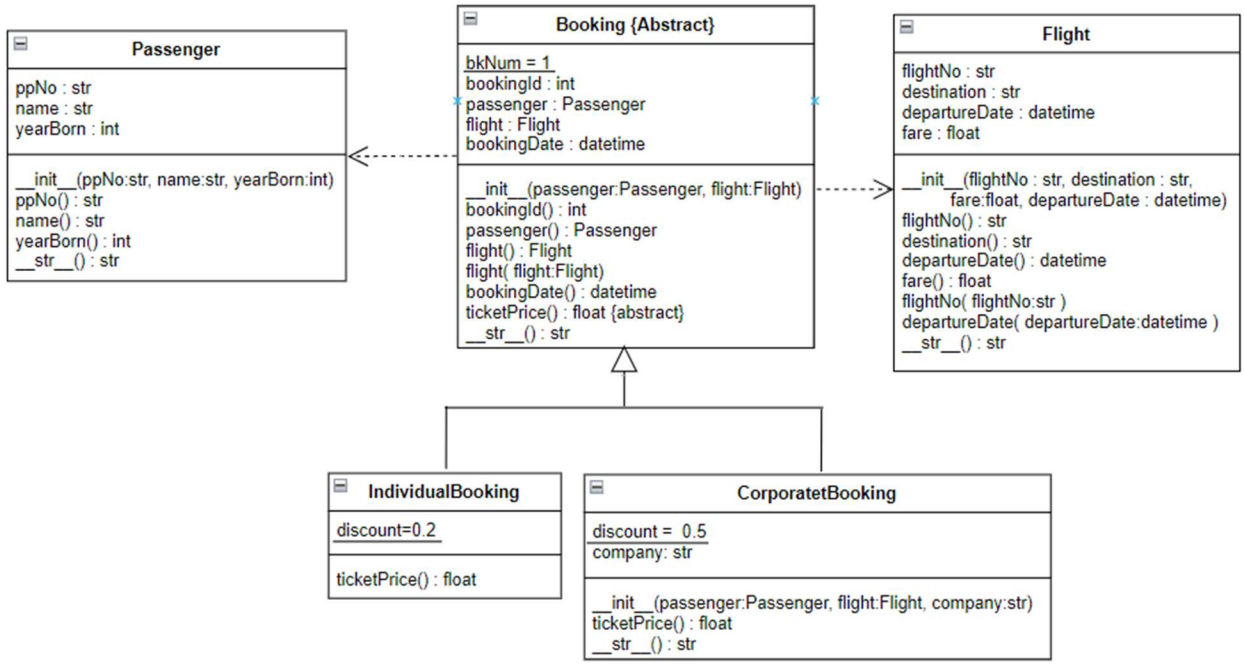
c. CommercialVehicle class

This class inherits from the Vehicle class. It has company registration number(String) and maxLadenWeight as attributes. Road tax is computed based on $1 per cc if the maxLadenWeight(int) is 3 metric tonnes or below and $1.50 per cc otherwise. The str method should print the company detail before the vehicle's details.

d. Write an application to test the PassengerVehicle and CommercialVehicle class. Add 2 Passenger cars and 2 commercial cars to a list. Use a loop to display the road tax to pay for each vehicle.

Indicate the statement in your code that exhibits polymorphism.

6. The Airline has individual and corporate bookings. The class diagram is given as follows:



Note the changes to the classes:

- Passenger class – additional instance variable yearBorn and property for yearBorn.
- Flight class – additional instance variable fare and property for fare.
- The Booking class is an abstract class. Other additional instance variable includes the bookingDate and property for bookingDate. It has an abstract method ticketPrice() which currently has no implementation.
- The IndividualBooking class is also a Booking. No additional instance variable. The class implements the ticketPrice() method. For individual booking, the ticket price for juniors below 18 and seniors 60 and above are entitled to 20% discount as declared in the class variable. Return the ticket price.

- The CorporateBooking class is also a Booking. It has the company name as additional instance variable. All corporate bookings are entitled to 50% discount as indicated in the class variable. Return the ticket price.

Test the Booking classes.