1)

interface Producto{

    Nombre:string;

    Precio:number;

    Oferta?:boolean;

}

const prendas : Array <Producto> = [

    {"Nombre":'camisa','Precio':60},

    {"Nombre":'pantalones','Precio':45,'Oferta':true},

    {"Nombre":'polo','Precio':35},

    {"Nombre":'slip','Precio':15,'Oferta':true}

    ];

prendas.forEach(function(producto:Producto) : void {

    if(producto.Oferta != undefined){

        console.log(`${producto.Nombre} con oferta siendo ${producto.Oferta}`);

    }

});

prendas.sort(function(a:Producto, b:Producto): any{

    if(a.Precio < b.Precio){

        return 1;

    }else if(b.Precio < a.Precio){

        return -1;

    }else{

        return 0;

    }

});

console.log(prendas);

2)

class Persona{

    Nombre:string;

    Edad:number;

    DNI:string;

    Domicilio:string;

    constructor(Nombre?:string,Edad?:number,DNI?:string,Domicilio?:string){

        this.Nombre =Nombre ?? 'Toni';

        this.Edad =Edad ?? 22;

        this.DNI =DNI ?? '77777777F';

        this.Domicilio =Domicilio ?? 'Calle del testeo';

    }

    private getNombre (){

        return this.Nombre;

    }

    private getEdad (){

        return this.Edad;

    }

    private getDNI (){

        return this.DNI;

    }

    private getDomicilio (){

        return this.Domicilio;

    }

    private setNombre (nuevoNombre:string){

        this.Nombre = nuevoNombre;

    }

    private setEdad (nuevaEdad:number){

        this.Edad = nuevaEdad;

    }

    private setDomicilio (nuevoDomicilio:string){

        this.Nombre = nuevoDomicilio;

    }

    private setDNI (nuevoDNI:string){

        this.Nombre = nuevoDNI;

    }

    public MostrarInfo(){

        console.log(`La persona ${this.Nombre} de edad ${this.Edad} y

        dni ${this.DNI} vive en ${this.Domicilio}`);

    }

}

let persona = new Persona();

let persona2 = new Persona("Marcos alejandro", 400, "66666666P", "La calle de adagio");

persona.MostrarInfo();

persona2.MostrarInfo();

3)

class Cafetera{

    CapacidadMaxima:number;

    CantidadActual:number;

    constructor(){

        this.CapacidadMaxima = 1000;

        this.CantidadActual = 0;

    }

    private getCapacidadMaxima (){

        return this.CapacidadMaxima;

    }

    private getCantidadActual (){

        return this.CantidadActual;

    }

    private setCapacidadMaxima (nuevaCapMax:number){

        this.CapacidadMaxima = nuevaCapMax;

    }

    private setCantidadActual (nuevaCanAct:number){

        this.CantidadActual = nuevaCanAct;

    }

    public LlenarCafetera(){

       this.setCantidadActual(this.getCapacidadMaxima());

       console.log(`Se ha llenado la cafetera`);

    }

    public ServirTaza(taza:number){

        if(taza > this.getCantidadActual()){

            console.log(`La capacidad de la cafetera no llegaba para llenar la taza, por tanto se ha llenado con:`);

            this.setCantidadActual(0);

        }else{

            this.setCantidadActual(this.getCantidadActual() - taza);

        }

    }

    public VaciarCafetera (){

        this.setCantidadActual(0);

        console.log(`Se ha vaciado la cafetera`)

    }

    public AgregarCafe(cafe:number){

        console.log(`Se ha agregado esta cantidad de cafe:`);

        this.setCantidadActual(this.getCantidadActual()+cafe);

    }

}

let cafetera = new Cafetera();

cafetera.LlenarCafetera();

cafetera.ServirTaza(1500);

cafetera.VaciarCafetera();

cafetera.AgregarCafe(500);

4)

class Producto{

    NombreProd:string;

    NLote:number;

    constructor(n:string, l:number){

        this.NombreProd = n;

        this.NLote = l;

    }

    public getNombreProd(){

        return this.NombreProd;

    }

    public getNLote(){

        return this.NLote;

    }

    public setNombreProd(nuevoNombre:string){

        this.NombreProd = nuevoNombre;

    }

    public setNLote(nuevoLote:string){

        this.NombreProd = nuevoLote;

    }

}

class ProductoFresco extends Producto{

    FEnvasado:string;

    PaisOr:string;

    constructor(NombreProd:string, NLote:number, FEnvasado:string, PaisOr:string){

        super(NombreProd,NLote);

        this.FEnvasado = "15/05/2023"

        this.PaisOr = "Arabia saudí";

    }

    public getFEnvasado(){

        return this.FEnvasado;

    }

    public getPaisOr(){

        return this.FEnvasado;

    }

    public setFEnvasado(nuevoFEnvasado:string){

        this.FEnvasado = nuevoFEnvasado;

    }

    public setPaisOr(nuevoPaisOr:string){

        this.PaisOr = nuevoPaisOr;

    }

    public Mostrar(){

        console.log(`Como producto fresco tenemos ${prodFresco.getNombreProd()} que es del lote ${prodFresco.getNLote()} con fecha de envasado: ${prodFresco.getFEnvasado()} y origen: ${prodFresco.getPaisOr()}`);

    }

}

class ProductoRefrigerado extends Producto{

    CodOrg:number;

    constructor(NombreProd:string, NLote:number, CodOrg:number){

        super(NombreProd,NLote);

        this.CodOrg = 2343553

    }

    public getCodOrg(){

        return this.CodOrg;

    }

    public setCodOrg(nuevoCodOrg:number){

        this.CodOrg = nuevoCodOrg;

    }

   public Mostrar(){

        console.log(`Como producto refrigerado tenemos ${prodRefrigerado.getNombreProd()} que es del lote: ${prodRefrigerado.getNLote()} y codigo de origen: ${prodRefrigerado.getCodOrg()}`);

    }

}

class ProductoCongelado extends Producto{

    Temp:number;

    constructor(NombreProd:string, NLote:number, Temp:number){

        super(NombreProd,NLote);

        this.Temp = -20;

    }

    public getTemp(){

        return this.Temp;

    }

    public setTemp(nuevoTemp:number){

        this.Temp = nuevoTemp;

    }

   public Mostrar(){

        console.log(`Como producto congelado tenemos ${prodCongelado.getNombreProd()} que tiene del lote: ${prodCongelado.getNLote()} y temperatura: ${prodCongelado.getTemp()}`);

    }

}

let prodFresco = new ProductoFresco("Huevos",3,"12/09/22","España");

let prodRefrigerado = new ProductoRefrigerado("Leche",2,123123);

let prodCongelado = new ProductoCongelado("Carne",1,-20);

prodFresco.Mostrar();

prodRefrigerado.Mostrar();

prodCongelado.Mostrar();