















```
s clases_01.ts ×
typescript > TS clases_01.ts > ...
      class Vehiculo{
          marca:string
          anio:number
          modelo:string
          precio:number
          velocidad:number
          constructor(_marca:string,_anio:number,_modelo:string,_precio:number) {
              this.marca = _marca
              this.anio = _anio
              this.modelo = _modelo
              this.precio = precio
              this.velocidad = 0
          acelerar():void{
              if (this.velocidad < 200){
                  this.velocidad = this.velocidad + 10
                  console.log(`El vehiculo va a una velocidad de ${this.velocidad} km/hr`)
                  this.velocidad = 200
                  console.log(`El vehiculo va a su velocidad maxima que es ${this.velocidad} km/hr`)
          frenar():void{
              if (this.velocidad > 10){
                  this.velocidad = this.velocidad - 10
                  console.log(`El vehiculo va a una velocidad de ${this.velocidad} km/hr`)
                  this.velocidad = 0
                  console.log(`El vehiculo se encuentra parado su velocidad es ${this.velocidad} km/hr`)
```







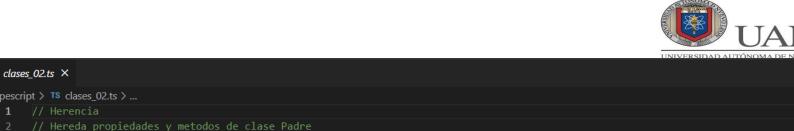


```
typescript > Ts clases_01.ts > ...

36
37     const auto_01 = new Vehiculo("Ford",2023,"Lobo",1300000)
38     auto_01.acelerar();
40     auto_01.acelerar();
41     auto_01.acelerar();
42
43     auto_01.frenar();
44     auto_01.frenar();
45     auto_01.frenar();
46     auto_01.frenar();
47     auto_01.frenar();
48     auto_01.frenar();
49
```









```
TS clases_02.ts X
typescript > TS clases_02.ts > ...
      // Hereda propiedades y metodos de clase Padre
      class Padre{
          nombre:string
          edad:number
          constructor(_nombre:string, _edad:number){
              this.nombre=_nombre
              this.edad= edad
          mostrarNombre():void{
              console.log(`el nombre es: ${this.nombre}`)
      class Hijo extends Padre{
          apellido:string
          constructor(_nombre:string, _edad:number,_apellido:string){
              // super ejecuta el constructor de la clase padre
              // que recibe dos parametros de acuerdo al tipo de dato
              super(_nombre,_edad)
              this.apellido = _apellido
          mostrarDatosHijo():void{
              console.log(`Los datos de la pesona son:`)
              console.log(`nombre: ${this.nombre}, apellido: ${this.apellido} y edad ${this.edad}`)
      const persona = new Hijo("Jose", 25,"Munoz")
      persona.mostrarNombre()
      persona.mostrarDatosHijo()
```









```
TS clases_03.ts X
typescript > TS clases_03.ts > ...
       // Modificadores Publicos
       // Por default las clases, propiedades y metodos son publicos
       class Animal01{
           nombre:string
           patas:number
           constructor(_nombre:string, _patas:number){
               this.nombre = nombre
               this.patas = _patas
           moverse(){
               console.log(`El ${this.nombre} se esta moviendo`)
 11
 12
 13
 14
 15
       const animal01 = new Animal01("perro",4)
       animal01.moverse();
 17
       animal01.nombre = "gato"
 18
       animal01.moverse()
```







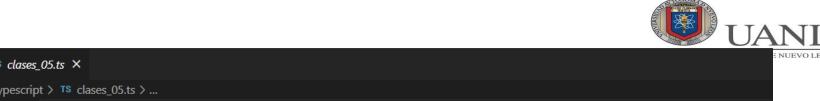




```
TS clases 04.ts X
typescript > TS clases_04.ts > ...
      // Modificadores Privados
       // Por default las clases, propiedades y metodos son publicos
       class Animal{
           private nombre:string
           private patas:number
           public constructor(_nombre:string, _patas:number){
                this.nombre = _nombre
                this.patas = _patas
           public moverse(){
                console.log(`El ${this.nombre} se esta moviendo`)
 11
 12
       const animal02 = new Animal("cocodrilo",4)
       animal02.moverse();
       // No se puede asignar el valor de una propiedad privada
       animal02.moverse()
```









```
TS clases_05.ts X
typescript > TS clases_05.ts > ...
  1 // Modificadores Protegidos
      // O accesibles de la clase heredada
      class Padre02{
          private nombre:string
          private apellido:string
          protected edad:number
          public constructor(_nombre:string, _apellido:string){
               this.nombre = _nombre
               this.apellido = _apellido
          mostrarDatosPadre(){
               console.log(`Los datos del hijo son: nombre ${this.nombre}, apellido${this.apellido} `)
      class Hijo02 extends Padre02{
          constructor(_nombre:string,_apellido:string,_edad){
               super(_nombre,_apellido)
               this.edad = _edad
          mostrarDatosHijo(){
               console.log(`La edad es: ${this.edad}`)
      const hijo02 = new Hijo02("Juan", "Sin Miedo", 6)
      hijo02.mostrarDatosPadre();
      hijo02.mostrarDatosHijo();
```









```
TS clases_06.ts X
typescript > TS clases_06.ts > ...
      class Persona{
          private nombre:string
          private edad:number
          constructor(_nombre:string, _edad:number){
              this.nombre = _nombre
              this.edad = _edad
          getNombre():string{
              return this.nombre
          getEdad():number{
              return this.edad
          setNombre(_nombre:string):void{
              this.nombre = _nombre
          setEdad(_edad:number):void{
              this.edad = _edad
          mostrarDatosPersona(){
              console.log(`Los datos de la persona son: nombre ${this.nombre} y edad ${this.edad}`)
      let persona01 = new Persona("Jose",25)
      persona01.mostrarDatosPersona()
      persona01.setNombre("Pepe")
      persona01.setEdad(26)
      persona01.mostrarDatosPersona()
```









```
TS clases_07.ts X
typescript > TS clases_07.ts > ...
  1 // la clase abstracta puede declarar metodos
       abstract class Animal02{
           abstract ruido():void;
           camina():void{
               console.log(`El animal esta caminando`)
       class gato extends Animal02{
           ruido(){
               console.log(`Miau`);
       class perro extends Animal02{
           ruido(){
               console.log("Guau")
       let perro01 = new perro()
       perro01.ruido()
       let gato01 = new gato()
       gato01.ruido()
```











```
TS interfaz_01.ts X
typescript > TS interfaz_01.ts > ◆○ Persona
       interface Persona{
           // Atributos
           nombre:string
           apellido: string
       let nueva_persona = {
           nombre:'Jose',apellido:'Munoz'
       function caminar (persona:Persona):void {
           console.log(`la persona ${persona.nombre} ${persona.apellido} esta caminando`)
 12
       caminar(nueva_persona)
```









```
TS interfaz_02.ts X
typescript > TS interfaz_02.ts > ♥ Figura
  1 interface Figura{
          figura:string
          valor 01:number
          valor_02?:number
      function MostrarAreaFigura(areafigura:Figura):number{
          let area:number
          if(areafigura.valor_02){
              area = areafigura.valor 01 * areafigura.valor 02
              return area
              if(areafigura.figura == "circulo"){
                  area = 3.14 * areafigura.valor_01 * areafigura.valor_01
                  return area
                  area = areafigura.valor_01 * areafigura.valor_01
                  return area
     let figura_01 = {figura:"circulo",valor_01:10}
     let figura_02 = {figura:"cuadrado", valor_01:10}
     let figura_03 = {figura:"rectangulo",valor_01:10,valor_02:5}
     let area:number
     area = MostrarAreaFigura(figura 01)
     console.log(`la figura ${figura_01.figura} tiene un area de ${area}`)
      area = MostrarAreaFigura(figura_02)
     console.log(`la figura ${figura_02.figura} tiene un area de ${area}`)
     area = MostrarAreaFigura(figura_03)
     console.log(`la figura ${figura_03.figura} tiene un area de ${area}`)
```



