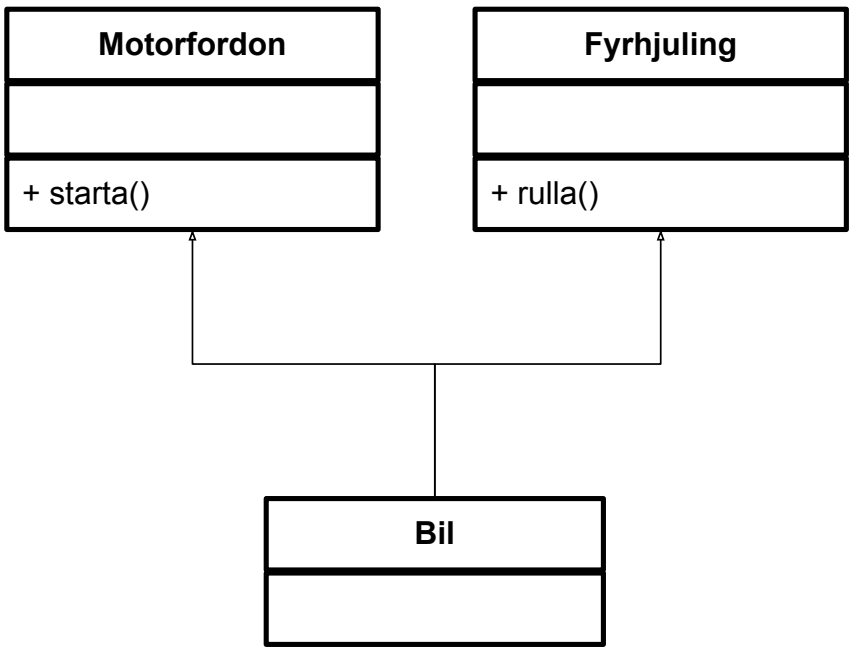
2.3 Flerfaldig ärvning

Holger Rosencrantz

Bil ärver från både Motorfordon och Fyrhjuling



Syntax

```
class Motorfordon:
    def starta(self):
        print("Vroom!")
class Fyrhjuling:
    def rulla(self):
        print("Rullar!")
class Bil(Motorfordon, Fyrhjuling):
    pass
b = Bil()
b.starta()  # Vroom!
b.rulla()   # Rullar!
print(isinstance(b, Bil) and isinstance(b, Motorfordon))  # True
```

Konstruktorer och attribut vid flerfaldig ärvning

```
class Motorfordon:
    def __init__(self, max_speed):
        self.max_speed = max_speed
class Fyrhjuling:
    def __init__(self, wheel_radius):
        self.wheel_radius = wheel_radius
class Bil(Motorfordon, Fyrhjuling):
    def __init__(self, max_speed, wheel_radius):
        Motorfordon.__init__(self, max_speed)
        Fyrhjuling.__init__(self, wheel_radius)
b = Bil(120, 15)
print(b.max_speed)      # 120
print(b.wheel_radius)   # 15
```

Den här veckan: Övningar metoder i klasshierarkier

Uppgifter:

1. Utöka den tidigare klasshierarkin så att `Djur` har publika metoder `at()` och `sov()`. `Fisk` ska ha en publik metod `simma()`. `Haj` ska ha en publik metod `at(djur)`. (Du får implementera metoderna som du vill, men se till att någon utskrift sker t.ex. “Fisken simmar” när du anropar metoden `simma()`.) Uppdatera även klassdiagrammet!
2. Lägg till klasserna `Cykel` och `Sportbil` i det inledande exemplet (med klasserna `Fordon` och `Bil`). I verkligheten är en cykel ett fordon och en sportbil en bil, så låt dem ärva på motsvarande sätt i din klasshierarki. Cykeln ska ha metoden `plinga` och sportbilen ska överskugga metoden `kör`. Du får själv bestämma vad som ska hända när man anropar metoderna, men enklast är att bara göra någon utskrift som i föregående övning. Rita ett klassdiagram.

Övrigt: Kattis, Progolymp