TASK 8

DHARMA DEVI K
ARULMIGU MEENAKSHI AMMAN COLLEGE OF ENGINEERING

B. TECH (INFORMATION TECHNOLOGY) – FINAL YEAR

1. Using inheritance, one class can acquire the properties of others. Consider the following Animal class: This class has only one method, walk. Next, we want to create a Bird class that also has a fly method. We do this using extends keyword. Finally, we can create a object in Bird class that can call this method both fly and walk.

PROGRAM:

```
class Animal:
    def walk(self):
        print("This animal walks.")
class Bird(Animal):
    def fly(self):
        print("This bird flies.")
bird_object = Bird()
bird_object.walk()
bird_object.fly()
```

OUTPUT:

```
This animal walks.
This bird flies.
```

2. Using inheritance, one class can acquire the properties of others. Consider the following Vechile class: This class has only one method, type of vechile Next, we want to create a Car class that also has a drive method. We do this using extends keyword. Finally, we can create a object Car class that can call this method both type of vechile and drive.

PROGRAM:

```
class Vehicle:
    def type_of_vehicle(self):
        print("This is a vehicle.")
class Car(Vehicle):
    def drive(self):
        print("This car drives on the road.")
```

```
car_object = Car()
car_object.type_of_vehicle()
car_object.drive()
```

OUTPUT:

```
This is a vehicle.
This car drives on the road.
```

3. Using inheritance, one class can acquire the properties of others. Consider the following Shape class: This class has only one method, display. Next, we want to create a two class Rectangle and cube that also has a two method area and volume. We do this using extends keyword. Finally, we can create a object in cube class that can call this method display, area and volume.

PROGRAM:

```
class Shape:
  def display(self):
    print("This is a shape.")

class Rectangle(Shape):
  def area(self, length, width):
    print("Area of rectangle:", length * width)

  def volume(self, length, width, height):
    print("Volume of rectangle:", length * width * height)

class Cube(Shape):
  def area(self, side):
    print("Area of cube:", 6 * (side ** 2))

  def volume(self, side):
    print("Volume of cube:", side ** 3)

rectangle_object = Rectangle()
rectangle_object.display()
rectangle_object.area(5, 3)

rectangle_object.volume(5, 3, 2)


cube_object = Cube()

cube_object.display()

cube_object.area(3)

cube_object.volume(3)
```

OUTPUT:

```
This is a shape.

Area of rectangle: 15

Volume of rectangle: 30

This is a shape.

Area of cube: 54

Volume of cube: 27
```

4. Using inheritance, one class can acquire the properties of others. Consider the following Add class: This class has only one method, addition. Next, we want to create a three class Sub, Mul and Div that also has a three method subtraction, Multiplication and division. We do this using extends keyword. Finally, we can create a object in division class that can call this method .addition, subtraction, Multiplication and division.

PROGRAM:

```
class Add:

    def addition(self, a, b):

        return a + b


class Sub(Add):

    def subtraction(self, a, b):

        return a - b


class Mul(Add):

    def multiplication(self, a, b):

        return a * b
```

```
class Div(Add):
    def division(self, a, b):
        if b != 0:
            return a / b
        else:
            return "Number is zero"

add_object = Add()
sub_object = Sub()
mul_object = Mul()
div_object = Div()

print("Addition:", add_object.addition(10, 5))
print("Subtraction:", sub_object.subtraction(10, 5))
print("Multiplication:", mul_object.multiplication(10, 5))
print("Division:", div_object.division(10, 5))
```

OUTPUT:

```
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
```

5. We are writing the program where class B, C and D extends class A. Next, we want to create a four class A,B,C and D that also has a Four method display1, display2, display 3, display4. Finally, we can create a object in B,C,D class that can call this all method.

PROGRAM:

```
class A:
    def display1(self):
        print("This is display1 in class A")


class B(A):
    def display2(self):
        print("This is display2 in class B")


class C(A):
    def display3(self):
        print("This is display3 in class C")


class D(A):
    def display4(self):
        print("This is display4 in class D")
b_object = B()
c_object = C()
d_object = D()


b_object.display1()
b_object.display2()


c_object.display1()
c_object.display3()


d_object.display1()
d_object.display4()
```

OUTPUT:

```
This is display1 in class A
This is display2 in class B
This is display1 in class A
This is display3 in class C
This is display1 in class A
This is display4 in class D
```