THE UNIVERSITY OF WESTERN AUSTRALIA

Lecture 24
Objects

Objectives of this Lecture

- To get familiar with Objects
- To understand the concept of objects and how they can be used to simplify programs
- Understand that in Python, everything is actually an object

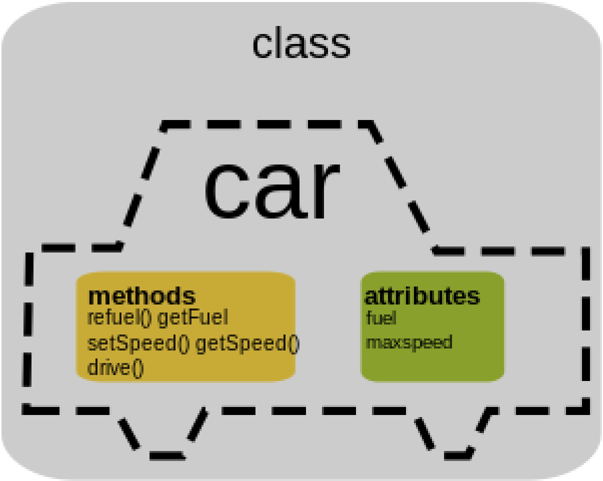
Overview

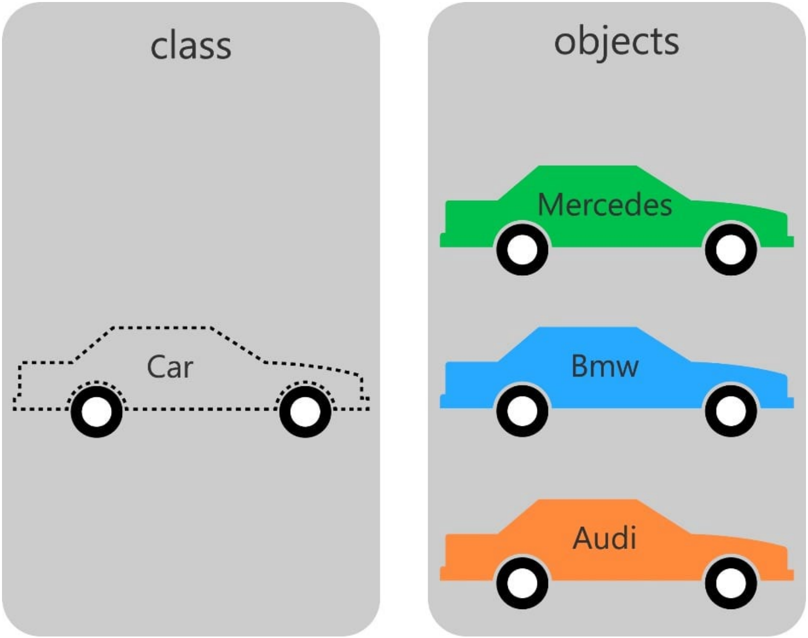
- So far, we saw that each data type can represent a certain set of values, and each had a set of associated operations.
- The traditional programming view is that data is passive – it is manipulated and combined using active operations.
- Modern computer programs are built using an object–oriented approach.

Objects and Object Oriented Programming

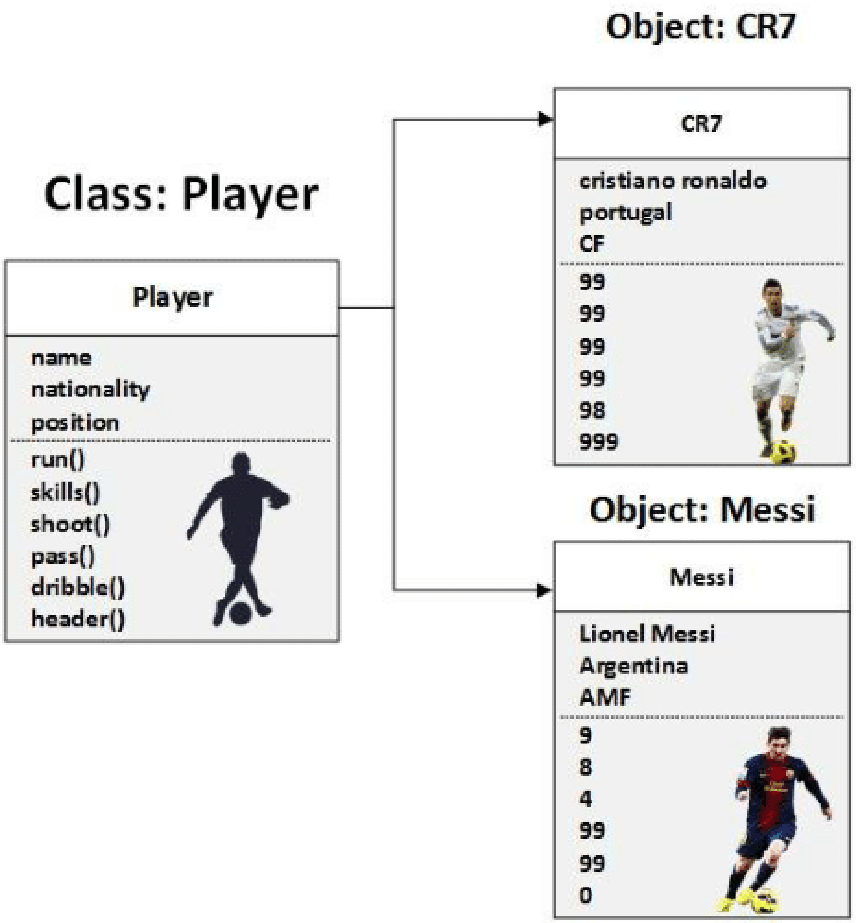
- Basic idea – view a complex system as the interaction of simpler objects.
- An object is a kind of active data type that combines data and operations.
  - *Objects know stuff (contain data) and they can do stuff (have operations).*
- Objects interact by sending each other messages *(requests to do stuff)*.

OOP concept


class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed


class
Car
objects
Mercedes
Bmw
Audi

Example



Class: Player

Player

name
nationality
position

run()
skills()
shoot()
pass()
dribble()
header()

Object: CR7

CR7

cristiano ronaldo
portugal
CF

99
99
99
99
98
999

Object: Messi

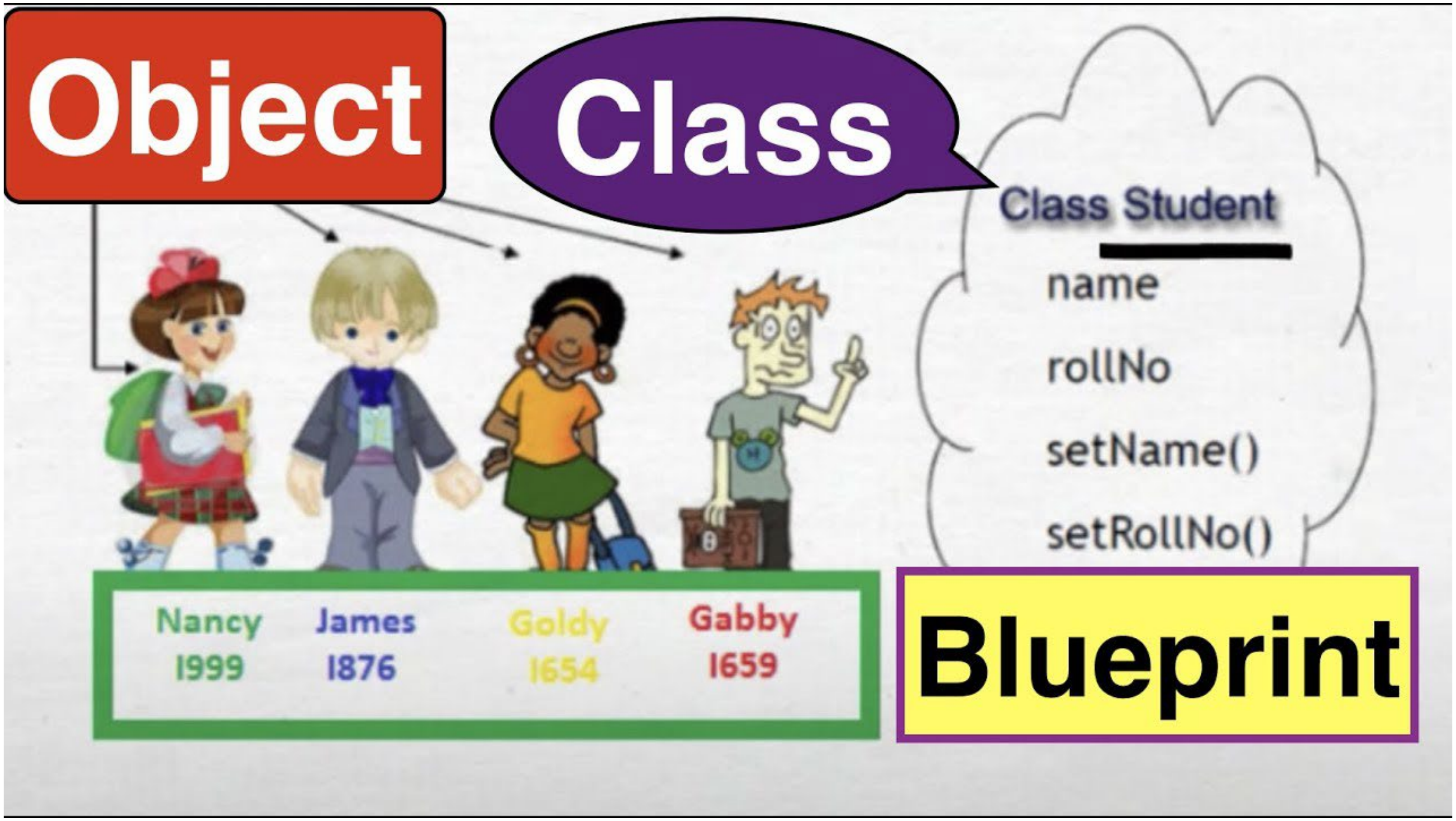
Messi

Lionel Messi
Argentina
AMF

9
8
4
99
99
0

How to learn OOP using football

Example (2)


Object
Class
Class Student
name
rollNo
setName()
setRollNo()
Nancy
1999
James
1876
Goldy
1654
Gabby
1659
Blueprint

Objects: Explained with an Example

- Suppose we want to develop a data processing system for a university.
- We must keep records of students who attend the university.
- Each student, each unit, etc., will be represented as different sorts of objects.

Univesity Student Object

- What information would be in a student object?
  - *Name*
  - *Home address*
  - *Residential address (if away from home)*
  - *Units*

What would the student object do?

- The student object should respond to requests.
- We may want to send out a campus-wide mailing, so we need a campus address for each student.
- We could send the getHomeAddress() to each student object. When the student object receives the message, it responds with the home address.

Course Object

- Each course might also be represented by an object:
- The Course-object:
  - *Instructor*
  - *Students enrolled*
  - *Pre-requisite courses*
  - *When and where the class meets*

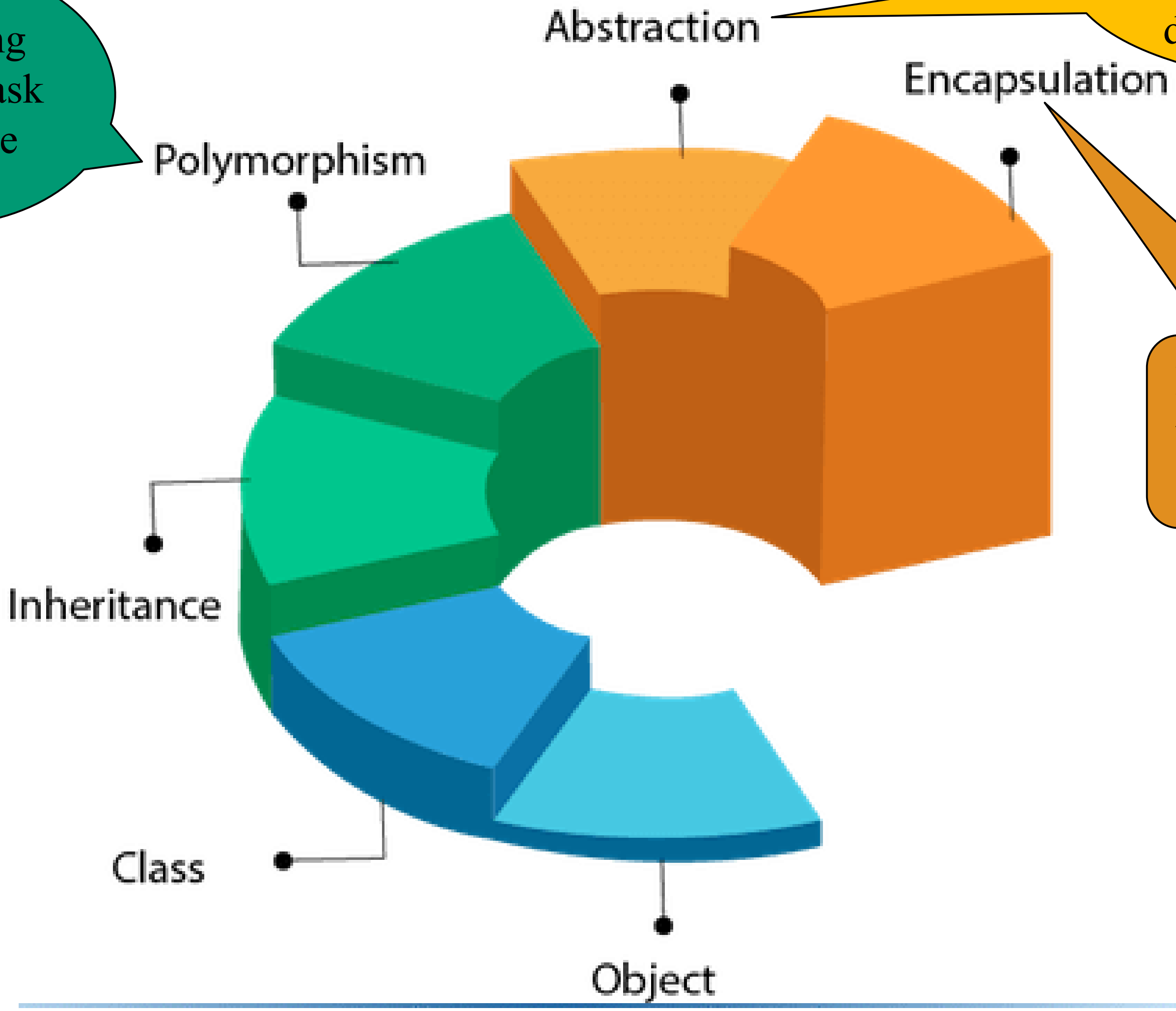
Objects within Objects

- An object can have one or more objects inside it
- For example, the course-object will have student-objects inside
- Similarly, the course-object may have an instructor-object.

Sample operations of the Course-object

- `addStudent()`
  - *`Student-object` added to course-object*
- `delStudent()`
- `changeRoom()`
- The point is that different operations are appropriate for objects (like different data-types)

Object Oriented Programming System



Process of hiding the implementation details

Performing a single task in multiple ways

Process of wrapping code and data together

Process of acquiring properties of other classes

Summary

- We learned some basics of Object Oriented programming.
- We learned what are objects and how to use them in our programs.
- We learned the difference between classes and objects.
- We haven’t learned how to define our own classes yet. This will be covered in the course CITS1001.