Name: Steve Hommy

Pair: -

Amount of completed tasks: 7

Which tasks were left undone or incomplete: 0

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

This exercise was easy for me because I have worked with these before. Only one thing gave me a headache and it was finding the txt file path.

Doing this exercise, I learned…

Using os module

I am still wondering…

-

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

I pretty much understood everything

Test report

Write the test report yourself to each coding task (task number, input/action, desired output and then the testing evidence (actual output)). Add rows if necessary. Include answers to theoretical questions and pseudocode to this return document as well in addition to code screen captures. Actual output can be a screen capture of the terminal showing the output.

Task	**Input / action**	**Desired output**	**Actual output (use red color if desired output != actual output)**
3	<Run Program>	Here are our students: First name: Steve Last name: Hommy Student ID: 1 First name: Jhon Last name: Snow Student ID: 2 Let's give our student a pet Species of the pet: Dog Name of the pet: Brak Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve	Here are our students: First name: Steve Last name: Hommy Student ID: 1 First name: Jhon Last name: Snow Student ID: 2 Let's give our student a pet Species of the pet: Dog Name of the pet: Brak Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve

		Species of the pet: Rabbit Name of the pet: Snug Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Owner of the pet: Jhon Would you like to remove one of these pets? yes Which student pet would you like to remove? Steve There are 2 pets in a list. How many would you like to remove? 1 Species of the pet: Dog Name of the pet: Brak Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve Frist index is 0. Which pet would you like to remove from the list? 0 Here are the pets that remain Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve Species of the pet: Rabbit Name of the pet: Snug Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Owner of the pet: Jhon	Species of the pet: Rabbit Name of the pet: Snug Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Owner of the pet: Jhon Would you like to remove one of these pets? yes Which student pet would you like to remove? Steve There are 2 pets in a list. How many would you like to remove? 1 Species of the pet: Dog Name of the pet: Brak Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve Frist index is 0. Which pet would you like to remove from the list? 0 Here are the pets that remain Species of the pet: Cat Name of the pet: Snuf Owner of the pet: Steve Species of the pet: Rabbit Name of the pet: Snug Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Owner of the pet: Jhon
5	<Run Program>	Here are our students: First name: Steve Last name: Hommy Student ID: 1 First name: Jhon Last name: Snow Student ID: 2 Let's give our student a pet and a car Steve owns these: Species of the pet: Dog Name of the pet: Brak	Here are our students: First name: Steve Last name: Hommy Student ID: 1 First name: Jhon Last name: Snow Student ID: 2 Let's give our student a pet and a car Steve owns these: Species of the pet: Dog Name of the pet: Brak

		Size of the pet: 150 Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Size of the pet: 100 Owner of the pet: Steve Brand: Toyota Model: Avensis Boot size: 200 Owner: Steve Jhon owns these: Species of the pet: Rabbit Name of the pet: Snug Size of the pet: 50 Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Size of the pet: 10 Owner of the pet: Jhon Brand: VW Model: Golf Boot size: 150 Owner: Jhon Would you like to remove one of these pets? no Here are the pets that remain Species of the pet: Dog Name of the pet: Brak Size of the pet: 150 Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Size of the pet: 100 Owner of the pet: Steve Species of the pet: Rabbit Name of the pet: Snug Size of the pet: 50 Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Size of the pet: 10 Owner of the pet: Jhon Let's check if pets will fit into the car Steve pets won't fit we need a trailer	Size of the pet: 150 Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Size of the pet: 100 Owner of the pet: Steve Brand: Toyota Model: Avensis Boot size: 200 Owner: Steve Jhon owns these: Species of the pet: Rabbit Name of the pet: Snug Size of the pet: 50 Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Size of the pet: 10 Owner of the pet: Jhon Brand: VW Model: Golf Boot size: 150 Owner: Jhon Would you like to remove one of these pets? no Here are the pets that remain Species of the pet: Dog Name of the pet: Brak Size of the pet: 150 Owner of the pet: Steve Species of the pet: Cat Name of the pet: Snuf Size of the pet: 100 Owner of the pet: Steve Species of the pet: Rabbit Name of the pet: Snug Size of the pet: 50 Owner of the pet: Jhon Species of the pet: Fish Name of the pet: Blub Size of the pet: 10 Owner of the pet: Jhon Let's check if pets will fit into the car Steve pets won't fit we need a trailer

		Jhon pets will fit	Jhon pets will fit
6	<Run Program> <User input>	Uruguay Give capital: Wrong answer the correct answer is Montevideo Philippines Give capital: Wrong answer the correct answer is Manila Samoa Give capital: Wrong answer the correct answer is Apia Mali Give capital: Wrong answer the correct answer is Bamako Liechtenstein Give capital: Wrong answer the correct answer is Vaduz Singapore Give capital: Wrong answer the correct answer is Singapore Kyrgyzstan Give capital: Give capital: Wrong answer the correct answer is Victoria Azerbaijan Give capital: Wrong answer the correct answer is Baku Brazil Give capital: Wrong answer the correct answer is Brasilia Score: 0/10 PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> & C:/Users/steve/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/steve/OneDrive/Desktop/Object-Oriented-Programming/Exercise7/exercise7/main.py Our first car is: Brand: Honda Brand: Tesla Tyre: Nokia Body style: Sedan	Uruguay Give capital: Wrong answer the correct answer is Montevideo Philippines Give capital: Wrong answer the correct answer is Manila Samoa Give capital: Wrong answer the correct answer is Apia Mali Give capital: Wrong answer the correct answer is Bamako Liechtenstein Give capital: Wrong answer the correct answer is Vaduz Singapore Give capital: Wrong answer the correct answer is Singapore Kyrgyzstan Give capital: Give capital: Wrong answer the correct answer is Victoria Azerbaijan Give capital: Wrong answer the correct answer is Baku Brazil Give capital: Wrong answer the correct answer is Brasilia Score: 0/10 PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> & C:/Users/steve/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/steve/OneDrive/Desktop/Object-Oriented-Programming/Exercise7/exercise7/main.py Our first car is: Brand: Honda Brand: Tesla Tyre: Nokia Body style: Sedan

		0 to 100 in: 4.5 seconds Electric power: 250W Battery size: 1000 000A Honda will reach 0 to 100 in 8.5 seconds Tesla will reach 0 to 100 in 4.5 seconds PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> > & C:/Users/steve/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/steve/OneDrive/Desktop/Object-Oriented-Programming/Exercise7/exercise6/main.py Ghana Give capital: Accra Correct! Comoros Give capital: Don't know Wrong answer the correct answer is: Moroni Kyrgyzstan Give capital: Bishek Wrong answer the correct answer is: Bishkek Finland Give capital: Helsinki Correct! Norway Give capital: Oslo Correct! Nicaragua Give capital: Don't know Wrong answer the correct answer is: Managua Comoros Give capital: Don't know Wrong answer the correct answer is: Moroni Belarus Give capital: Minsk Correct! Liechtenstein Give capital: Vaduz Correct! Luxembourg Give capital: Luxembourg Correct!	0 to 100 in: 4.5 seconds Electric power: 250W Battery size: 1000 000A Honda will reach 0 to 100 in 8.5 seconds Tesla will reach 0 to 100 in 4.5 seconds PS C:\Users\steve\OneDrive\Desktop\Object-Oriented-Programming> > & C:/Users/steve/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/steve/OneDrive/Desktop/Object-Oriented-Programming/Exercise7/exercise6/main.py Ghana Give capital: Accra Correct! Comoros Give capital: Don't know Wrong answer the correct answer is: Moroni Kyrgyzstan Give capital: Bishek Wrong answer the correct answer is: Bishkek Finland Give capital: Helsinki Correct! Norway Give capital: Oslo Correct! Nicaragua Give capital: Don't know Wrong answer the correct answer is: Managua Comoros Give capital: Don't know Wrong answer the correct answer is: Moroni Belarus Give capital: Minsk Correct! Liechtenstein Give capital: Vaduz Correct! Luxembourg Give capital: Luxembourg Correct!

		Score: 6/10	Score: 6/10
7	<Run Program>	Our first car is: Brand: Honda Tyre: Continental Body style: Hatchback 0 to 100 in: 8.5 seconds Engine size: 1.6l Tank size: 100l Our second car is: Brand: Tesla Tyre: Nokia Body style: Sedan 0 to 100 in: 4.5 seconds Electric power: 250W Battery size: 1000 000A Honda will reach 0 to 100 in 8.5 seconds Tesla will reach 0 to 100 in 4.5 seconds	Our first car is: Brand: Honda Tyre: Continental Body style: Hatchback 0 to 100 in: 8.5 seconds Engine size: 1.6l Tank size: 100l Our second car is: Brand: Tesla Tyre: Nokia Body style: Sedan 0 to 100 in: 4.5 seconds Electric power: 250W Battery size: 1000 000A Honda will reach 0 to 100 in 8.5 seconds Tesla will reach 0 to 100 in 4.5 seconds

1. Answer the following questions.

a. What does polymorphism (in object-oriented programming) mean? Also give a short (coding) example, e.g. google for examples).

Polymorphism is the method in an object-oriented programming language that performs different things as per the object's class, which calls it. With Polymorphism, a message is sent to multiple class objects, and every object responds appropriately according to the properties of the class.

```
class Bird:
  def intro(self):
    print("There are many types of birds.")

  def flight(self):
    print("Most of the birds can fly but some cannot.")

class sparrow(Bird):
  def flight(self):
    print("Sparrows can fly.")

class ostrich(Bird):
```

```
  def flight(self):
    print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
obj_spr.flight()

obj_ost.intro()
obj_ost.flight()
```

Output:

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

b. What is a class variable and how are they used?

A class variable defines a specific attribute or property for a class and may be referred to as a member variable or static member variable. They are associated with the class, rather than with any object. Every instance of the class shares a class variable, which is in one fixed location in memory. Any object can change the value of a class variable, but class variables can also be manipulated without creating an instance of the class

c. What is an instance variable and how is it different from the class variable?

Instance variables are owned by instances of the class. This means that for each object or instance of a class, the instance variables are different. Unlike class variables, instance variables are defined within methods.

d. What is a UML sequence diagram used for?

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

e. What is a lifeline in UML sequence diagrams?

A lifeline represents an individual participant in a sequence diagram. A lifeline will usually have a rectangle containing its object name. If its name is "self", that indicates that the lifeline represents the classifier which owns the sequence diagram.

2. More theory tasks

a. Multiple choice

i. In an inheritance relationship, the _____ is the general class.

1. Child class

2. Subclass

3. Superclass

4. Specialized class

ii. In an inheritance relationship, the _____ is the specialized class:

1. Superclass

2. Master class

3. Parent class

4. Subclass

iii. Let's say we have two classes in our program: BankAccount and SavingsAccount. Which one of them would most likely be the subclass?

1. BankAccount

2. SavingsAccount

3. Neither of them

4. Both of them.

iv. Which one of the options you will use if you want to check whether an object is an instance of a class?

1. The instance operator

2. The is_object_of function

3. The isinstance function

4. There is not a way to check that at all.

v. Which one of the UML diagrams is a behavioral diagram?

1. Class diagram

2. Sequence diagram

3. Object diagram

4. Deployment diagram

vi. Which one of the UML diagrams is a structural diagram?

1. Use case diagram

2. State machine diagram

3. Activity diagram

4. Composite structure diagram

vii. In UML class diagrams, what does the notation * mean.

1. Multiplication operation

2. Power of operation

3. Multiplicity 0..n

4. Multiplicity 0..1

b. True or false?

i. It is not possible to call a superclass's __init__ method from a subclass's __init__ method.

False

ii. A subclass never inherits any methods or attributes from the superclass.

False

iii. A superclass can inherit methods from subclass, if they have been denoted with pass_to_super function.

False

iv. In a subclass it is possible to have methods and attributes in addition to those that the subclass inherits from superclass.

True

v. In Python, multiple inheritance does not exist.

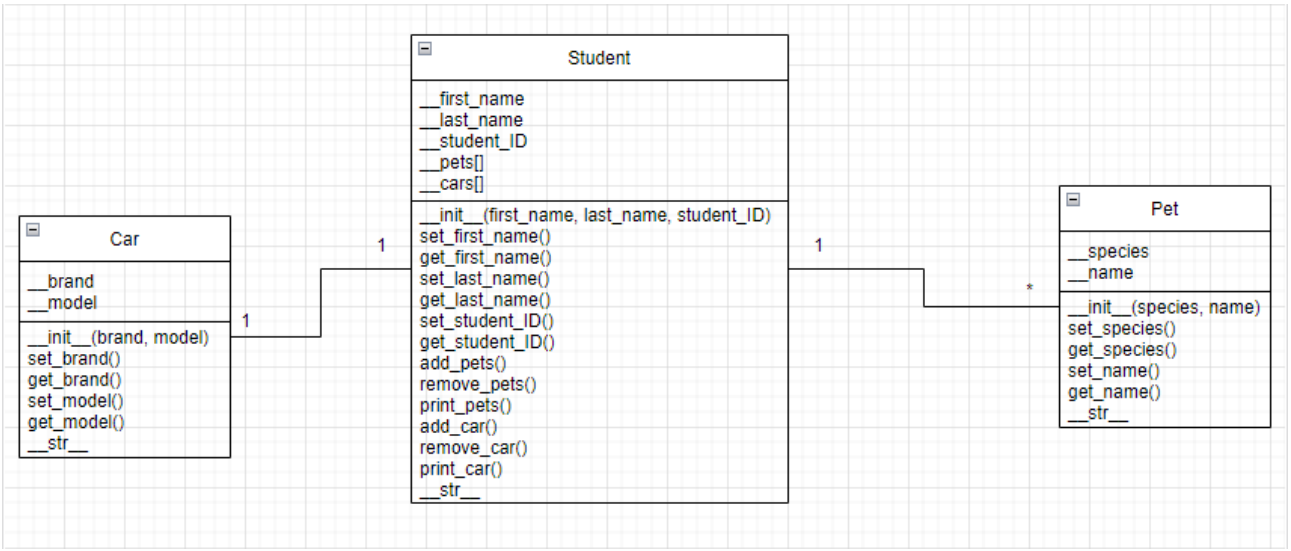
False

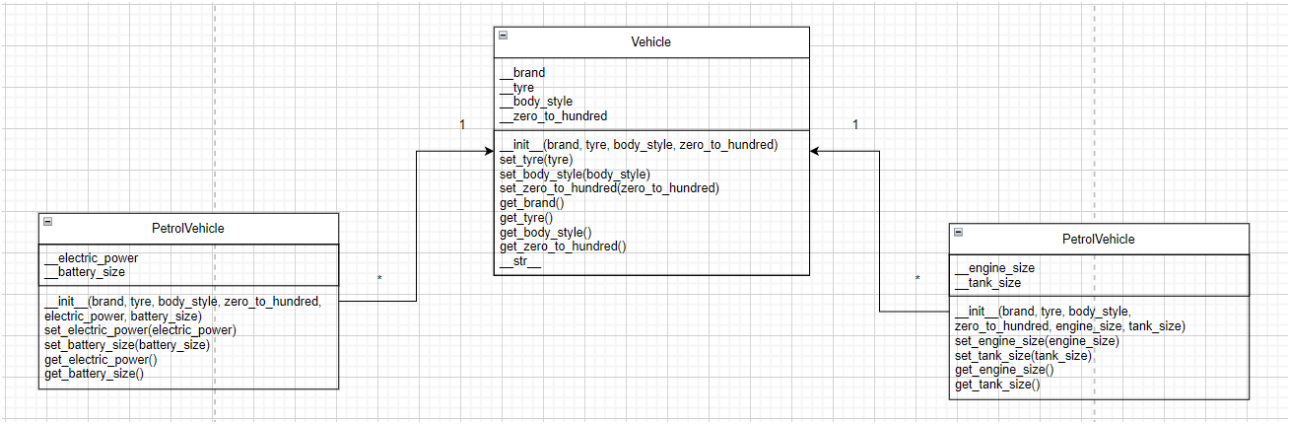
vi. Aggregation and composition shall never be used in UML class diagrams.

False

vii. Aggregation and composition mean exactly the same thing in UML class diagrams.

False





Code

```
# File name: pet.py
# Author: Steve Hommy
# Description: Create a Pet Class


class Pet:
    def __init__(self, species, name):
        self.__species = species
        self.__name = name
        self.__owner = None

    def set_species(self):
        self.__species = input("Give species for pet: ")
```

```
    def get_species(self):
        return self.__species

    def set_name(self):
        self.__name = input("Give name for pet: ")

    def get_name(self):
        return self.__name

    def set_owner(self, owner):
        self.__owner = owner

    def get_owner(self):
        return self.__owner

    def __str__(self):
        return f"""
        Species of the pet: {self.__species}
        Name of the pet: {self.__name}
        Owner of the pet: {self.__owner}"""
```

```
# File name: student.py
# Author: Steve Hommy
# Description: Create a Student Class


class Student:
    def __init__(self, first_name, last_name, student_ID):
        self.__first_name = first_name
        self.__last_name = last_name
        self.__student_ID = student_ID
        self.__pets = []

    def set_first_name(self):
        self.__first_name = input("Student first name: ")

    def get_first_name(self):
        return self.__first_name

    def set_last_name(self):
        self.__last_name = input("Student last name: ")

    def get_last_name(self):
        return self.__last_name

    def set_student_ID(self):
        self.__student_ID = input("Student ID: ")
```

```
    def get_student_ID(self):
        return self.__student_ID

    def add_pets(self, add_pet):
        try:
            if add_pet.get_owner() == None:
                self.__pets.append(add_pet)
                add_pet.set_owner(self.__first_name)
            else:
                print("Pet has an owner already")
        except ValueError:
            return print("Wrong value given")

    def remove_pets(self):
        for i in range(int(input("\nThere are " + str(len(self.__pets)) + " pets in
a list.\nHow many would you like to remove? "))):
            for pets in self.__pets:
                print(pets)
            self.__pets.pop(int(input("\nFrist index is 0. Which pet would you like
to remove from the list? ")))

    def print_pets(self):
        for pets in self.__pets:
            print(pets)

    def __str__(self):
        return f"""
        First name: {self.__first_name}
        Last name: {self.__last_name}
        Student ID: {self.__student_ID}
        """
```

```
# File: main.py
# Author: Steve Hommy
# Description: Main function

from student import Student
from pet import Pet


def main():
    student1 = Student("Steve", "Hommy", 1)
    student2 = Student("Jhon", "Snow", 2)
    pet1 = Pet("Dog", "Brak")
    pet2 = Pet("Cat", "Snuf")
    pet3 = Pet("Rabbit", "Snug")
```

```
    pet4 = Pet("Fish", "Blub")

    print("Here are our students:\n", student1, student2)

    print("\nLet's give our student a pet")
    student1.add_pets(pet1)
    student1.add_pets(pet2)
    student2.add_pets(pet3)
    student2.add_pets(pet4)

    student1.print_pets()
    student2.print_pets()
    question = input("\nWould you like to remove one of these pets? ")
    if question == "yes":
        question2 = input("\nWhich student pet would you like to remove? ")
        if question2 == "Steve":
            student1.remove_pets()
        elif question2 == "Jhon":
            student2.remove_pets()
        else:
            print("\nHere are the pets that remain")
            student1.print_pets()
            student2.print_pets()

    print("\nHere are the pets that remain")
    student1.print_pets()
    student2.print_pets()


main()
```

```
# File name: car.py
# Author: Steve Hommy
# Description: Create a Car Class


class Car:
    def __init__(self, brand, model, boot_size):
        self.__brand = brand
        self.__model = model
        self.__boot_size = boot_size
        self.__owner = None

    def set_brand(self):
        self.__brand = input("What brand is the car? ")

    def get_brand(self):
```

```
        return self.__brand

    def set_model(self):
        self.__model = input("What model is the car? ")

    def get_model(self):
        return self.__model

    def set_boot_size(self):
        self.__boot_size = int(input("What is the boot size? "))

    def get_boot_size(self):
        return self.__boot_size

    def set_owner(self, owner):
        self.__owner = owner

    def get_owner(self):
        return self.__owner

    def __str__(self):
        return f"""
        Brand: {self.__brand}
        Model: {self.__model}
        Boot size: {self.__boot_size}
        Owner: {self.__owner}
        """
```

```
# File name: pet.py
# Author: Steve Hommy
# Description: Create a Pet Class


class Pet:
    def __init__(self, species, name, pet_size):
        self.__species = species
        self.__name = name
        self.__pet_size = pet_size
        self.__owner = None

    def set_species(self):
        self.__species = input("Give species for pet: ")

    def get_species(self):
        return self.__species

    def set_name(self):
```

```
        self.__name = input("Give name for pet: ")

    def get_name(self):
        return self.__name

    def set_pet_size(self):
        self.__pet_size = int(input("Size of the pet"))

    def get_pet_size(self):
        return self.__pet_size

    def set_owner(self, owner):
        self.__owner = owner

    def get_owner(self):
        return self.__owner

    def __str__(self):
        return f"""
        Species of the pet: {self.__species}
        Name of the pet: {self.__name}
        Size of the pet: {self.__pet_size}
        Owner of the pet: {self.__owner}"""
```

```
# File name: student.py
# Author: Steve Hommy
# Description: Create a Student Class


class Student:
    def __init__(self, first_name, last_name, student_ID):
        self.__first_name = first_name
        self.__last_name = last_name
        self.__student_ID = student_ID
        self.__pets = []
        self.__cars = []

    def set_first_name(self):
        self.__first_name = input("Student first name: ")

    def get_first_name(self):
        return self.__first_name

    def set_last_name(self):
        self.__last_name = input("Student last name: ")

    def get_last_name(self):
```

```
        return self.__last_name

    def set_student_ID(self):
        self.__student_ID = input("Student ID: ")

    def get_student_ID(self):
        return self.__student_ID

    def add_pets(self, add_pet):
        try:
            if add_pet.get_owner() == None:
                self.__pets.append(add_pet)
                add_pet.set_owner(self.__first_name)
            else:
                print("Pet has an owner already")
        except ValueError:
            return print("Wrong value given")

    def remove_pets(self):
        for i in range(int(input("\nThere are " + str(len(self.__pets)) + " pets in
a list.\nHow many would you like to remove? "))):
            for pets in self.__pets:
                print(pets)
            self.__pets.pop(int(input("\nFrist index is 0. Which pet would you like
to remove from the list? ")))

    def print_pets(self):
        for pets in self.__pets:
            print(pets)

    def add_cars(self, add_car):
        if len(self.__cars) < 1:
            self.__cars.append(add_car)
            add_car.set_owner(self.__first_name)
        else:
            print("You already own 1 car")

    def remove_car(self):
        self.__cars.clear()

    def print_cars(self):
        for cars in self.__cars:
            print(cars)

    def __str__(self):
        return f"""
        First name: {self.__first_name}
        Last name: {self.__last_name}
        Student ID: {self.__student_ID}
```

```
        """
```

```
# File: main.py
# Author: Steve Hommy
# Description: Main function

from student import Student
from pet import Pet
from car import Car


def main():
    student1 = Student("Steve", "Hommy", 1)
    student2 = Student("Jhon", "Snow", 2)
    pet1 = Pet("Dog", "Brak", 150)
    pet2 = Pet("Cat", "Snuf", 100)
    pet3 = Pet("Rabbit", "Snug", 50)
    pet4 = Pet("Fish", "Blub", 10)
    car1 = Car("Toyota", "Avensis", 200)
    car2 = Car("VW", "Golf", 150)

    print("Here are our students:\n", student1, student2)

    print("\nLet's give our student a pet and a car")

    student1.add_pets(pet1)
    student1.add_pets(pet2)
    student1.add_cars(car1)

    student2.add_pets(pet3)
    student2.add_pets(pet4)
    student2.add_cars(car2)

    print(student1.get_first_name() + " owns these:")
    student1.print_pets()
    student1.print_cars()

    print(student2.get_first_name() + " owns these:")
    student2.print_pets()
    student2.print_cars()

    question = input("\nWould you like to remove one of these pets? ")
    if question == "yes":
        question2 = input("\nWhich student pet would you like to remove? ")
        if question2 == "Steve":
            student1.remove_pets()
        elif question2 == "Jhon":
```

```
            student2.remove_pets()
        else:
            print("\nHere are the pets that remain")
            student1.print_pets()
            student2.print_pets()

    print("\nHere are the pets that remain")
    student1.print_pets()
    student2.print_pets()

    print("\nLet's check if pets will fit into the car")
    if car1.get_boot_size() >= pet1.get_pet_size() + pet2.get_pet_size():
        print(student1.get_first_name() + " pets will fit")
    else:
        print(student1.get_first_name() + " pets won't fit we need a trailer")

    if car2.get_boot_size() >= pet3.get_pet_size() + pet4.get_pet_size():
        print(student2.get_first_name() + " pets will fit")
    else:
        print(student2.get_first_name() + " pets won't fit we need a trailer")


main()
```

```
# File: main.py
# Author: Steve Hommy
# Description: Main function

import random

filename = "Exercise7/exercise6/capitals.txt"
dictionary = {}

try:
    with open(filename) as file:
        for line in file:
            (key, value) = line.split()
            dictionary[key] = value
except FileNotFoundError:
    msg = "Sorry, the file " + filename + " does not exist.\n"
    print(msg)

while True:
    points = 0
    for i in range(10):
        country, capital = random.choice(list(dictionary.items()))
        print(country)
```

```
        answer = input("Give capital: ")
        if answer == dictionary[country]:
            print("Correct!\n")
            points += 1
        else:
            print("Wrong answer the correct answer is:", dictionary[country])
            print()

    print("Score:\n" + str(points) + "/10")
    break
```

```
# File name: electricVehicle.py
# Author: Steve Hommy
# Description: Inherit Vehicle Class and creating ElectricVehicle Class


from vehicle import Vehicle


class ElectricVehicle(Vehicle):
    def __init__(self, brand, tyre, body_style, zero_to_hundred, electric_power,
battery_size):
        Vehicle.__init__(self, brand, tyre, body_style, zero_to_hundred)
        self.__electric_power = electric_power
        self.__battery_size = battery_size

    def __str__(self):
        return super().__str__() + f"""Electric power: {self.__electric_power}
        Battery size: {self.__battery_size}
        """

    def set_electric_power(self, electric_power):
        self.__electric_power = electric_power

    def set_battery_size(self, battery_size):
        self.__battery_size = battery_size

    def get_electric_power(self):
        return self.__electric_power

    def get_battery_size(self):
        return self.__battery_size
```

```
# File name: petrolVehicle.py
# Author: Steve Hommy
# Description: Inherit Vehicle Class and creating PetrolVehicle Class
```

```python
from vehicle import Vehicle


class PetrolVehicle(Vehicle):
    def __init__(self, brand, tyre, body_style, zero_to_hundred, engine_size,
tank_size):
        Vehicle.__init__(self, brand, tyre, body_style, zero_to_hundred)
        self.__engine_size = engine_size
        self.__tank_size = tank_size

    def __str__(self):
        return super().__str__() + f"""Engine size: {self.__engine_size}
        Tank size: {self.__tank_size}
        """

    def set_engine_size(self, engine_size):
        self.__engine_size = engine_size

    def set_tank_size(self, tank_size):
        self.__tank_size = tank_size

    def get_engine_size(self):
        return self.__engine_size

    def get_tank_size(self):
        return self.__tank_size
```

```python
# File name: vehicleClass.py
# Author: Steve Hommy
# Description: Create a Vehicle Class


class Vehicle:
    def __init__(self, brand, tyre, body_style, zero_to_hundred):
        self.__brand = brand
        self.__tyre = tyre
        self.__body_style = body_style
        self.__zero_to_hundred = float(zero_to_hundred)

    def __str__(self):
        return f"""
        Brand: {self.__brand}
        Tyre: {self.__tyre}
        Body style: {self.__body_style}
        0 to 100 in: {self.__zero_to_hundred} seconds
        """
```

```
    def set_brand(self, brand):
        self.__brand = brand

    def set_tyre(self, tyre):
        self.__tyre = tyre

    def set_body_style(self, body_style):
        self.__body_style = body_style

    def set_zero_to_hundred(self, zero_to_hundred):
        self.__zero_to_hundred = zero_to_hundred

    def get_brand(self):
        return self.__brand

    def get_tyre(self):
        return self.__tyre

    def get_body_style(self):
        return self.__body_style

    def get_zero_to_hundred(self):
        return self.__zero_to_hundred
```

```
# File: main.py
# Author: Steve Hommy
# Description: Main function


from petrolVehicle import PetrolVehicle
from electricVehicle import ElectricVehicle


def main():
    honda = PetrolVehicle("Honda", "Continental", "Hatchback", 8.5, "1.6l", "100l")
    tesla = ElectricVehicle("Tesla", "Nokia", "Sedan", 4.5, "250W", "1000 000A")

    print("Our first car is:", honda)
    print("Our second car is:", tesla)

    how_fast_dict = {
        honda.get_brand(): honda.get_zero_to_hundred(),
        tesla.get_brand(): tesla.get_zero_to_hundred()
    }

    for key in how_fast_dict:
```

```
        print(key, "will reach 0 to 100 in", how_fast_dict[key], "seconds")


main()
```