2.1 Klasshierarkier: introduktion till ärvning

Holger Rosencrantz

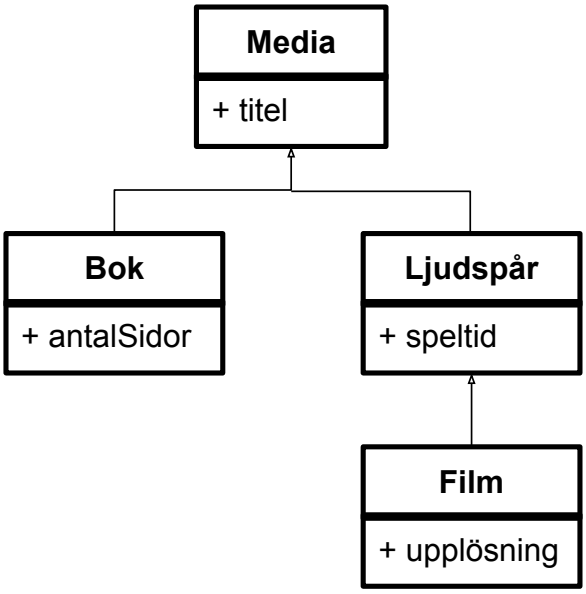
Klasshierarkier i naturen

- Världen är full av klasser: fordon, människor, bilar, mat, djur, myror, korv osv.
- Vissa av dessa är specialfall av andra: t.ex. både människor och myror är djur; alla bilar är fordon; korv är en sorts mat
- Man kan ordna dessa i hierarkier: t.ex. djur -> ryggradsdjur -> däggdjur -> människa eller djur -> insekter -> myror eller mat -> kötträtter -> korv
- Likheter mellan liknande klasser: t.ex. alla djur (både människor och myror) kan äta; alla fordon kan köra; all mat innehåller ett visst antal kalorier
- Likheter i både beteende och egenskaper

Klasshierarkier i objektorienterad programmering

- Vi har tidigare lärt oss att implementera klasser i Python och skapa objekt av dessa klasser (genom att anropa klassens konstruktor)
- Beteende motsvarar **metoder** och egenskaper motsvarar **attribut**
- När man programmerar vill man gärna **återanvända** kod
- Istället för att skapa en separat klass för bil och en separat klass för motorcykel från scratch, kanske vi tjänar på att först skapa en allmän klass för motorfordon
- Denna allmänna klass innehåller alla gemensamma egenskaper, t.ex. registreringsnummer och maxhastighet, som alla motorfordon har
- Vi kan sedan låta nya klasser, som t.ex. bil och motorcykel, **ärva** de gemensamma egenskaperna från den allmänna klassen
- Klasser som ärver kallas för **subklasser** (”underklass”)
- Klasser som de ärver från kallas för **superklasser** (eller ”basklass”)

Exempel: mediabibliotek



- Ett bibliotek består av **medier** (superklass)
- Ett media kan vara en **bok** eller ett **ljudspår** (subklasser)
- Ett ljudspår kan vara en **film** (subklass till ljudspår)
- Ljudspår är alltså både subklass och superklass
- Alla medier har attributet titel
- Alla böcker har attributen titel och antalSidor
- Alla ljudspår har attributen titel och speltid
- I båda fallen **ärvs** egenskapen titel från superklassen
- Vilka attribut har klassen film?

Implementera mediabiblioteket i Python

```
class Media:
    def __init__(self, titel):
        self.titel = titel
class Bok(Media):
    def __init__(self, titel, antalSidor):
        super().__init__(titel)
        self.antalSidor = antalSidor
class Ljudspår(Media):
    def __init__(self, titel, speltid):
        super().__init__(titel)
        self.speltid = speltid
class Film(Ljudspår):
    def __init__(self, titel, speltid, upplösning):
        super().__init__(titel, speltid)
        self.upplösning = upplösning
```

Implementera mediabiblioteket i Python

```
class Media:
    def __init__(self, titel):
        self.titel = titel
class Bok(Media):
    def __init__(self, titel, antalSidor):
        super().__init__(titel)
        self.antalSidor = antalSidor
class Ljudspår(Media):
    def __init__(self, titel, speltid):
        super().__init__(titel)
        self.speltid = speltid
class Film(Ljudspår):
    def __init__(self, titel, speltid, upplösning):
        super().__init__(titel, speltid)
        self.upplösning = upplösning
```

superklass med attributet `titel`

subklasser som ärver superklassens attribut `titel`

Implementera mediabiblioteket i Python

```
class Media:
    def __init__(self, titel):
        self.titel = titel
class Bok(Media):
    def __init__(self, titel, antalSidor):
        super().__init__(titel)
        self.antalSidor = antalSidor
class Ljudspår(Media):
    def __init__(self, titel, speltid):
        super().__init__(titel)
        self.speltid = speltid
class Film(Ljudspår):
    def __init__(self, titel, speltid, upplösning):
        super().__init__(titel, speltid)
        self.upplösning = upplösning
```

superklass med attributen `titel` och `speltid`

subklass som ärver superklassens attribut `titel` och `speltid`; har dessutom det egna attributet `upplösning`

Implementera mediabiblioteket i Python

```
class Media:
    def __init__(self, titel):
        self.titel = titel
class Bok(Media):
    def __init__(self, titel, antalSidor):
        super().__init__(titel)
        self.antalSidor = antalSidor
class Ljudspår(Media):
    def __init__(self, titel, speltid):
        super().__init__(titel)
        self.speltid = speltid
class Film(Ljudspår):
    def __init__(self, titel, speltid, upplösning):
        super().__init__(titel, speltid)
        self.upplosning = upplosning
```

`super()` anropar superklassens konstruktor; för att skapa en Bok eller ett Ljudspår måste man först skapa ett Media

Samma här: för att skapa en Film måste man först skapa ett Ljudspår

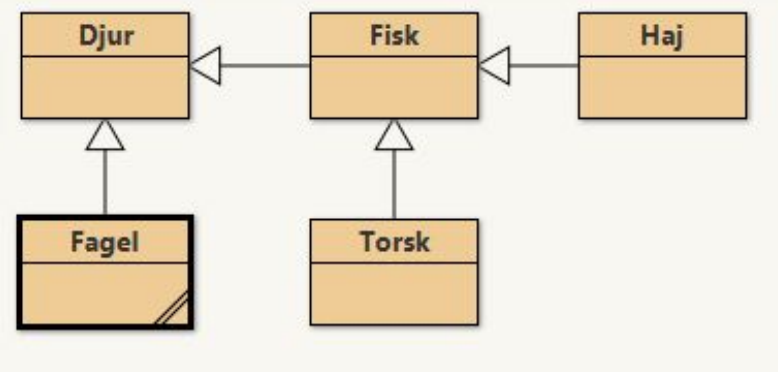
Testa implementationen av mediabibliteket

```
b = Bok("Bröderna Karamazov", 840)
print(b.titel)      # Bröderna Karamazov
print(b.antalSidor) # 840
if isinstance(b, Media) and isinstance(b, Bok): # True
    print("b är samtidigt av typen Media och Bok")
f = Film("De sju samurajerna", "3:27:02", "1080")
print(f.titel)      # De sju samurajerna
print(f.speltid)    # 3:27:02
print(f.upplösning) # 1080
if isinstance(f, Media) and isinstance(f, Ljudspår) and
    isinstance(f, Film):    # True
    print("f är samtidigt av typen Media, Ljudspår och Film")
```

Idag och nästa lektion: Övningar introduktion till ärvning

Studera fler exempel från w3schools enligt länkar i lektionsanteckningarna.

1. Implementera en klass `Djur` med attributet `namn`. Skapa två subklasser till `Djur`: `Fagel` och `Fisk`. `Fagel` ska ha ett attribut `vingspann`. `Fisk` ska ha ett attribut `maxdjup`. Skapa två subklasser till `Fisk`: `Haj` och `Torsk`. `Haj` ska ha ett attribut `antalTänder`. `Torsk` ska ha ett attribut `hastighet`. Se bild (pil betyder ärvning):



2. Skriv en funktion `fånga(haj, torsk)` som returnerar `True` ifall (a) torskens `hastighet` är mindre än `30` och (b) hajens `maxdjup` är minst lika stort som torskens, annars `False`.

Övrigt: Kattis-utmaningar inför progolymp, glöm inte att pusha till GitHub