INHERITANCE :

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:

```
    // Parent class
class Animal {
  walk() {
    console.log("This animal can walk.");
  }
}

// Child class extending the Animal class
class Bird extends Animal {
  fly() {
    console.log("This bird can fly.");
  }
}

// Creating an instance of Bird class
const bird = new Bird();
bird.walk(); // Calling method from parent class
bird.fly();  // Calling method from child class
```

Output:

```
This animal can walk.
```

This bird can fly.

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:

```
// Parent class class Vehicle { type_of_vehicle() {

console.log("This is a general vehicle.");

 }

 }

// Child class extending the Vehicle class class Car extends Vehicle

{

drive() { console.log("This car can be driven.");

}

 }

// Creating an instance of Car class const myCar = new Car();

 myCar.type_of_vehicle();

// Calling method from parent class myCar.drive(); // Calling method from child class
```

Output:

This is a general vehicle.

This car can be driven.

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:

```
// Parent class class Shape

{

display()

{

console.log("This is a shape.");

}}

// Child class Rectangle extending Shape class Rectangle extends Shape

{

area(length, breadth)

{ return length * breadth;

}}

// Child class Cube extending Rectangle class Cube extends Rectangle

{

volume(length, breadth, height)

{ return length * breadth * height;

}}

// Creating an instance of Cube class const myCube = new Cube(); myCube.display();

// Calling method from parent class console.log("Area of Rectangle:", myCube.area(5, 10));
```

// Calling area method from Rectangle class console.log("Volume of Cube:", myCube.volume(5, 10, 3));

// Calling volume method from Cube class

Output:

This is a shape.

Area of Rectangle: 50

Volume of Cube: 150

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:

```
// Parent class
class Add {



    addition(a, b) {

      return a + b;
    }
}

// Sub class extending

 Add class Sub extends Add {

subtraction(a, b)

{

return a - b;
```

```
}

}
// Mul class extending

 Sub class Mul extends Sub

{ multiplication(a, b)

{

return a * b;

}

}

// Div class extending

 Mul class Div extends Mul

 { division(a, b)

{ if (b !== 0)

 { return a / b;

}

else {

 return "Division by zero is not allowed.";

}

}

} // Creating an instance of

Div class const myCalc = new Div();

// Calling methods from different levels of inheritance
console.log("Addition:", myCalc.addition(10, 5));
```

```
// Method from Add class console.log("Subtraction:",
myCalc.subtraction(10, 5));

// Method from Sub class console.log("Multiplication:",
myCalc.multiplication(10, 5));

 // Method from Mul class console.log("Division:", myCalc.division(10,
5));

// Method from Div class
```

Output:

```
Addition: 15

Subtraction: 5

Multiplication: 50

Division: 2
```

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,display3,display4. Finally, we can create a object in B,C,D class that can call this all **method.**

Program:

```
// Parent class A

class A {

    display1() {

        console.log("This is display1 from class A.");

    }

}



// Class B extending A
```

```
class B extends A {
    display2() {
        console.log("This is display2 from class B.");
    }
}

// Class C extending A
class C extends A {
    display3() {
        console.log("This is display3 from class C.");
    }
}

// Class D extending A
class D extends A {
    display4() {
        console.log("This is display4 from class D.");
    }
}

// Creating objects of each class
const objB = new B();
const objC = new C();
```

```
const objD = new D();

// Calling methods using B object
console.log("Using object of class B:");
objB.display1(); // Method from class A
objB.display2(); // Method from class B

// Calling methods using C object
console.log("\nUsing object of class C:");
objC.display1(); // Method from class A
objC.display3(); // Method from class C

// Calling methods using D object
console.log("\nUsing object of class D:");
objD.display1(); // Method from class A
objD.display4(); // Method from class D
```

Output:

Using object of class B:

This is display1 from class A.

This is display2 from class B.

Using object of class C:

This is display1 from class A.

This is display3 from class C.

Using object of class D:

This is display1 from class A.

This is display4 from class D.