Student Name:_____ Roll No:_____ Section: _____

Lab Series No 14

Lab 14 –Inheritance and Composition

Lab Objectives:

1. Inheritance and Composition
2. What is inheritance?
3. What is composition?

1. Inheritance and Composition

Classes are written to organize and structure code into meaningful blocks, which can then be used to implement the business logic. These implementations are used in such a way that more complex parts are abstracted away to provide for simpler interfaces which can then be used to build even simpler blocks. While doing this we will find that there are lots of times when we will need to establish relationships between the classes that we build. These relationships can then be established using either inheritance or composition. In this lab you will get to know how to build relationships between classes using inheritance and composition and the syntax that is needed.

2. What is Inheritance?

As we know already that inheritance an object is based on another object. When inheritance is implemented, the methods and attributes that were defined in the base class will also be present in the inherited class. This is generally done to abstract away similar code in multiple classes. The abstracted code will reside in the base class and the previous classes will now inherit from the base class. Python allows the classes to inherit commonly used attributes and methods from other classes through inheritance. We can define a base class in the following manner:

```
class DerivedClassName(BaseClassName):
    pass
```

Let's look at an example of inheritance. In the following example, Rocket is the base class and MarsRover is the inherited class.

Program 1: Write a Python program to show the inheritance in real world problem of rocket and launcher.

```
class Rocket:
    def_init_(self, name, distance):
        self.name = name
        self.distance = distance

    def launch(self):
```

```
        return "%s has reached %s" % (self.name, self.distance)


class MarsRover(Rocket): # inheriting from the base class
    def_init_(self, name, distance, maker):
        Rocket._init_(self, name, distance)
        self.maker = maker

    def get_maker(self):
        return "%s Launched by %s" % (self.name, self.maker)


if_name_== "_main__":
    x = Rocket("simple rocket", "till stratosphere")
    y = MarsRover("mars_rover", "till Mars", "ISRO")
    print(x.launch())
    print(y.launch())
    print(y.get_maker())
```

Output:

3. What is Composition?

In composition, we do not inherit from the base class but establish relationships between classes through the use of instance variables that are references to other objects. Talking in terms of pseudocode you may say that

```
class GenericClass:
    define some attributes and methods

class ASpecificClass:
    Instance_variable_of_generic_class = GenericClass

# use this instance somewhere in the class
    some_method(Instance_variable_of_generic_class)
```

So you will instantiate the base class and then use the instance variable for any business logic.

To achieve composition, you can instantiate other objects in the class and then use those instances. For example in the below example we instantiate the Rocket class using self.rocket and then using self.rocket in the method get_maker.

Program 2: Write a Python program to show the composition instead of inheritance in real world problem of rocket and launcher.

```
class MarsRoverComp():
    def_init_(self, name, distance, maker):
        self.rocket = Rocket(name, distance) # instantiating the
base

        self.maker = maker

    def get_maker(self):
        return "%s Launched by %s" % (self.rocket.name,
self.maker)


if__name__== "_main__":
    z = MarsRover("mars_rover2", "till Mars", "ISRO")
    print(z.launch())
    print(z.get_maker())
```

Output:

In composition one of the classes is composed of one or more instance of other classes. In other words one class is container and other class is content and if you delete the container object then all of its contents objects are also deleted.

What is the Difference between Inheritance and Composition?

In Inheritance, a class is inherited (extended) by a new sub-class that will add custom attributes and behavior to the inherited ones.

In Composition, a class is utilized by creating an instance of it, and including that instance inside another larger object.

To make it simpler and easier to remember, let's say it in one sentence:

Inheritance extends (inherits) a class, while Composition uses an instance of the class.

Program 3:

```
class Salary:
    def_init_(self, pay):
        self.pay = pay

    def get_total(self):
        return (self.pay*12)


class Employee:
    def_init_(self, name, age ,pay, bonus):
        self.name=name
        self.age=age
        self.pay = pay
        self.bonus = bonus
        self.obj_salary = Salary(self.pay)

    def annual_salary(self):
        return "Total: " + str(self.obj_salary.get_total() +
self.bonus)


obj_emp = Employee( "ali", 35 ,600, 500)
print(obj_emp.annual_salary())
```

Output:

Programming Exercise

1. Create a class People and Birthday and perform the composition
   Create class `People` **and** `Birthday` which contains instance variables and also implement following methods `printDetails()` to print the information