**Assignment - 6**

Problem Statement: Inheritance, Access Modifiers, Encapsulation, and Static

1. Create a class BMW with ‘make, model, and year’ properties, Create two child classes to BMW and perform some common functionality with start and stop methods, Create a child class objects and display results.
2. class BMW{
3. make:string;model:string;year:number;
4. constructor(make:string,model:string,year:number){
5. this.make=make;
6. this.model=model;
7. this.year=year;
8. }
9. start() {
10. console.log("Starting.....");
11. }
12. stop() {
13. console.log("Stopped.....");
14. }
15. }
16. class child1 extends BMW{
17. series:string;
18. constructor(make:string,model:string,year:number,series:string){
19. super(make,model,year);
20. this.series=series;
21. }
22. }
23. class child2 extends BMW{
24. series:string;
25. constructor(make:string,model:string,year:number,series:string){
26. super(make,model,year);
27. this.series=series;
28. }
29. }
30. var c1=new child1("Honda","Civic",2018,"ThirdSeries");
31. c1.start();
32. console.log("Series: ",c1.series,"\n","Make: ",c1.make,"\n","Model: ",c1.model,"\n","Year: ",c1.year);
33. c1.stop();
34. var c2=new child2("Honda","Civic",2020,"FifthSeries");
35. c2.start();
36. console.log("Series: ",c2.series,"\n","Make: ",c2.make,"\n","Model: ",c2.model,"\n","Year: ",c2.year);
37. c2.stop();

Graphical user interface, text, application

Description automatically generated

2. Perform sample overriding functionality program for BMW class.

class BMW{

    make:string;model:string;year:number;

    constructor(make:string,model:string,year:number){

        this.make=make;

        this.model=model;

        this.year=year;

    }

    start() {

        console.log("Starting.....");

    }

    stop() {

        console.log("Stopped.....");

    }

}

class child1 extends BMW{

    series:string;

    constructor(make:string,model:string,year:number,series:string){

        super(make,model,year);

        this.series=series;

    }

    start() {

        console.log("Button Start.....");

    }

    stop() {

        console.log("Button Stop.....");

    }

}

class child2 extends BMW{

    series:string;

    constructor(make:string,model:string,year:number,series:string){

        super(make,model,year);

        this.series=series;

    }

    start() {

        console.log("Remote Start.....");

    }

    stop() {

        console.log("Remote Stop.....");

    }

}

var c1=new child1("Honda","Civic",2018,"ThirdSeries");

console.log("Series: ",c1.series,"\n","Make: ",c1.make,"\n","Model: ",c1.model,"\n","Year: ",c1.year);

c1.start();

c1.stop();

var c2=new child2("Honda","Civic",2020,"FifthSeries");

console.log("Series: ",c2.series,"\n","Make: ",c2.make,"\n","Model: ",c2.model,"\n","Year: ",c2.year);

c2.start();

c2.stop();

3. Create a student class with name property and perform access modifiers program using public and read-only.

Text

Description automatically generated

4. Perform sample encapsulation operation using private properties and access the properties with get method in typescript.

class student{

      //readonly name:string="John";

       \_name:string;

       display(){

           console.log(this.\_name);

       }

       get getName():string{

           return this.\_name;

       }

       set setName(name:string){

           this.\_name=name;

       }

}

var s=new student();

s.setName="Rossy";

console.log(s.getName);

5. Create a check class with ‘bankName, customerName, accNo, and routingNo’ properties, make bankName as static property, create an object and access the property in browser.

class check{

    static name1:string="John";

    display(){

        check.name1="Rossy";

        console.log(check.name1);

    }

}

var c=new check();

console.log(c.display());