LAB ASSIGNMENT 14

U24CS076

RUSHANG BAGADA

1. Write a python programs that illustrate how the following forms of inheritance are supported:

a. Single inheritance

b. Multiple inheritance

c. Multilevel inheritance

d. Hierarchical inheritance

```
# a. Single inheritance
class Parent:
    def display(self):
        print("This is the parent class.")

class Child(Parent):
    def show(self):
        print("This is the child class.")

# Creating objects
print("Single Inheritance:")
parent = Parent()
child = Child()
parent.display()
child.show()
child.display()
```

```
# b. Multiple inheritance
class Father:
    def display(self):
        print("This is the father class.")

class Mother:
    def show(self):
        print("This is the mother class.")

class Child(Father, Mother):
    def introduce(self):
        print("This is the child class.")

print("\nMultiple Inheritance:")
child = Child()
child.display()
child.show()
child.introduce()

# c. Multilevel inheritance
class Grandparent:
    def greet(self):
        print("This is the grandparent class.")

class Parent(Grandparent):
    def display(self):
        print("This is the parent class.")
```

```
class Child(Parent):
    def show(self):
        print("This is the child class.")

print("\nMultilevel Inheritance:")
child = Child()
child.greet()
child.display()
child.show()

# d. Hierarchical inheritance
class Parent:
    def display(self):
        print("This is the parent class.")

class Child1(Parent):
    def show(self):
        print("This is the first child class.")

class Child2(Parent):
    def show(self):
        print("This is the second child class.")

print("\nHierarchical Inheritance:")
child1 = Child1()
child2 = Child2()
child1.display()
```

```
child1.show()
child2.display()
child2.show()
```

OUTPUT

```
Single Inheritance:
This is the parent class.
This is the child class.
This is the parent class.

Multiple Inheritance:
This is the father class.
This is the mother class.
This is the child class.

Multilevel Inheritance:
This is the grandparent class.
This is the parent class.
This is the child class.

Hierarchical Inheritance:
This is the parent class.
This is the first child class.
This is the parent class.
This is the second child class.
```

2. Write a python program calculates the percentage of a student using multilevel inheritance.

a. (Enter Marks for 5 Subjects: Maths:--, Chem:--, Phy:--, WP:--, DS:--

, b. Percentage of a student : —)

```
class Student:
    def __init__(self, name):
        self.name = name


class Marks(Student):
    def __init__(self, name, marks):
        super().__init__(name)
        self.marks = marks


class Percentage(Marks):
    def calculate_percentage(self):
        total = sum(self.marks.values())
        percentage = (total / 500) * 100
        return percentage


# Input marks for 5 subjects
marks = {
    "Maths": int(input("Enter marks for Maths: ")),
    "Chemistry": int(input("Enter marks for Chemistry: ")),
    "Physics": int(input("Enter marks for Physics: ")),
    "Web Programming": int(input("Enter marks for Web Programming: ")),
    "Data Structures": int(input("Enter marks for Data Structures: "))
}


student = Percentage("John Doe", marks)
print(f"\nPercentage of {student.name}: {student.calculate_percentage()}%")
```

OUTPUT:

Enter marks for Maths: 56

Enter marks for Chemistry: 45

Enter marks for Physics: 78

Enter marks for Web Programming: 96

Enter marks for Data Structures: 55

Percentage of John Doe: 66.0%

3. Write a Python program to create a class Vehicle, Twowheeler and Fourwheeler. Then, derive classes Car and Scooter from it to illustrate the multiple inheritance.

```
class Vehicle:
  def __init__(self, name):
    self.name = name

class TwoWheeler(Vehicle):
  def __init__(self, name, type):
    super().__init__(name)
    self.type = type

class FourWheeler(Vehicle):
  def __init__(self, name, type):
    super().__init__(name)
    self.type = type

class Car(FourWheeler):
  def details(self):
    print(f"Car Name: {self.name}, Type: {self.type}")
```

```
class Scooter(TwoWheeler):
    def details(self):
        print(f"Scooter Name: {self.name}, Type: {self.type}")

car = Car("Honda City", "Sedan")
scooter = Scooter("Activa", "Scooter")

print("Vehicle Details:")
car.details()
scooter.details()
```

OUTPUT:

1. Details: Car x1 Red 20000

Car max speed is 240

Car change 7 gear

2. Details: Truck x1 white 75000

Vehicle max speed is 150

Vehicle change 6 gear

4. Write a python program to illustrate polymorphism using this example.

```
class Vehicle:
    def details(self, name, color, price):
        print(f"Details: {name}{color}{price}")

    def max_speed(self):
        print("Vehicle max speed is 150")
```

```
    def change_gear(self):
        print("Vehicle changes 6 gears")

class Car(Vehicle):
    def max_speed(self):
        print("Car max speed is 240")

    def change_gear(self):
        print("Car changes 7 gears")

class Truck(Vehicle):
    pass

# Polymorphism in action
car = Car()
truck = Truck()

print("Car:")
car.details("Car", "Red", 20000)
car.max_speed()
car.change_gear()

print("\nTruck:")
truck.details("Truck", "White", 75000)
truck.max_speed()
truck.change_gear()
```

Vehicle Details:

Car Name: Honda City, Type: Sedan

Scooter Name: Activa, Type: Scooter

OUTPUT:

Car:

Details: Car Red 20000

Car max speed is 240

Car changes 7 gears

Truck:

Details: Truck White 75000

Vehicle max speed is 150

Vehicle changes 6 gears

5. Write a python program to create two classes CAT and DOG . They share a similar structure and have the same method names Info() and Make_sound().

```
class Cat:
  def info(self):
    print("I am a cat. I like to meow.")

  def make_sound(self):
    print("Meow!")

class Dog:
  def info(self):
    print("I am a dog. I like to bark.")

  def make_sound(self):
    print("Woof!")
```

```
# Polymorphism in action
animals = [Cat(), Dog()]

for animal in animals:
    animal.info()
    animal.make_sound()
```

OUTPUT:

```
I am a cat. I like to meow.
Meow!
I am a dog. I like to bark.
Woof!
```