Name: Dhwani Chauhan
Roll Number: 21BCP376
Division: 3

# OOPs Principle

May 21, 2022

## Q-1] Implement the OOPs examples provided in the PPT.

[1]:
```
class Animal:
 def __init__(self, name,age1):
   self.givenname = name
   self.age = age1

 def printname(self):
   print(self.givenname,self.age)

class Dog(Animal):
 def __init__ (self, name, age):
   super().__init__ (name, age)
   self.owner = 'Sam'

x = Dog("Scooby", "5")
x.printname()
print(x.owner)
```

```
Scooby 5
Sam
```

[2]:
```
class Animal:
 def __init__(self, name,age1):
  self.givenname = name
  self.age = age1

 def  printname(self):
  print(self.givenname,self.age)

x = Animal("Scooby", "5")
x.printname()
```

```
Scooby 5
```

[3]:
```
class Animal:
 def __init__(self, name,age1):
  self.givenname = name
```

```
  self.age = age1

 def printname(self):
  print(self.givenname,self.age)

class Dog(Animal):
 pass

x = Dog("Scooby", "5")
x.printname()
```

```
Scooby 5
```

[4]:
```
class Dog:
 def __init__ ( self,n,age ):
  self.name = n
  self.lifespan = 20
  self.age = el
  print("Dog created!!")
```

[28]:
```
class Rectangle:
 __length = 0 #privatevariable
 __breadth = 0 #private variable
 def __init__(self): #constructor
    self.__length = 5
    self.__breadth = 3
#printing valuesoftheprivatevariablewithintheclass
    print(self.__length)
    print(self.__breadth)

rec = Rectangle() #object created for the class 'Rectangle'
#printing values of the private variable outside the class using the object␣
 ↪created for the class 'Rectangle'

print(rec.length)
print(rec.breadth)
```

```
5
3
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [28], in <cell line: 14>()
     11 rec = Rectangle() #object created for the class 'Rectangle'
     12 #printing values of the private variable outside the class using the␣
 ↪object created for the class 'Rectangle'
---> 14 print(rec.length)
     15 print(rec.breadth)
```

```
AttributeError: 'Rectangle' object has no attribute 'length'
```

[6]:
```
class Dog():
 def makeNoise(self):
  print("He says bow wow.")

class Cat():
 def makeNoise(self):
  print("He says mew mew")

obj_d = Dog()
obj_c = Cat()

obj_d.makeNoise()
obj_c.makeNoise()
```

```
He says bow wow.
He says mew mew
```

## Q-2] Practice the examples of Inheritance, Encapsulation and Polymorphism provided in the

[7]:
```
class Parrot:

    # class attribute
    species = "bird"

    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age

# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the class attributes
print("Blu is a {}".format(blu.__class__.species))
print("Woo is also a {}".format(woo.__class__.species))

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```

```
Blu is a bird
Woo is also a bird
```

```
Blu is 10 years old
Woo is 15 years old
```

[8]:
```python
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing("'Happy'"))
print(blu.dance())
```

```
Blu sings 'Happy'
Blu is now dancing
```

[9]:
```python
# parent class
class Bird:

    def __init__(self):
        print("Bird is ready")

    def whoisThis(self):
        print("Bird")

    def swim(self):
        print("Swim faster")

# child class
class Penguin(Bird):

    def __init__(self):
        # call super() function
        super().__init__()
        print("Penguin is ready")

    def whoisThis(self):
```

```
        print("Penguin")

    def run(self):
        print("Run faster")

peggy = Penguin()
peggy.whoisThis()
peggy.swim()
peggy.run()
```

```
Bird is ready
Penguin is ready
Penguin
Swim faster
Run faster
```

[10]:
```
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

```
Selling Price: 900
Selling Price: 900
Selling Price: 1000
```

[11]:
```
class Parrot:

    def fly(self):
        print("Parrot can fly")

    def swim(self):
```

```
        print("Parrot can't swim")

class Penguin:

    def fly(self):
        print("Penguin can't fly")

    def swim(self):
        print("Penguin can swim")

# common interface
def flying_test(bird):
    bird.fly()

#instantiate objects
blu = Parrot()
peggy = Penguin()

# passing the object
flying_test(blu)
flying_test(peggy)
```

```
Parrot can fly
Penguin can't fly
```

[ ]: