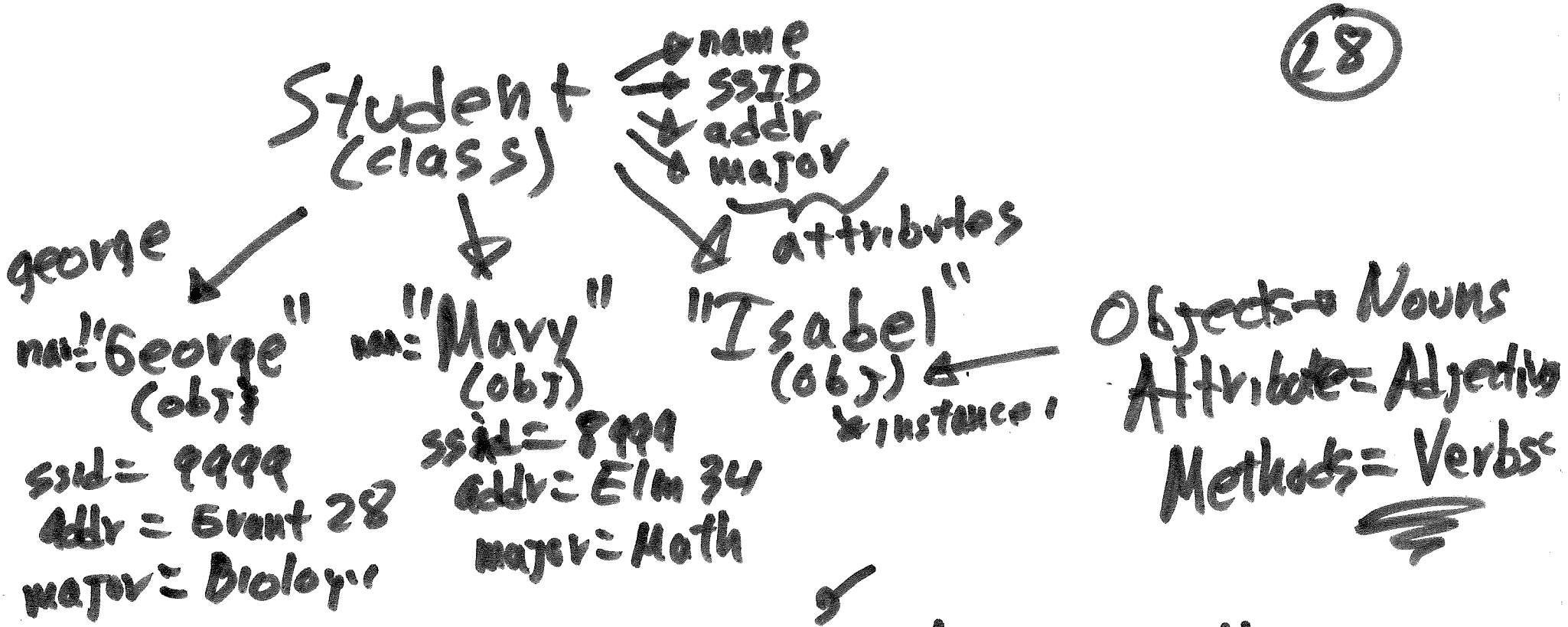
Object Oriented Programming

Chapter 10

- People model the world based on objects.
- Modern languages incorporate "objects" to facilitate programming.
- A "Class" is a type of object.
  Example "Student" is a class for every individual student.
- An object is an instance of a class.
  Example "George" is a member of the class "Student" or you can say an instance of the Student class.



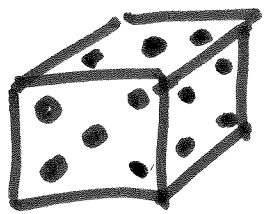
A object has attributes like in this case name, ssid, addr, major

An object also has methods that ar actions the the object can do.

```
george.attendClass()
george.setAddress("Washington 43")
```

Object Oriented Programming (cont).

- **Class** → a type of object
- **Object** → It is an instance of a class that has attributes and methods
- **Attribute** → Characteristics of an object
- **Method** → Functions or activities that object can do.



Example: A Dice class

```
#
#dice.py
# Class definition for a dice with n faces
#
from random import randrange
class Dice:                    # You always have to start with
                               # "class" in a class definition
    def __init__(self,nsides):
        self.nsides=nsides     # Always use "self" as the first
        self.value=1           # parameter in a class method.
```

Name of the class

```
def roll(self):
    self.value = randrange(1, self.nsides+1)
                    #generates a random number
                    #from 1 to nsides inclusive.

def getValue(self):
    return self.value

def setValue(self, value):
    self.value = value
```

range goes from 1 to nsides (10)

To use the dice class we import it from another file,

play.py

```
from dice import *

def Main():
    dice1 = Dice(6) #Create dice1
    dice1.roll() #roll the dice
    print("Val of dice1 is", dice1.getValue())
```

Encapsulation:

+ Once you write a class, and test it, you can use it in other places without having to think about how the class is implemented.
+ When you have to write a large program, think about the objects that make the program. Then write the classes that implement those objects.
+ If an object's implementation is too large divide the object into smaller objects.