class Pies:

    def \_\_init\_\_(self, imie, wiek, waga):   # \_\_init\_\_ to konstruktor klasy

        self.imie = imie

        self.wiek = wiek               # definiowanie atrybutow w ciele konstruktora

        self.waga = waga

    def szczekanie(self):  # zdefiniowalismy metode o takiej samej skladni jak funkcja

        if self.waga > 10:   # zawsze uzywamy self, gdy korzystamy z atrybutow obiektu

            print(self.imie, 'robi HAU')

        else:                             # poniewaz zawsze przekazujemy jako pierwszy atrybut instancję, na której

            print(self.imie, 'miauczy')   # wywolywana jest metoda, uzyskujemy dostep do atrybutow przez  parametr self

    def ludzkie\_lata(self):

        ludzkie\_lata = int(self.wiek\*7)

        print('kodi mialby', ludzkie\_lata, 'lat')

    def chodzenie(self):

        print(self.imie, 'i jego opiekun', 'idą na spacer')

    def \_\_str\_\_(self):         # \_\_str\_\_ tworzy lancuch do wyswietlenia przez funkcje print

        return 'Jestem psem o imieniu ' + self.imie

class PiesTowarzyszacy(Pies): # deklarujemy podklasę PiesTowarzyszacy, ktora bedzie dziedzyczyla po klasie(nadklasie) głowniej Pies

    def \_\_init\_\_(self, imie, wiek, waga, opiekun): # kontruktor klasy PiesTowarzyszacy, z dodatkowym argumentem 'opiekun'

        Pies.\_\_init\_\_(self, imie, wiek, waga) # wywolujemy konstruktor klasy Pies i przekazuje mu dziedziczone argumenty

        self.opiekun = opiekun # dodajemy nowy atrybut opiekun do self. (do tej podklasy) bo tylko ona ma ten atrybut

        self.pelni\_sluzbe = False # dodajemy do klasy PiesTowarzyszacy nowy atrybut z wartoscia false

    def szczekanie(self):           # przy wywolaniu funkcji szczekanie na psie typu PiesTowarzyszacy  wykonewane beda instrukcje

        if self.pelni\_sluzbe:       # zawarte tutaj a nie w klasie Pies (klasa PT nadpisuje metode szczekaniu klasy pies)

            print(self.imie, 'mowi : Nie moge szczekac bo jestem na sluzbie')

        else:                         # jesli pelni\_sluzbe ma wartos True, pies odpowiada, ze pelni sluzbe i nie moze szczeka, w przeciwnym

            Pies.szczekanie(self)        # razie pies szczeka

class Frisbee:         # tworzymy nowa klase psa, który aportuje frisbee ( niczego nie dziedziczy)

    def \_\_init\_\_(self, kolor): # ma okreslony tylko kolor a ponizej metode str

        self.kolor = kolor

    def \_\_str\_\_(self):    # \_\_str\_\_ okresla tekst zwracany przez print

        return 'Jestem Frisbee i mam kolor ' + self.kolor

class PiesAportujacy(Pies): # nowa klasa PiesAportujacy dziedziczy po klasie Pies

    def \_\_init\_\_(self, imie, wiek, waga):

        Pies.\_\_init\_\_(self, imie, wiek, waga)

        self.frisbee = None   # mamy kontruktor definiujacy atrybut frisbee. Frisbee jest innym obiektem (agregacja)

    def szczekanie(self):

        if self.frisbee != None:   # jezeli mam frisbee to nie moge szczekac(szczekanie isa Pies)

            print(self.imie, 'nie moge szczekac bo mam frisbee')

        else:

            Pies.szczekanie(self) # jezeli nie mam frisbee to szczekam jak zwykly (Pies)

    def lapanie(self, frisbee):

        self.frisbee = frisbee  # metoda lapanie przyjmuje frisbee i przypisuje je do atrybutu frisbee obiektu

        print(self.imie, 'zlapal frisbee w kolorze', frisbee.kolor + 'm')

    def zwracanie(self):

        if self.frisbee != None:   # metoda zwracanie nadaje atrybutowi frisbee wartosc None a nastepnie

            frisbee = self.frisbee   # zwraca frisbee

            self.frisbee = None

            print(self.imie, 'zwraca frisbee w kolorze', frisbee.kolor + 'm')

            return frisbee

        else:

            print('nie ma frisbee')

            return None

    def chodzenie(self): # jezeli pies aportujacy ma w gebie frisbee to nie idzie

        if self.frisbee != None:

            print(self.imie, ' Nie moge isc bo mam frisbee')

        else:                   # w przeciwnym wypadku wywolana zostaje metoda chodzenie nadklasy Pies i pies

            PiesTowarzyszacy.chodzenie(self) # robi to samo co obiekty klasy Pies

    def \_\_str\_\_(self):       # metoda str warunkowo zwraca lancuhc w zaleznosci od tego czy pies ma frisbee

        str = "Jestem psem o imieniu " + self.imie

        if self.frisbee != None:

            str = str + ' i mam frisbee'

        return str

def kod\_testowy():   # funkcja ogolna

    drab = PiesAportujacy('Drab', 5, 20)        # tworzymy psa z klasy PiesAportujacy i obiekt Frisbee

    niebieskie\_frisbee = Frisbee('niebieski')

    print(drab)                     # wyswietlamy psa i kazemy mu szczekac a nastepnie zlapac frisbee

    drab.szczekanie()

    drab.lapanie(niebieskie\_frisbee)

    drab.szczekanie()           # kazemy mu szczekac gdy ma w pysku frisbee

    print(drab)                 # wyswietlamy psa (ktory ma frisbee)  i kazemy mu je zwrocic

    frisbee = drab.zwracanie()

    print(frisbee)           # wyswietlamy zwrocone frisbee i ponownie wyswietlamy psa

    print(drab)

    drab.chodzenie()

kod\_testowy()