INTRODUCCIÓN AL PARADIGMA ORIENTADO A OBJETOS

PROFESOR: CARLOS SPERANZA

REPASANDO: LA PROGRAMACIÓN ESTRUCTURADA

- En la programación estructurada aparece la tipificación de datos.
- Tenemos acá un flujo de ejecución “secuencial” que se ve afectado por ciertas estructuras de control y saltos a funciones (o subrutinas).
- Podemos encapsular ciertos procedimientos en funciones.
- Por otro lado, la información podemos agruparla en estructuras de datos.

LIMITACIONES DE LA PROGRAMACIÓN ESTRUCTURADA

- A medida que los programas y sistemas crecen la programación estructurada no puede acompañar este crecimiento.
- Se busca pasar a un paradigma (o forma de pensar una solución) en una forma que represente mejor la realidad.
- Al observar la realidad se ve como modelar el entorno del mundo real en un sistema.
  - Aparece el concepto de dominio, límites, sistema y **objetos**.

POO - CONCEPTOS INICIALES

- Sistema: Conjunto de elementos o entidades que trabajan relacionándose entre si para cumplir con un objetivo común.
  - Los sistemas tendrán un límite (define que entra o no en el sistema) y un dominio (donde trabaja el sistema).
  - Podemos pensar a los objetos como estas entidades que se interrelacionan.
- Dominio: Es una parte de la realidad en la cual impactará el sistema que se desarrollará y a su vez este modificará.
  - El alcance de un sistema se determina a través de los limites.
  - Por ejemplo: Si desarrollamos el sistema de inscripciones a materias para la FRH de la UTN nos interesa una parte del dominio de la universidad.

CONCEPTO DE UN OBJETO EN UN SISTEMA

- Es una entidad conceptual que representa un elemento que tiene lugar en el dominio del sistema.
- Un objeto puede ser una entidad física, abstracta o transaccional.
- Un objeto puede presentar (o saber) ciertos datos, tendrá ciertos comportamientos y se relacionará con otras entidades del sistema.
- Podemos encontrar a los objetos en las etapas de análisis, diseño e implementación de un sistema.

EJEMPLO

- Para el dominio que hablamos antes, (inscripciones a materias), Qué tipo de entidades serían los alumnos, las asignaturas y las asistencias?

EJEMPLO

- Para el dominio que hablamos antes, (inscripciones a materias), Qué tipo de entidades serían los alumnos, las asignaturas y las asistencias?

Físicas

Abstractas

Transaccionales

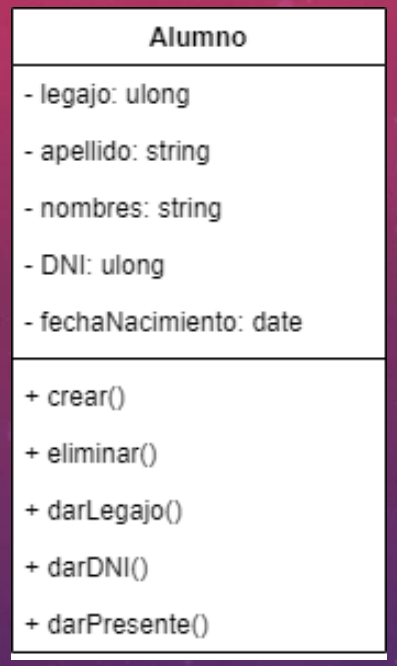
ETAPAS DEL DESARROLLO DE UN SISTEMA (GENERALES)

1. Análisis: Se evalúan las necesidades del cliente y se busca entender el dominio del problema. Etapa de comprensión. Al comenzar el análisis podemos identificar los objetos.
2. Diseño: Se proyecta una posible solución para el problema. Se identifican las clases de los objetos identificados que se deben modelar, qué atributos nos importan y qué funcionalidades deben tener.
3. Implementación: Se construye el sistema. Se desarrollan las clases diseñadas y se realizan las funcionalidades deseadas.

CONCEPTO DE CLASE

- Una clase es la modelización del caso general de los objetos.
- Podemos pensarla como un molde o prototipo de los objetos que encontramos en el dominio del sistema.
- En una clase podemos establecer características (datos + comportamientos) generales que nos interesan de los objetos.
- Todos los objetos (instancias) de una clase tendrán las mismas características.

LA CLASE ALUMNO



Vista de diseño (UML)

```
namespace modeloUTN
{
    class Alumno
    {
        ulong legajo;
        string apellido;
        string nombres;
        ulong DNI;
        DateTime fechaNacimiento;

        public Alumno(ulong legajo, string apellido, string nombres, ulong dNI, DateTime fechaNacimiento)
        {
            this.legajo = legajo;
            this.apellido = apellido;
            this.nombres = nombres;
            DNI = dNI;
            this.fechaNacimiento = fechaNacimiento;
        }

        public string darDNI()
        {
            return DNI.ToString();
        }
    }
}
```

Vista de implementación (C#.NET)

INSTANCIAS DE UNA CLASE

- Una clase tendrá varios casos particulares (objetos) o instancias de la misma.
- Representan el caso concreto y puntual construido en base al molde de la clase
- Tanto la clase como las instancias las podemos encontrar en la etapa de implementación mientras que la clase sin instancias aparece en la etapa de diseño

OPERANDO CON INSTANCIAS DE UNA CLASE

```
Alumno auxAlumno; //creamos la instancia o variable de referencia (puntero) del tipo Alumno
auxAlumno = new Alumno(1234, "Jackson", "Tito", 40258558, new DateTime(1997, 03, 05)); //creamos la instancia invocando a un CONSTRUCTOR
//Una vez que se crea se le asigna la instancia creada a la referencia
```

```
Alumno aux2Alumno = new Alumno(3333, "Jose", "Pekerman",20379555, new DateTime(1974, 20, 3));//podemos hacer todo dentro de una misma linea
```

EL CONSTRUCTOR DE UNA CLASE

- Es una función propia de una clase que le permite inicializar un objeto de una clase.
- Puede recibir por parámetros los valores iniciales del objeto.
- Asigna luego internamente el lugar en memoria (similar al malloc)
- Son métodos sobrecargados
  - Pueden tener distinto funcionamiento según los parámetros que recibe

CONSTRUCTORES DE UNA CLASE

```
public Alumno(ulong legajo, string apellido, string nombres, ulong dNI, DateTime fechaNacimiento)
{
    this.legajo = legajo;
    this.apellido = apellido;
    this.nombres = nombres;
    DNI = dNI;
    this.fechaNacimiento = fechaNacimiento;
}

public Alumno(ulong legajo, string apellido, ulong dNI)
{
    this.legajo = legajo;
    this.apellido = apellido;
    DNI = dNI;
}

public Alumno()
{
    this.legajo = 0;
    this.apellido = "vacio";
    this.nombres = "vacio";
    this.DNI = 0;
}
```

GETTERS Y SETTERS

- Desde otra clase externa NO PODEMOS acceder o modificar los atributos privados de un objeto directamente.
  - Viola el principio de ocultamiento de las clases.
- En su lugar tenemos métodos llamados getters para obtener atributos de una clase o setters para establecer un valor a un atributo

```
static void Main(string[] args)
{
    Alumno auxAlumno = new Alumno();
    auxAlumno.setDNI(402585558); //llamamos a un setter para setear el DNI
    auxAlumno.setNombres("Josecito"); //llamamos a un setter

    Console.WriteLine(auxAlumno.getNombres()); //llamamos a los getters para acceder a los atributos
    Console.WriteLine(auxAlumno.getDNI().ToString());


}
```

GETTERS Y SETTERS

```
public ulong getDNI()
{
    return this.DNI;
}

public void setDNI(ulong dni)
{
    this.DNI = dni;
}

public string getNombres()
{
    return nombres;
}

public void setNombres(string nombres)
{
    this.nombres = nombres;
}
```

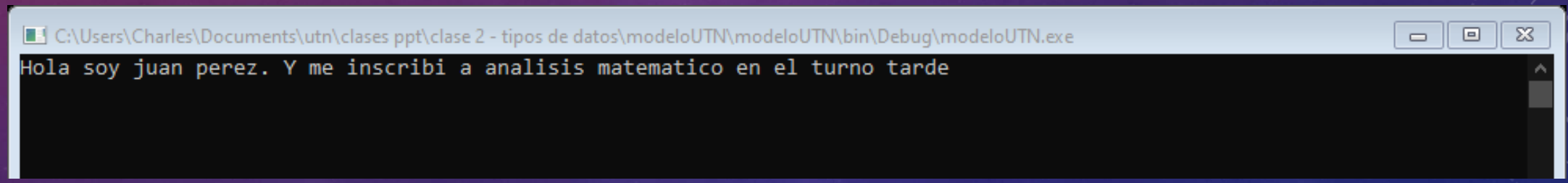
INVOCAR A UNA FUNCIONALIDAD DE UNA CLASE

- Los comportamientos funcionales o métodos le dan sentido a una clase.
- Una clase sin métodos no tiene razón de ser.
- Los métodos ***encapsulan*** una responsabilidad asignada a una clase.
- Este método se ejecuta cuando llegue el mensaje de invocación correspondiente

```
public Alumno(ulong legajo, string apellido, string nombres, ulong dNI, DateTime fechaNacimiento)
{
    this.legajo = legajo;
    this.apellido = apellido;
    this.nombres = nombres;
    DNI = dNI;
    this.fechaNacimiento = fechaNacimiento;
}

public void inscribirMateria(string materia, string turno)
{
    Console.WriteLine("Hola soy" + nombres + " " + apellido + ". Y me inscribi a " + materia + "en el turno " + turno);
}
```

INVOCANDO MÉTODOS

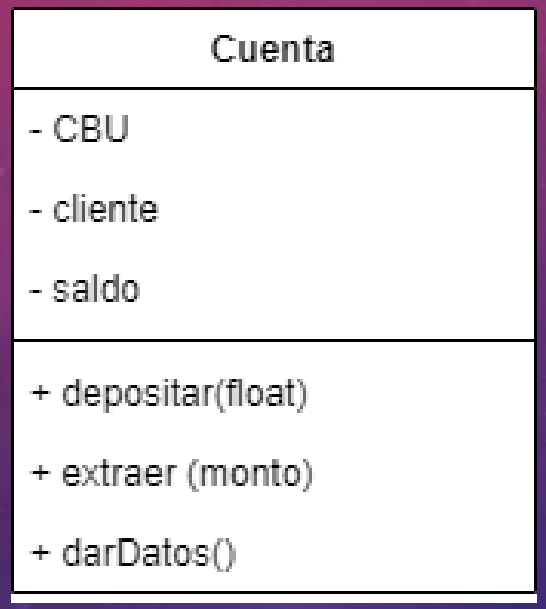
```
class Program
{
    static void Main(string[] args)
    {
        Alumno auxAlumno = new Alumno(1234, "perez", "juan", 233333, new DateTime(2020, 12, 1)); //método constructor
        auxAlumno.inscribirMateria("analisis matematico","tarde"); //llamamos a otro método

    }
}
```



DESTRUCCIÓN DE UN OBJETO

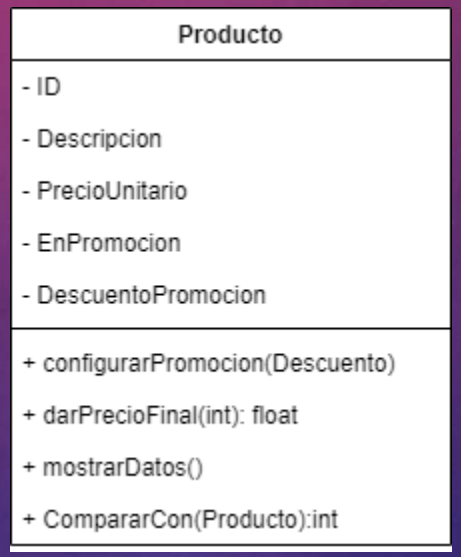
- En .NET framework la liberación del espacio de memoria de un objeto se produce automáticamente con el garbage collector.
- Puede detectar cuando un objeto se deja de usar para destruirlo.
- En consecuencia, acá no tenemos métodos destructores (como sí por ejemplo en C++).
- Tenemos métodos finalizadores que pueden determinar un comportamiento antes de destruir una instancia pero no sabremos cuando sucederá

IMPLEMENTANDO UNA CLASE ELEMENTAL



IMPLEMENTAMOS OTRA?

- Vamos a crear una clase con los siguientes elementos



- En una clase principal pedir un número indeterminado de Productos (finalizando con el ID 0) e informar el de mayor precio

FORMULARIO CLASE 3

https://bit.ly/formularioClase3