TEC GURUS

www.tecgurus.net

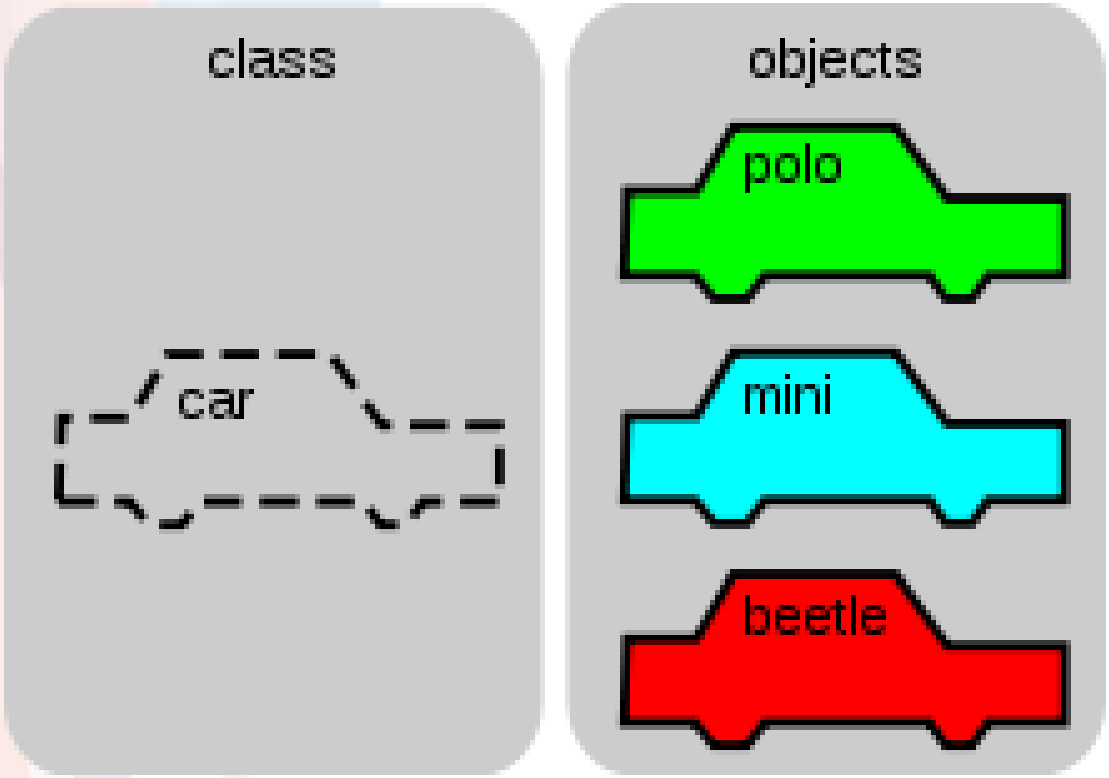
Repaso

1. Diferencias entre clase y objeto
2. Servicios REST y SOAP
3. Visual Studio
4. Lenguajes que soporta Visual Studio
5. Templates en Visual Studio
6. IDE
7. Solution Explorer
8. Team Explorer
9. Server Explorer
10. NuGet

Ing. Jeniffer Ramirez

Índice

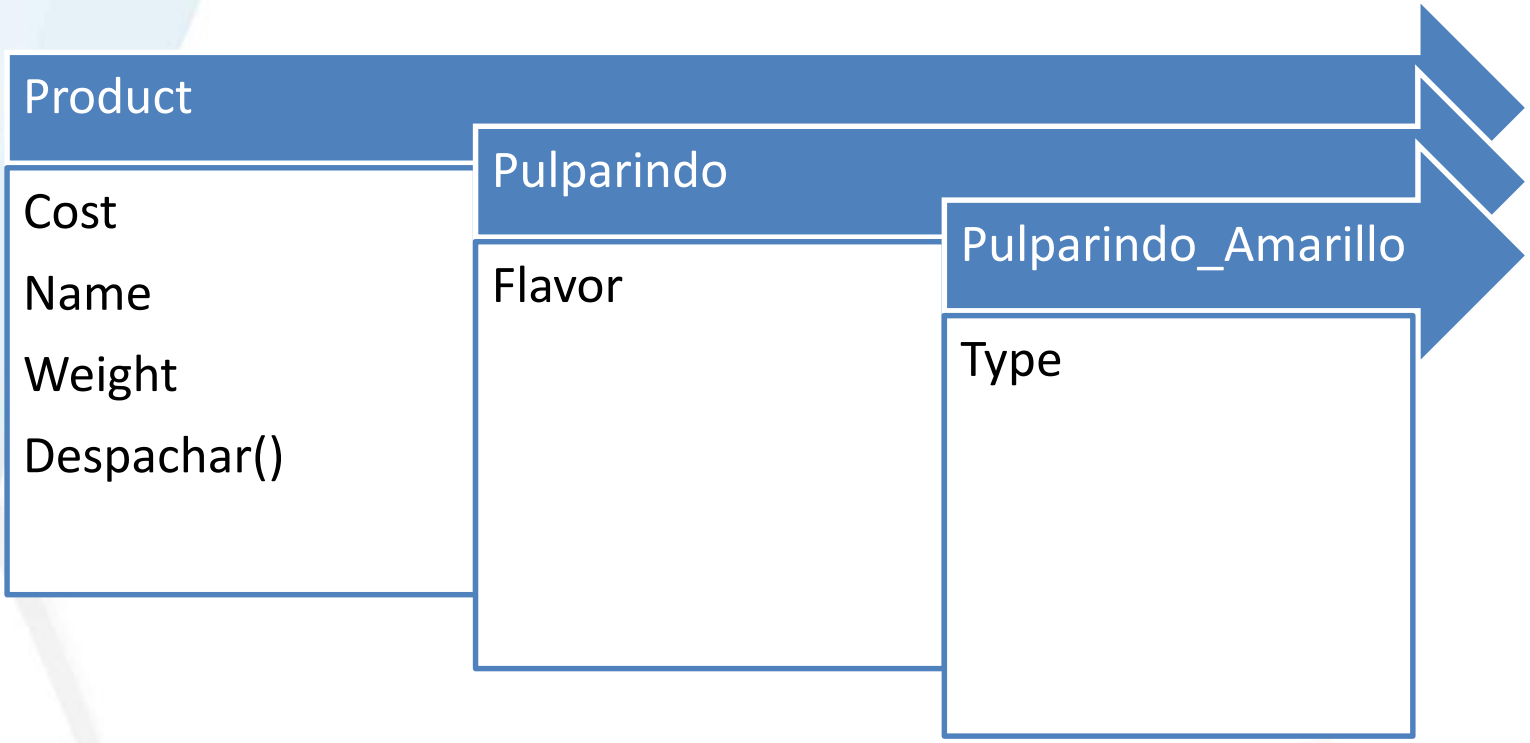
POLIMORFISMO



- Por herencia
- Por abstracción
- Por interface

Ing. Jeniffer Ramirez

HERENCIA



Ing. Jeniffer Ramirez

Método Despachar()

```
public string Despachar()
{
    return "Despachando";
}
```

Este método siempre se va a heredar, pero siempre va a regresar el mismo string “Despachando”.

```
public virtual string Despachar()
{
    return "Despachando";
}
```

Para poder reescribir este método al ser heredado, es necesario marcarlo como virtual.

Ing. Jeniffer Ramirez

Ejercicios

1. Eliminar clase base
2. Editar clase Product como marcada en el diagrama
3. Crear clase Pulparindo, heredando de Product
4. Sobreescribir método Despachar para que regrese cadena “Despachando Pulparindo” usando la palabra exclusiva *override.*
5. Instanciar el objeto Pulparindo en Program.cs e imprimir método despachar al ejecutar el método *Main*.

Ing. Jeniffer Ramirez

ABSTRACCIÓN

CLASE ABSTRACTA

```
public abstract class ProductAbstract
{

}
```

- Se define utilizando la palabra exclusiva ***abstract*** después del nivel de acceso definido para la clase.
- Todos sus métodos deben ir definidos con la palabra exclusiva ***abstract***.

- ❑ Se da por el uso de clases abstractas
- ❑ Contiene comportamiento definido pero NO IMPLEMENTADO

Ejercicios

1. Crear clase abstracta AbstractProduct
2. Definir método Despachar en la clase AbstractProduct
3. Crear clase Mazapan, heredando de AbstractProduct
4. Implementar método Despachar para que regrese cadena “Despachando Mazapan” usando la palabra exclusiva *override.*
5. Instanciar el objeto Mazapan en Program.cs e imprimir método despachar al ejecutar el método *Main*.

Ing. Jeniffer Ramirez

INTERFACE

Declaración de una interface:

```
public interface IProduct
{
    |
}
```

- Se define utilizando la palabra exclusiva ***interface*** después del nivel de acceso definido para la interface.
- Todos sus métodos deben ir definidos.

- ❑ De los polimorfismos más importantes
- ❑ Permite definir qué se debe hacer
- ❑ Permite definir que NO se debe hacer
- ❑ Permite definir cómo hacerlo
- ❑ Oculta el comportamiento o la implementación

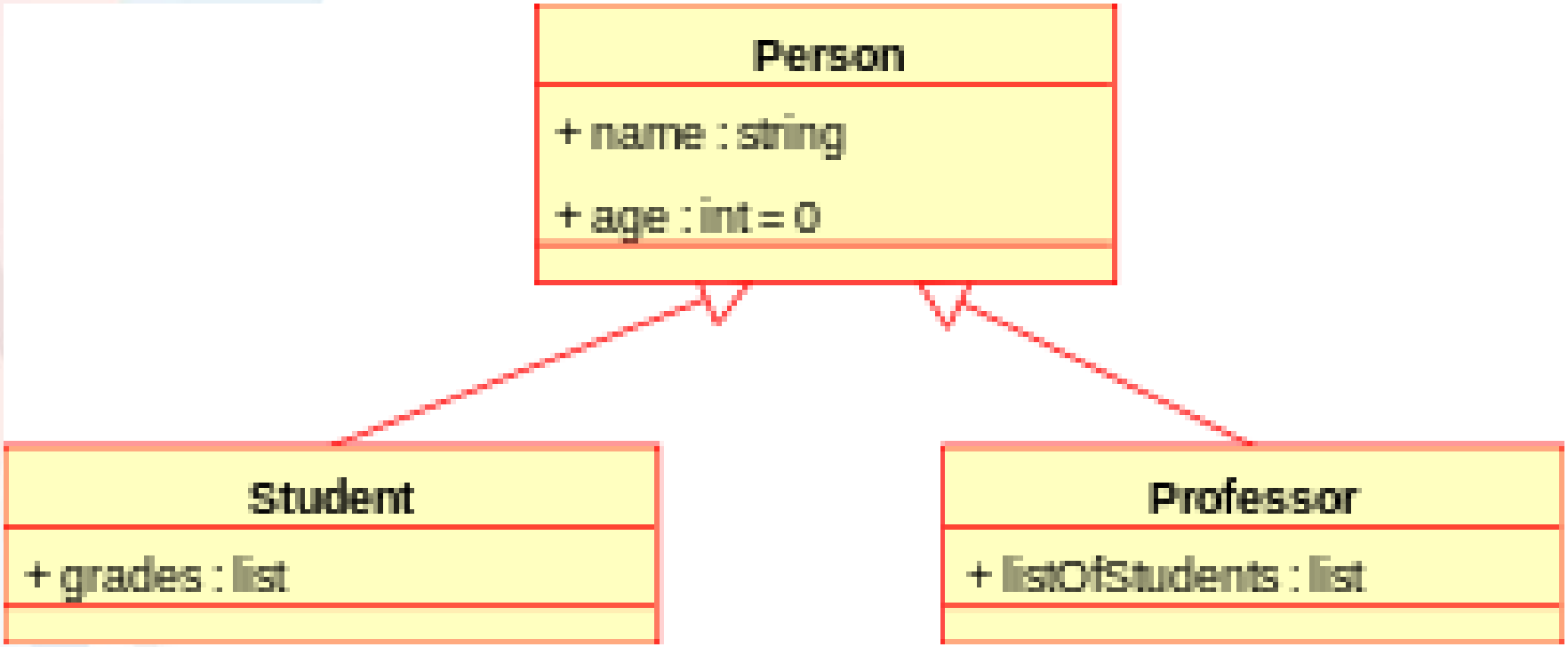
Ing. Jeniffer Ramirez

Ejercicios

1. Crear interface IProduct
2. Definir método Despachar en la interface
3. Crear clase Paleta, heredando de IProduct
4. Implementar método Despachar para que regrese cadena “Despachando Paleta” SIN usar la palabra *override.*
5. Instanciar el objeto Paleta en Program.cs e imprimir método despachar al ejecutar el método *Main*.

Ing. Jeniffer Ramirez

UML class diagram



Ing. Jeniffer Ramirez

Ejercicios

1. Entrar a https://www.gliffy.com/examples/uml-diagrams
2. Crear una cuenta (Google, Facebook u otro correo)
3. Realizar diagrama de clases UML del actual proyecto