OBJECT ORIENTED DESIGN

Workshop: Scientific Computing 01, March 2022

Ida Marie Lassen, Center for Humanities Computing Aarhus

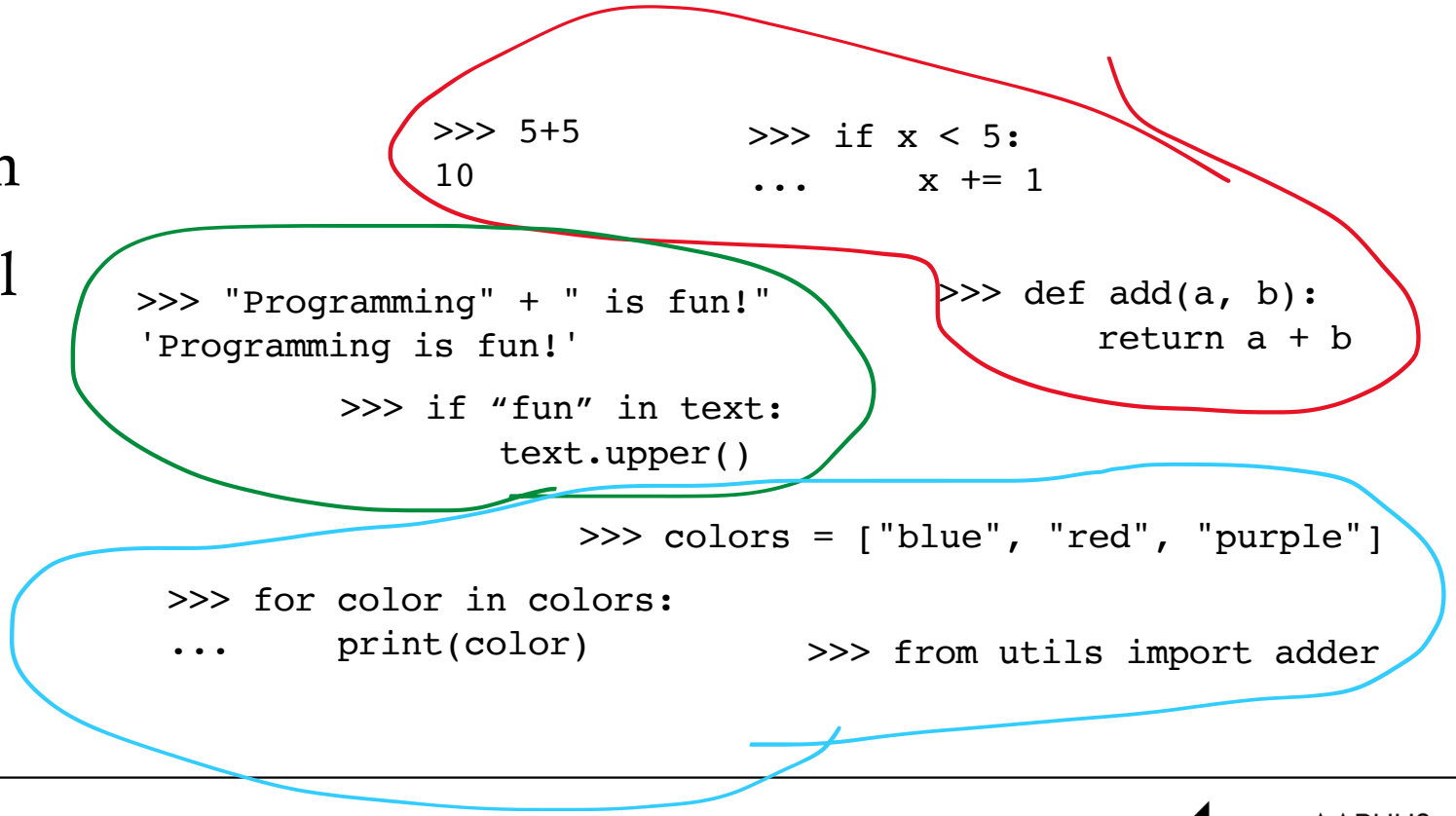
FIRST WE THINK, THEN WE CODE...

4 FUNDAMENTAL CONCEPTS

- ABSTRACTION
- ENCAPSULATION
- INHERITANCE
- POLYMORPHISM

WHAT HAVE WE LEARNED SO FAR?

- Basic python
- Flow control
- Functions
- Modules



GROUPING DATA – AND ASSOCIATED FUNCTIONALITY

ABSTRACTION

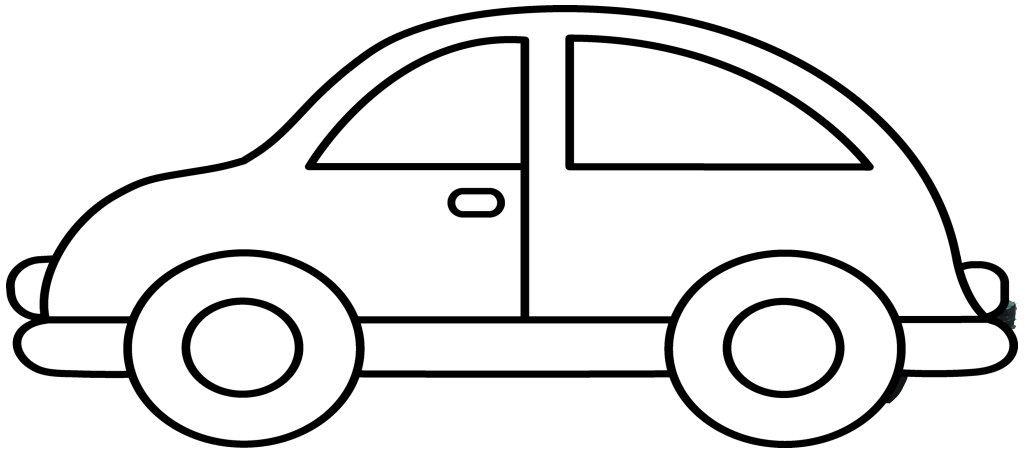
CLASSES

- An abstract blueprint.

ABSTRACTION

CLASSES

- An abstract blueprint.
- Collection of functionality.
- Definitions for the data format and available procedures for a given class of object

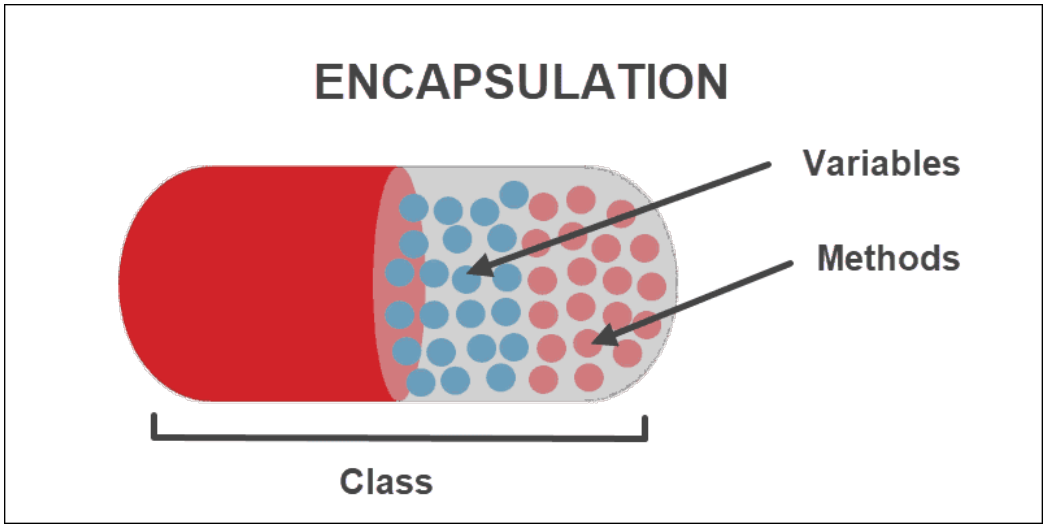


OBJECTS

- Individual instances of a class.

ENCAPSULATION

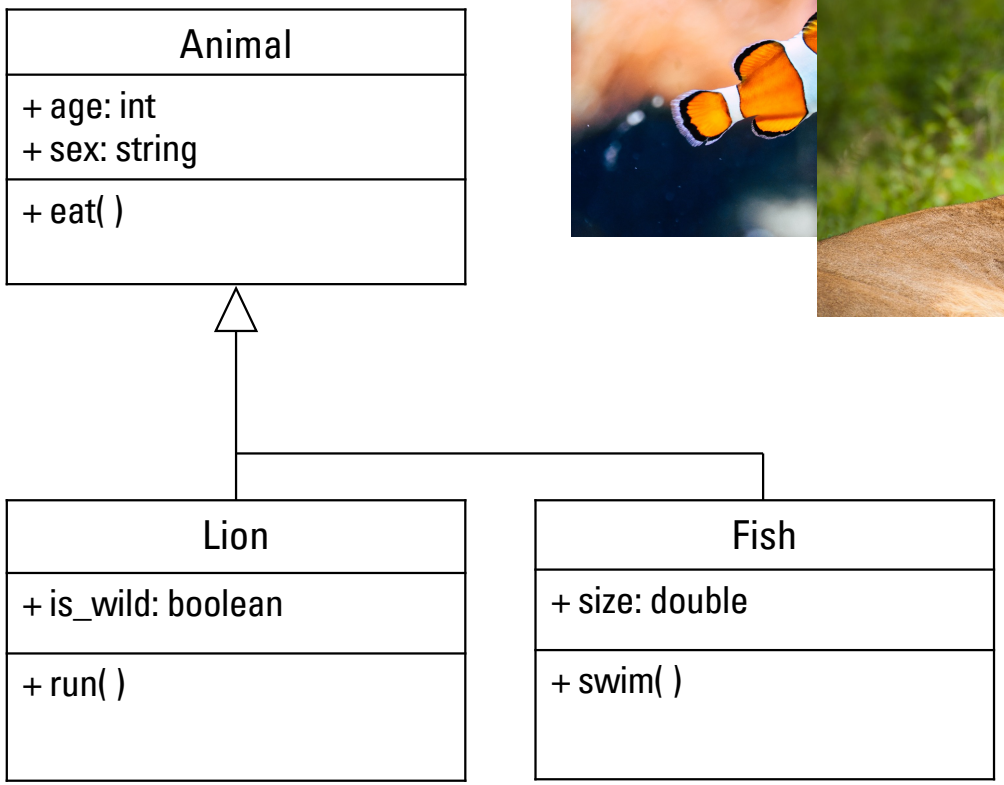
ONE CLASS – ONE ROLE – ONE RESPONSIBILITY



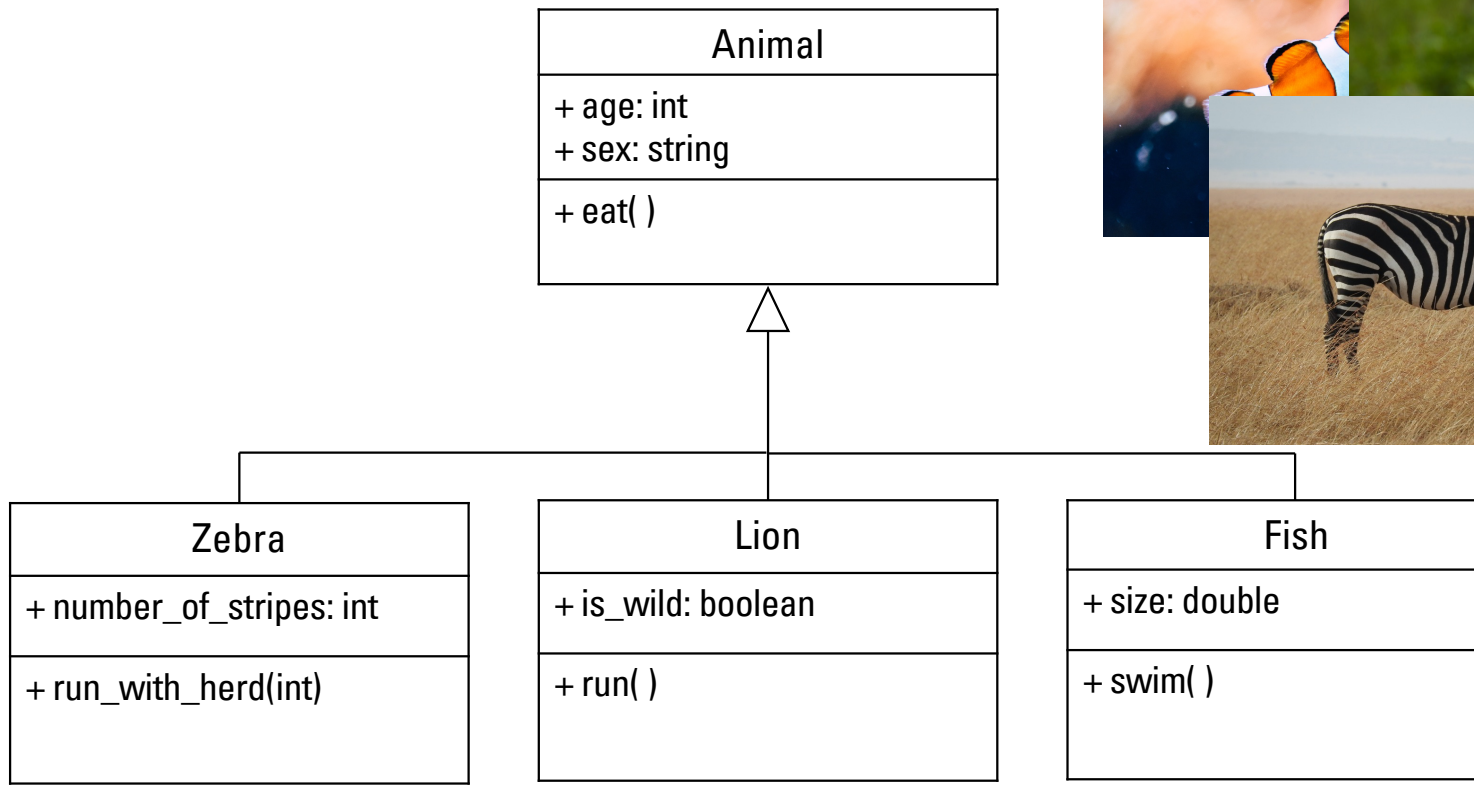
- With encapsulation we prevent external code from being concerned with the internal workings of an object.
- Also known as decoupling.

INHERITANCE – ZOO EXAMPLE

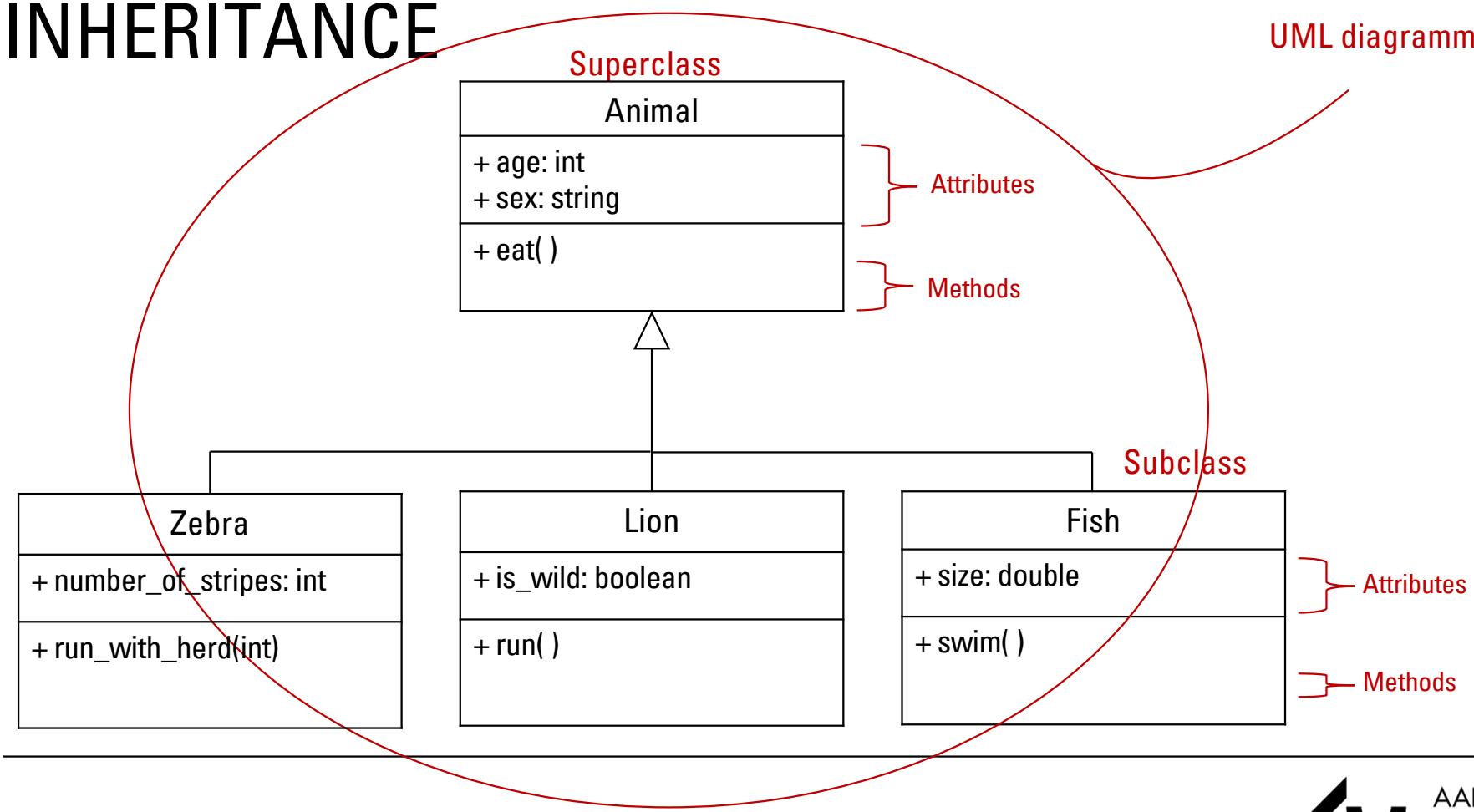
INHERITANCE – ZOO EXAMPLE

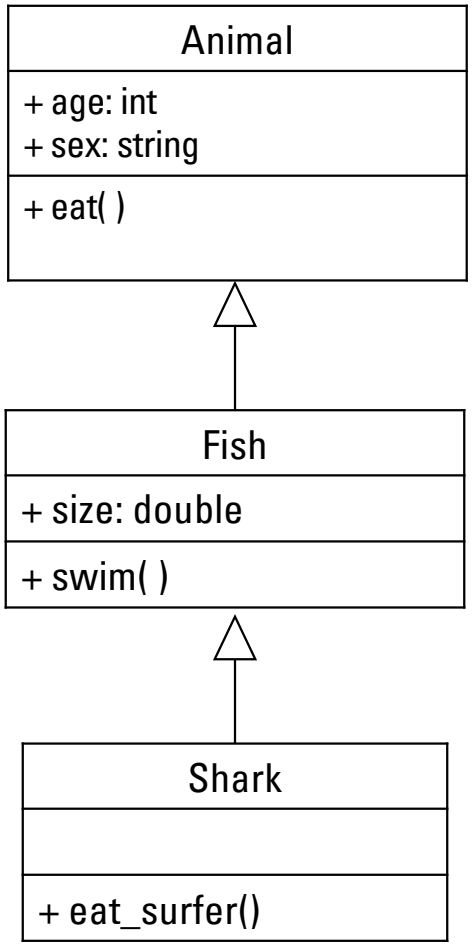


INHERITANCE – ZOO EXAMPLE



INHERITANCE

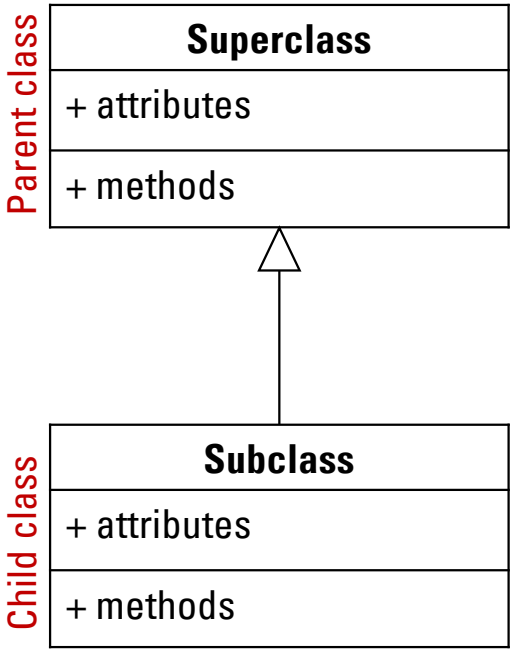




MULITPLE INHERITANCE

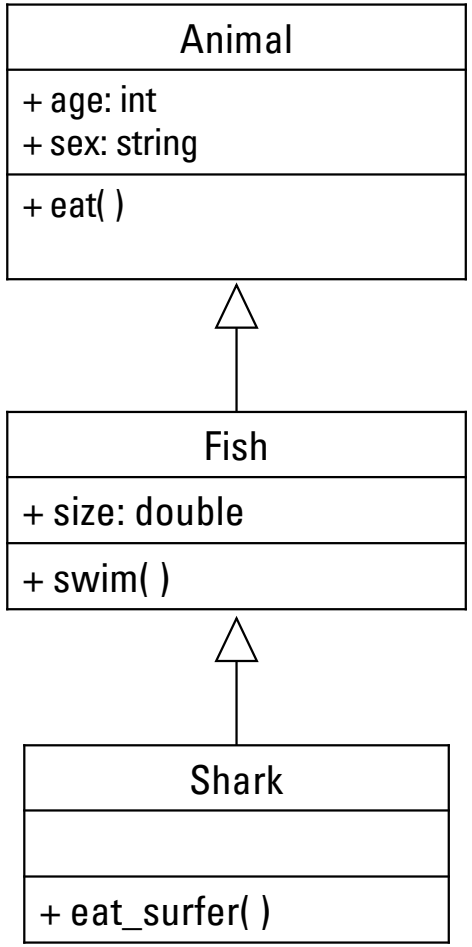
AARHUS UNIVERSITY

INHERITANCE – "IS A" RELATIONSHIPS!



Liskov's Substitution Principle:

"You can use an instance of an object of a subclass whenever an object of the superclass is exptected"



Liskov's Substitution Principle:

"You can use an instance of an object of a subclass whenever an object of the superclass is exptected"

POLYMORPHISM – THE ABILITY TO TAKE DIFFERENT FORMS

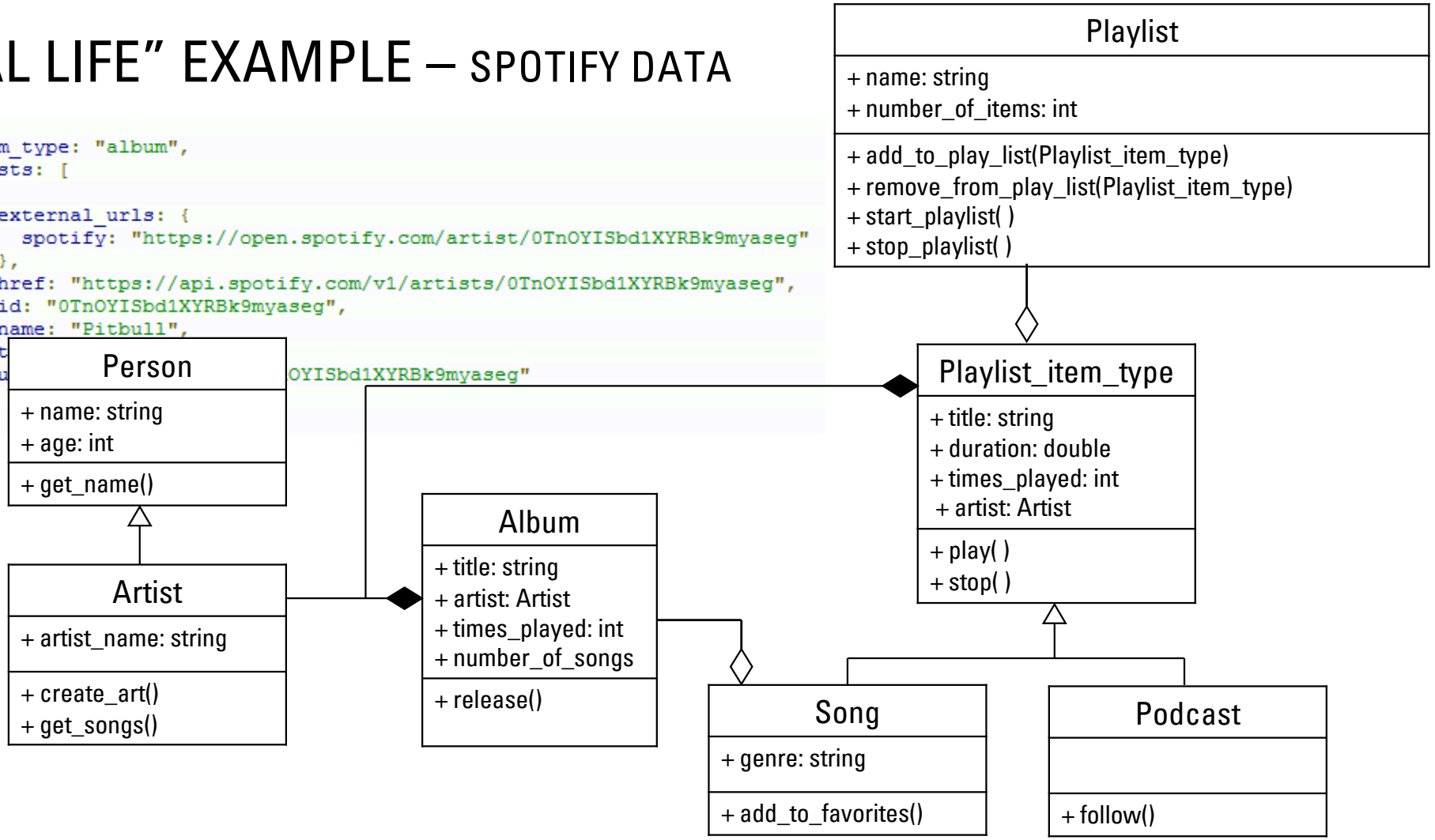
```python
class Animal:
    def eat(self):
        print("Animal is now eating")

class Lion(Animal):
    def eat(self, type_of_meat):
        print(f'Lion is eating {type_of_meat}')

def main():
    animal_obj = Animal()
    lion_obj = Lion()
    animal_obj.eat()
    lion_obj.eat("zebra meat")

if __name__ == '__main__':
    main()
```

In simple words, polymorphism allows us to perform the same action in different ways.

Overloading: Overriding methods in subclasses.

“REAL LIFE” EXAMPLE – SPOTIFY DATA


```
{
  album_type: "album",
  artists: [
    {
      external_urls: {
        spotify: "https://open.spotify.com/artist/0TnOYISbd1XYRBk9myaseg"
      },
      href: "https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg",
      id: "0TnOYISbd1XYRBk9myaseg",
      name: "Pitbull",
      t
      u                                    OYISbd1XYRBk9myaseg"
    }
  ],
```

RECAP

- ABSTRACTION:
  generalize functionalities and properties into a class.
- ENCAPSULATION:
  encapsulate responsibility within one class
  and decouple it from other classes.
- INHERITANCE:
  “is a” relationships where functionalities and
  properties are inherited.
- POLYMORPHISM:
  the ability of inherited methods to redefine behavior.

```
>>> some_text = “Programming is fun”
>>> some_list = [2,5,3,6,7,1]
>>> len(some_text)
18
>>> len(some_list)
6
```

PEN-AND-PAPER DESIGN EXERCISE

0. Decide on a “is a”-relationship you want to work with.
1. Design several classes – at least two subclasses and one superclass.
2. Identify the attributes and methods. Which ones belong in the superclass? And which in the subclasses?
3. Create a UML diagram to visualize your classes.

Tip: Keep it simple!

OBJECT ORIENTED PROGRAMMING
IN PYTHON

AARHUS
UNIVERSITY

REFERENCES

CONTENT

- Phillips 2015, chp 1, Object-oriented Design
- Phillips 2015, chp 2, Objects in Python

IMAGES

- Encapsulation: https://medium.com/future-vision/intro-to-oop-with-python-39ba63967e45
- Car: Photo by Dan Gold on Unsplash
- Spaceship: Photo by NASA on Unsplash
- Book: Photo by Christian Wiediger on Unsplash
- Person: Photo by Pablo Heimplatz on Unsplash
- Student: Photo by Element5 Digital on Unsplash
- Appointment: Photo by Element5 Digital on Unsplash
- Reservation: Photo by Rafael Maggion on Unsplash
- Lion: Photo by Sten Nijssen on Unsplash
- Fish: Photo by Rachel Hisko on Unsplash
- Zebra: Photo by Ron Dauphin on Unsplash
- Spotify json: https://engineering.atspotify.com/2015/03/understanding-spotify-web-api/

FURTHER READINGS/ONLINE RESOURCES

- Intro to Object-oriented Programming in Python: https://medium.com/future-vision/intro-to-oop-with-python-39ba63967e45
- What is Polymorphism in OOPs programming?: https://www.edureka.co/blog/polymorphism-in-python/
- UML class diagram arrow types: explanations and examples: https://www.gleek.io/blog/class-diagram-arrows.html
- Python Design Patterns:
- https://python-patterns.guide/