	UNIVERSIDAD AUTÓNOMA "TOMÁS FRÍAS" **CARRERA DE INGENIERÍA DE SISTEMAS**
	ESTUDIANTE:
MATERIA: Seminario de Sistemas SIS-719	
DOCENTE: Ing. Ditmar Castro	**PRACTICA Nº:** 2
AUXILIAR: Univ.	**GRUPO:** 1

ENTENDIMIENTO DE CLASES Y HERENCIA EN TYPESCRIPT

Instrucciones:

- Crear un repositorio en GitHub con su nombre completo en camelCase y al final agregue “P2Aux”
- Presentar el link a su repositorio en formato PDF con la cabecera de este documento al inicio de su tarea, llenando su nombre completo y al lado su CI
- Si uno de los incisos es una pregunta responderla justo debajo de la misma.

Desarrollo de la practica:

Introducción a Typescript

- Cree una variable “name” y dele como valor su nombre
- Luego intente cambiar el valor de “name” por un numero
- ¿Por qué TypeScript no le permite realizar la acción anterior?
- Cree una función llamada “greet” para poder saludar a una persona y además mostrarle su edad, es decir, la salida de la función debe ser algo parecido a “hello Josie, your age is 25”,
- ¿Qué tipo de retorno le asigno TypeScript a esta función?
- Cree una función donde el tipo de retorno “never” sea útil
- Cambie el tipo de retorno de la función “greet” a “never”
- ¿La acción anterior le da algún error? ¿Sí? ¿No? ¿Por qué?
- Modifique la función “greet” para poder implementar una función como parámetro la cual se encargará de imprimir o generar el saludo, establezca los tipos de datos necesarios para que la función cumpla con los requerimientos de TypeScript
- Haga una llamada a la función para comprobar sus resultados
- Modifique una vez más la función de “greet” para poder imprimir su año de nacimiento, pero agregue un parámetro para poder imprimirlo como numero o como cadena, de manera que dependiendo que argumento le pasen, lo imprima como uno de estos dos tipos de datos
- Es necesario que TODOS los parámetros o variables estén tipados apropiadamente (no valen tipos como “any”, “unknow”, etc)

Clases e Interfaces

- Cree una clase llamada “Department”
- Agregue las siguientes propiedades a la clase:
    - owner: su nombre completo
    - id: un id para identificar el departamento
    - workers: una lista de los trabajadores
    - createWorker: un método para agregar empleados al departamento
    - showEmployeesInfo: un método para mostrar la información de los empleados (la cantidad y los nombres)

- Cree una instancia de la clase “Department” tomando como argumento su nombre y un id para identificar el Departamento (programe el constructor como desee para este fin) y luego agregue 3 empleados
- Muestre la información de los empleados haciendo uso del metodo
- Cree otra clase llamada “CEODeparment”
- Herede las propiedades de la clase “Deparment”
- Agregue las siguientes propiedades:
    - admins: una lista con los roles disponibles. Como ser: “AUTHOR”, “ADMIN”, etc
- Cree una instancia de “CEODepartment” y agregue tres roles tomados como uno de los argumentos
- Muestre la propiedad “admins” de la instancia creada
- Cree otra clase llamada “ReportsDepartment” y herede la clase “Department”
- Agregue las siguientes propiedades:
    - reports: una lista que contendrá reportes. Por ejemplo: “task 004 failed”, etc
    - addReport: un método para agregar nuevos reportes
- Cree una instancia de la clase “ReportsDepartment”
- Ejecute el método “addReport” para agregar tres reportes
- Agregue un reporte sin usar el método addReport
- Impida que se agreguen mas reportes salvo utilizando el método creado (use modificadores)
- Use el modificador “private” en la propiedad name de la clase “Deparment”
- Agregue un método “greet” en la instancia de la clase “ReportsDepartment” para mostrar un mensaje de la siguiente manera: “Hello ${name} there are ${reports.length} reports” donde “name” y “reports” son las propiedades correspondientes.
- Si es que los tiene arregle los errores que le aparezcan usando modificadores
- Cree una clase llamada “Person”
- Agregue las siguientes propiedades a la clase:
    - name: su nombre completo
    - age: su edad
    - greet: un método para saludar
- Cree una interfaz para la clase “Person” y relacionela
- Divida las propiedades de la interfaz creada en dos interfaces diferentes (Data y Greeting), en” Data” estarán: “name” y “age” y en “Greeting” estarán solamente el método “greet”
- Herede las propiedades de “Data” a la “Greeting”
- Agregue una propiedad mas a “Data” con el nombre “lastname” y modifíquela de tal manera que sea opcional al momento de implementarla a una clase