UNIVERSIDAD NACIONAL DE INGENIERÍA

Área de Conocimiento de Tecnología de la Información y Comunicación

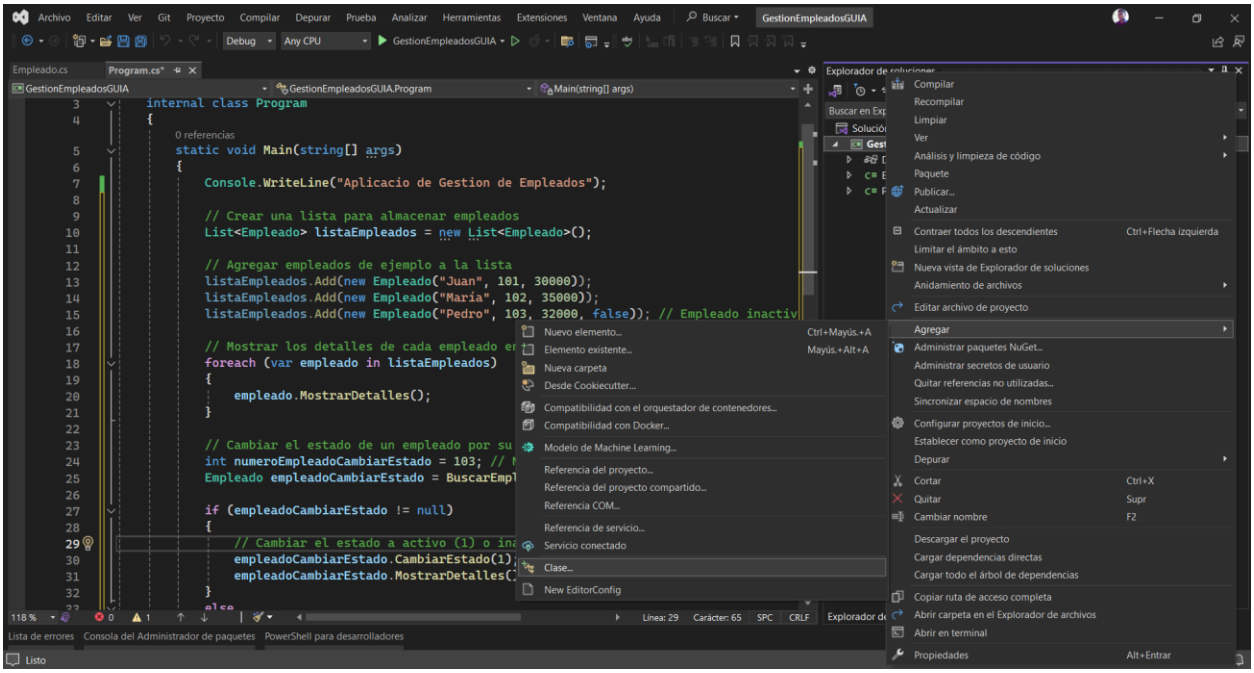
Paralelo Programación 1

Docente: Ing. Cristopher Larios

Guía didáctica #3: Clases Y objetos

I. Procedimiento

a. Creación de una clase



b. Declaración de Variables Miembros

```
{
    9 referencias
    internal class Empleado
    {

        // Variables miembro de la clase Empleado
        private string nombre;
        private int numeroEmpleado;
        private decimal salario;
        private bool activo; // Estado del empleado (activo o inactivo)

```

c. Método para Mostrar los detalles del empleado

```
// Método para mostrar los detalles del empleado
2 referencias
public void MostrarDetalles()
{
    string estado = (activo) ? "Activo" : "Inactivo";
    Console.WriteLine($"Nombre: {nombre}, Número de Empleado: {numeroEmpleado}, Salario: {salario:C}, Estado: {estado}");
}
```

d. Método para obtener el número de empleados

```
// Método para obtener el número de empleado
1 referencia
public int ObtenerNumeroEmpleado()
{
    return this.numeroEmpleado;
}
```

e. Método para cambiar el estado del empleado

```
// Método para cambiar el estado del empleado
1 referencia
public void CambiarEstado(int nuevoEstado)
{
    if (nuevoEstado == 1)
    {
        activo = true; // Activar empleado
        Console.WriteLine("Empleado activado.");
    }
    else if (nuevoEstado == 0)
    {
        activo = false; // Desactivar empleado
        Console.WriteLine("Empleado desactivado.");
    }
    else
    {
        Console.WriteLine("Estado no válido.");
    }
}
```

f. Crear una lista para almacenar empleados

```
Console.WriteLine("Aplicacion de Gestion de Empleados");

// Crear una lista para almacenar empleados
List<Empleado> listaEmpleados = new List<Empleado>();

// Agregar empleados de ejemplo a la lista
listaEmpleados.Add(new Empleado("Juan", 101, 30000));
listaEmpleados.Add(new Empleado("María", 102, 35000));
listaEmpleados.Add(new Empleado("Pedro", 103, 32000, false)); // Empleado inactivo
```

g. Mostrar los detalles de cada empleado en la lista

```
// Mostrar los detalles de cada empleado en la lista
foreach (var empleado in listaEmpleados)
{
    empleado.MostrarDetalles();
}
```

h. Cambiar el estado de un empleado por su número de empleado

```
// Cambiar el estado de un empleado por su número de empleado
int numeroEmpleadoCambiarEstado = 103; // Número de empleado a cambiar estado
Empleado empleadoCambiarEstado = BuscarEmpleado(listaEmpleados, numeroEmpleadoCambiarEstado);

if (empleadoCambiarEstado != null)
{
    // Cambiar el estado a activo (1) o inactivo (0)
    empleadoCambiarEstado.CambiarEstado(1); // Cambiar a activo
    empleadoCambiarEstado.MostrarDetalles(); // Mostrar detalles actualizados
}
else
{
    Console.WriteLine($"Empleado con número {numeroEmpleadoCambiarEstado} no encontrado.");
}
```

i. Método para buscar un empleado por su número de empleado

```
// Método para buscar un empleado por su número de empleado
1 referencia
static Empleado BuscarEmpleado(List<Empleado> empleados, int numeroEmpleado)
{
    foreach (var empleado in empleados)
    {
        if (empleado.ObtenerNumeroEmpleado() == numeroEmpleado)
        {
            return empleado;
        }
    }
    return null; // Retorna null si no se encuentra ningún empleado con ese número

}
```

II. Consideraciones

- La clase Empleado define un modelo para representar un empleado con propiedades como nombre, numeroEmpleado, salario y activo (estado del empleado).
- El constructor Empleado inicializa los valores del empleado incluyendo su estado (activo).
- El método MostrarDetalles() muestra los detalles del empleado incluyendo su estado (activo o inactivo).
- ObtenerNumeroEmpleado() devuelve el número de empleado.
- CambiarEstado(int nuevoEstado) permite cambiar el estado del empleado a activo (1) o inactivo (0).
- En el método Main, se crea una lista listaEmpleados para almacenar objetos Empleado.
- Se agregan empleados de ejemplo a la lista utilizando el constructor de Empleado.
- Se muestra la información de cada empleado utilizando el método MostrarDetalles().

- Se busca un empleado por su número de empleado utilizando el método BuscarEmpleado() y se cambia su estado utilizando CambiarEstado().

III. Resultados

- Elaborar un documento, con una portada con los datos del estudiante.
- Replicar los pasos abordado en esta guía, y documentarlos con capturas de pantalla, explicando sus pasos.
- Llevar un seguimiento del cambio con el control de versiones GIT, deberá documentar en una sección del documento con capturas de pantalla y explicación del seguimiento con GIT.
- Crear un repositorio en GITHUB y subir los cambios, deberá documentar el proceso con capturas de pantalla en el documento.
- Agregar funcionalidad de aumentar un porcentaje de salario X, a un empleado considerando su número de empleado.
- Esta nueva funcionalidad deberá anexarse como un nuevo commit en el seguimiento del repositorio.
- Entregar via Teams, documento PDF y URL del repositorio.