Introduction to OOP

Course Title: Programming Language II
Course Code: CSE 111
Semester: Summer 2020

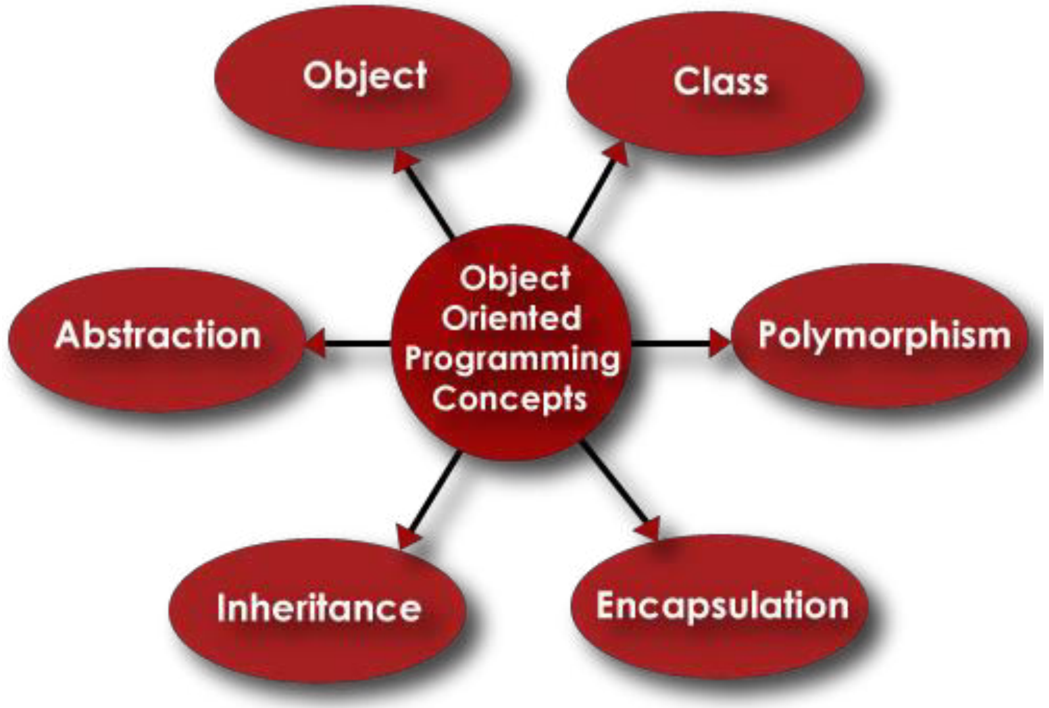
Today’s Lecture

- Concept of OOP
- Pillars of OOP
- Class
- Object

Concept of OOP

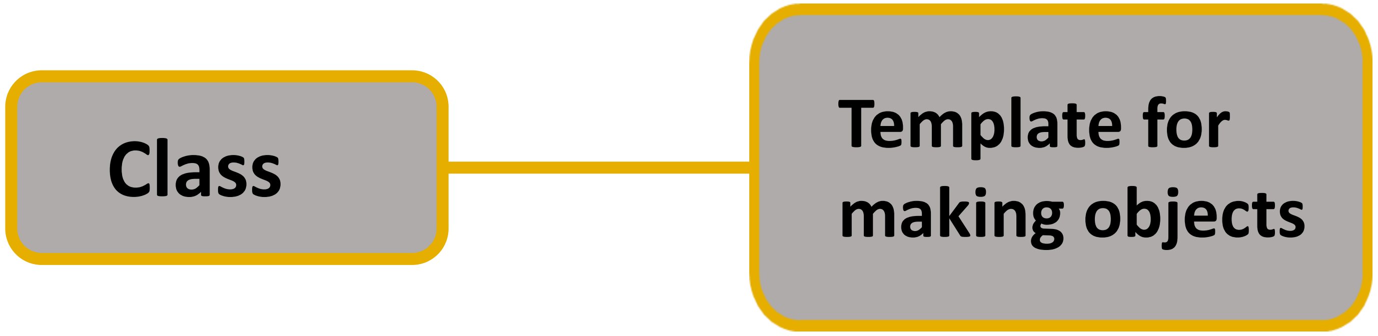
- Provides a means of structuring programs so that properties and behaviors are bundled into individual objects.
- OOP reflects the real world behavior of how things work
- It make visualization easier because it is closest to real world scenarios.
- We can reuse the code through inheritance , this saves time, and shrinks our project.
- There are flexibility through polymorphism

Pillars of OOP



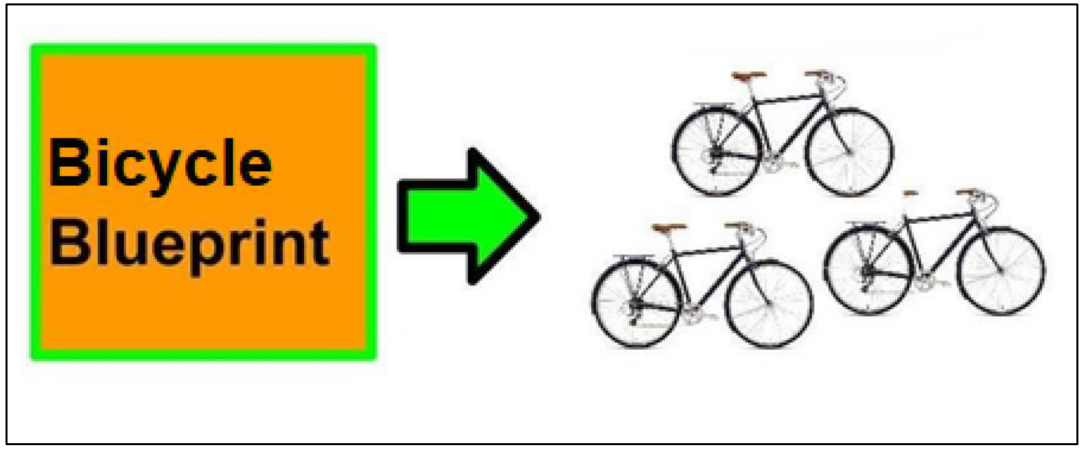
Class

- A class is the blueprint for the objects created from that class
- Each class contains some data definitions(called fields), together with methods to manipulate that data
- When the object is instantiated from the class, an instance variable is created for each field in the class

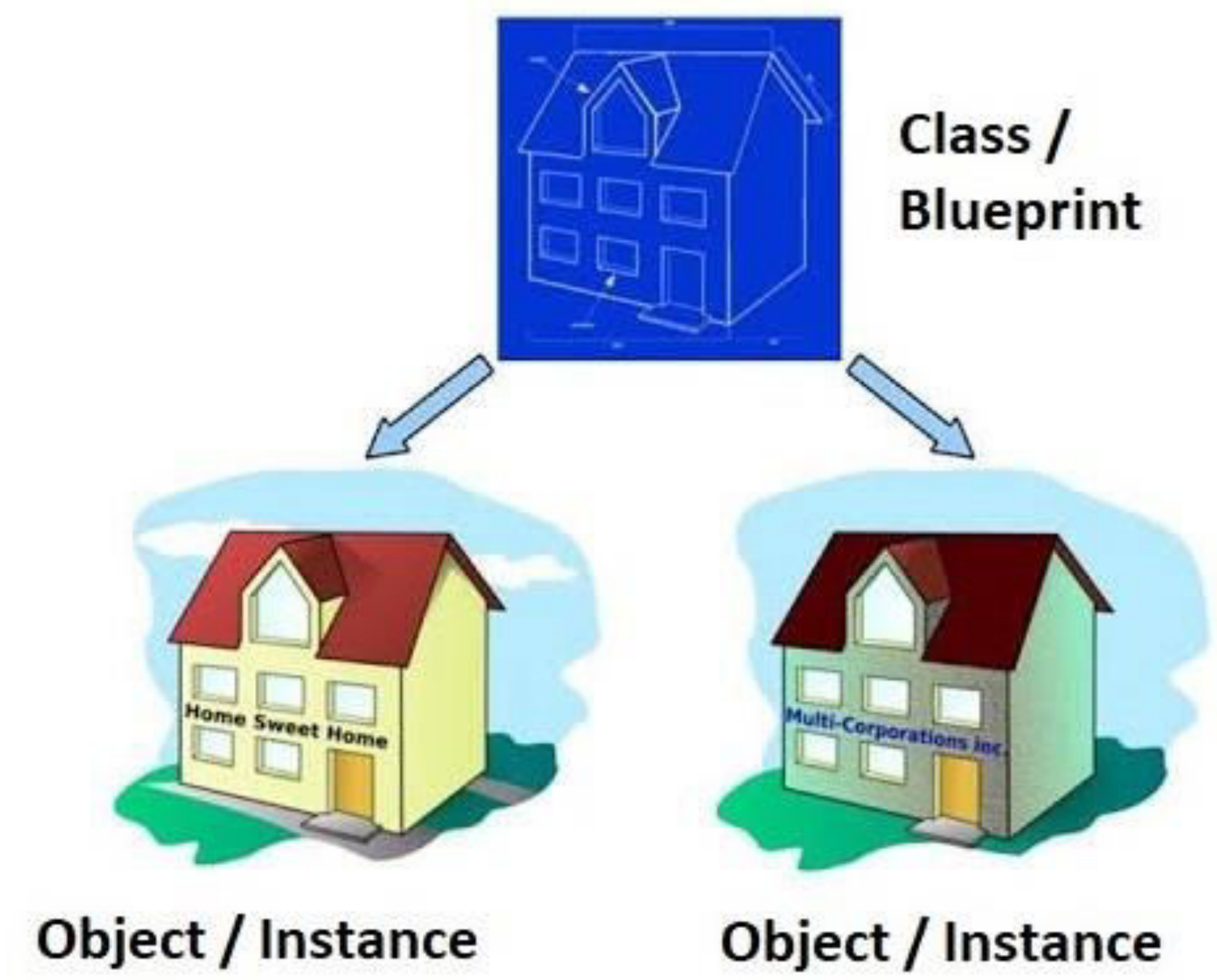


Object

- Objects are the basic run time entities in an object oriented system
- It is an instance of class
- We can say that objects are the variables of the type class



Class & Object



Class Components

- Data (the attributes about it)
- Behavior (the methods)

Data	Method()
• driver_name • num_passenger	• pick_up_passenger() • drop_off_passenger()

Method

- A Python method is like a Python function
- It must be called on an object.
- It must put it inside a class
- A method has a name, and may take parameters and have a return statement

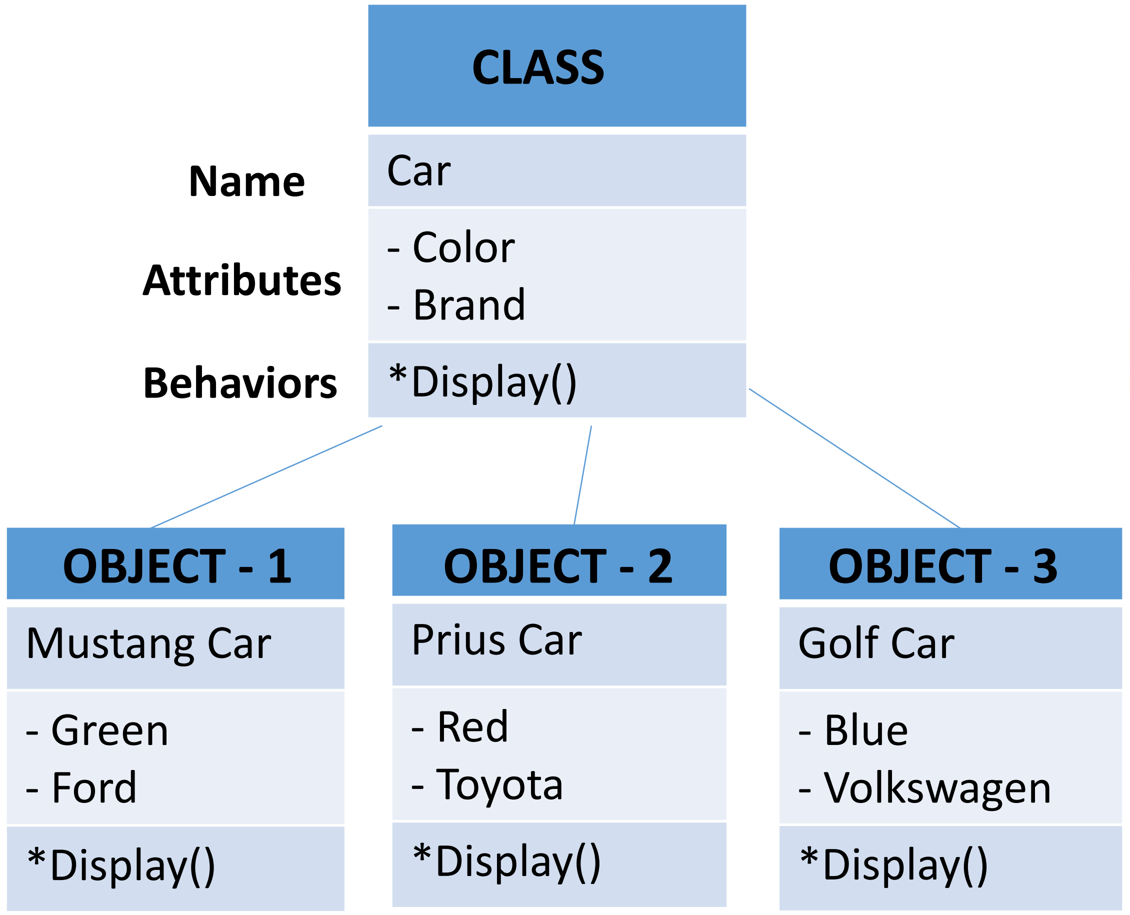
Class Components

Taxi

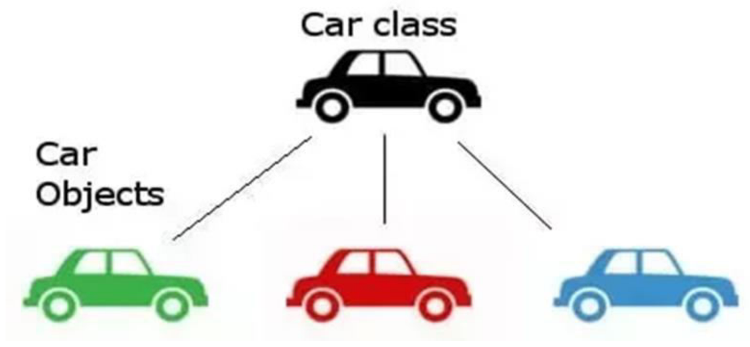
- driver_name: **string**
- num_passenger: **int**

- Pick_up_passenger()
- Drop_off_passenger()

Class and Objects



Blueprint that describes a car



Instances of the cars describe by the blueprint

Next Lecture

- Constructor
  - Non parameterized
  - Parameterized

Thank You!

Introduction to OOP

Course Title: Programming Language II
Course Code: CSE 111
Semester: Summer 2020

Today’s Lecture

- Constructor
    - Non parameterized
    - Parameterized

Constructor

- A special kind of method we use to initialize instance members of that class
- It is used for initializing the instance members when we create the object of a class.
- If you create four objects, the class constructor is called four times.
- Every class must have a constructor, even if it simply relies on the default constructor.
- Constructors can be of two types.

    * Non-parameterized Constructor (Default constructor)

    * Parameterized Constructor

Python __init__

- "__init__" is a reserved method in python classes.
- It is known as a constructor in OOP concepts.
- This method is called when an object is created from the class and it allows the class to initialize the attributes of a class.
- It accepts the **self** -keyword as a first argument which allows accessing the attributes or method of the class.
- We can pass any number of arguments at the time of creating the class object, depending upon the __init__() definition.

Non-parameterized Constructor (default constructor)

- When we do not include the constructor in the class or forget to declare it, then that becomes the default constructor.
- It does not perform any task but initializes the objects
- In the following example, we do not have a constructor but still we are able to create an object for the class.

```
class Employee:
    pass

emp1 = Employee()
emp2 = Employee()
```

Non-parameterized Constructor (default constructor)

```
class Employee:
    def __init__(self):
        print(“Employee object created”)

emp1 = Employee()
emp2 = Employee()
```

Output:

```
Employee object created
Employee object created
```

Python Parameterized Constructor

- The parameterized constructor has multiple parameters along with the **self**.
- It accepts the arguments during object creation

```
class Employee:

    #parameterized constructor
    def __init__(self, name):
        self.name = name  #instance variable
        print(self.name, “created”)

emp1 = Employee("John")  #instance 1
emp2 = Employee("David") #instance 2
```

```
Output:
John created
David created
```

Thank You!