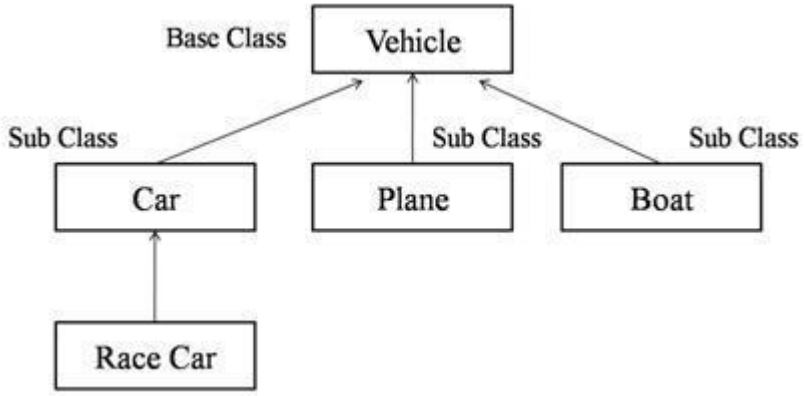
Лаб 7.1 Создать пустые классы **Vehicle, Car, Plane, Boat, RaceCar**, которые находятся в следующей иерархии:



Класс **Vehicle** является базовым классом, от которого наследуются все остальные.

Необходимо написать только определение классов

Лаб 7.2 Создайте базовый класс Vehicle, у которого есть:

1. конструктор __init__, принимающий название транспортного средства, его максимальную скорость и пробег. Их необходимо сохранить в атрибуты экземпляра name, max_speed и mileage соответственно
2. метод display_info , который печатает информацию в следующем виде:
3. Vehicle Name: {name}, Speed: {max_speed}, Mileage: {mileage}

Затем создайте подкласс Bus , унаследованный от Vehicle. Оставьте его пустым

```
bus_99 = Bus("Ikarus", 66, 124567)
bus_99.display_info() #печатает "Vehicle Name: Ikarus, Speed: 66, Mileage: 124567"
```

Лаб 7.3 Создайте базовый класс Person, у которого есть:

1. конструктор __init__, который должен принимать на вход имя и записывать его в атрибут name

2. метод get_name, который возвращает атрибут name;
3. метод is_employee, который возвращает False

Затем создайте подкласс Employee , унаследованный от Person. В нем должен быть реализован:

1. метод is_employee, который возвращает True

```
emp1 = Person("Resti")
print(emp1.get_name(), emp1.is_employee()) # Resti False

emp2 = Employee("Bim")

print(emp2.get_name(), emp2.is_employee())# Bim True
```

Лаб 7.4 Создайте класс **NewInt**, который унаследован от целого типа int, то есть мы будем унаследовать поведение целых чисел и значит экземплярам нашего класса будут поддерживать те же операции, что и целые числа.

Дополнительно в классе **NewInt** нужно создать:

- метод **repeat**, который принимает одно целое положительное число n (по умолчанию равное 2), обозначающее сколько раз нужно продублировать данное число. Метод repeat должен возвращать новое число, продублированное n раз (см пример ниже);
- метод **to_bin**, который возвращает двоичное представление числа в виде числа (может пригодиться функция bin)

```
a = NewInt(9)
print(a.repeat())  # печатает число 99
d = NewInt(a + 5)
print(d.repeat(3)) # печатает число 141414
b = NewInt(NewInt(7) * NewInt(5))
print(b.to_bin()) # печатает 100011 - двоичное представление числа 35
```

Лаб 7.5 Создайте базовый класс Person, у которого есть:

1. метод __init__, принимающий имя и возраст человека. Их необходимо сохранить в атрибуты экземпляра nameи age соответственно
2. метод display_person_info , который печатает информацию в следующем виде:

   Person: {name}, {age}

Затем создайте класс Company , у которого есть:

1. метод __init__, принимающий название компании и город ее основания. Их необходимо сохранить в атрибуты экземпляра company_name и location соответственно
2. метод display_company_info , который печатает информацию в следующем виде:

   Company: {company_name}, {location}

И в конце создайте класс Employee , который:

1. унаследован от классов Person и Company
2. имеет метод __init__, принимающий название имя человека, его возраст, название компании и город основания. Необходимо делегировать создание атрибутов nameи age классу Person , а атрибуты company_name и location должен создать класс Company

После множественного наследования у экземпляров класса Employee будут доступны методы родительских классов

```
emp = Employee('Jessica', 28, 'Google', 'Atlanta')

emp.display_person_info()

emp.display_company_info()
```