

## **A2.Prácticas Formativas autogestionadas**

Sitio: Agencia de Habilidades para el Futuro  
Curso: Desarrollo de Sistemas Orientado a Objetos 1º D  
Libro: A2.Prácticas formativas autogestionadas

Imprimido por: Eduardo Moreno  
Día: martes, 27 de mayo de 2025, 01:38

# Tabla de contenidos

## **S12. Interfaces y Polimorfismo**

- Claves de autocorrección

## **S10. Polimorfismo**

- Claves de autocorrección

## **S8. Herencia y UML**

- Claves de autocorrección

## **S4. Abstraer y encapsular**

- Claves de autocorrección

## **S4. Interface de usuario**

- Claves de autocorrección

## **S3. Manipular clases**

- Claves de autocorrección

## **S2: ¡A jugar con superhéroes!**

- Claves de autocorrección

## **S2: Primera solución en el entorno visual**

- Claves de autocorrección

## **S2: Ingresando datos**

- Claves de autocorrección

## **S1: Escribir en C#**

- Claves de autocorrección



## Práctica Formativa: Interfaces y Polimorfismo

- Tipo de actividad: Práctica Formativa autogestionada con autocorrección.
- Objetivo: Realizar una práctica guiada de Interfaces y Polimorfismo

### Enunciado:

Estos ejercicios te permitirán practicar la creación de interfaces, implementación de métodos, herencia de interfaces, uso de polimorfismo y trabajar con objetos que implementan interfaces.

¡Espero que te sean útiles para fortalecer tus habilidades en el uso de interfaces en C#!

- Ejercicio 1: Crear una interfaz

Crea una interfaz llamada `IAreaCalculable` que defina el método `calcularArea()`.

- Ejercicio 2: Implementar una interfaz

Crea una clase llamada `Cuadrado` que implemente la interfaz `IAreaCalculable`. Implementa el método `calcularArea()` para calcular el área del cuadrado.

- Ejercicio 3: Herencia de Interfaces

Crea otra interfaz llamada `IInformacionFigura` con un método `obtenerInformacion()`.

- Ejercicio 4: Herencia de Interfaces en una clase

Crea una clase llamada `CuadradoInformacion` que herede de la clase `Cuadrado` y también implemente la interfaz `IInformacionFigura`. Implementa el método `obtenerInformacion()` para mostrar información sobre el cuadrado (lado y área).

- Ejercicio 5: Utilizando Polimorfismo

Crea una lista de tipo `IAreaCalculable` y agrega objetos de las clases `Cuadrado` y `CuadradoInformacion` a la lista. Luego, itera sobre la lista y muestra el área de cada cuadrado usando el método `calcularArea()`. Utiliza polimorfismo para mostrar cómo los objetos se comportan según la interfaz implementada.

- Ejercicio 6: Extender la funcionalidad

Agrega una nueva clase llamada `Circulo` que implemente `IAreaCalculable`. Implementa los métodos necesarios para calcular el área del círculo. Luego, extiende la clase `Circulo` para que también implemente `IInformacionFigura` y muestre información sobre el círculo.


- Ejercicio 7: Utilizar interfaces en una función

Crea una función llamada `ImprimirInformacionFigura` que tome como parámetro un objeto que implemente la interfaz `IInformacionFigura`. Luego, dentro de la función, llama al método `obtenerInformacion()` del objeto y muestra la información por pantalla.



### Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "Interfaces y Polimorfismo" para que puedas cotejar con tu producción.

img-template-01

Descargá los documentos de la carpeta [Claves de autocorrección](#), para acceder al modelo de resolución de la práctica formativa de esta semana.



- Tipo de actividad: Práctica Formativa autogestionada con autocorrección.
- **Objetivo:** Aplicar polimorfismo al problema propuesto

**Enunciado:**

Eres el desarrollador de software de una compañía de envíos llamada "FastShip", la cual se encarga de realizar entregas rápidas y seguras de paquetes a nivel nacional e internacional. La compañía ha experimentado un aumento en la demanda y necesita una nueva solución de software para gestionar los diferentes tipos de paquetes que manejan.

Desarrolla un sistema de software para "FastShip" que permita manejar los diferentes tipos de paquetes utilizados en la empresa. Los paquetes pueden ser de tres tipos:

- paquetes regulares,
- paquetes frágiles y
- paquetes refrigerados.

Cada uno de los paquete tiene atributos como ser:

- su peso en kg (double),
- sus dimensiones en m2 (double)
- su dirección de destino (string)

Además, los paquetes refrigerados tienen el atributo **temperatura** (double) en la que debe ser transportado.

Nos informan algunos detalles:

- el paquete regular tiene un costo fijo de \$10 por cada kg transportado
- el paquete frágil se cobra un recargo del 100% del paquete regular
- el paquete refrigerado si la temperatura es menor o igual a 10 grados tiene un incremento del 200% con respecto al paquete regular, en cambio, si es mayor a 10 grados tiene un incremento del 50% con respecto al paquete regular.

**Se pide:**

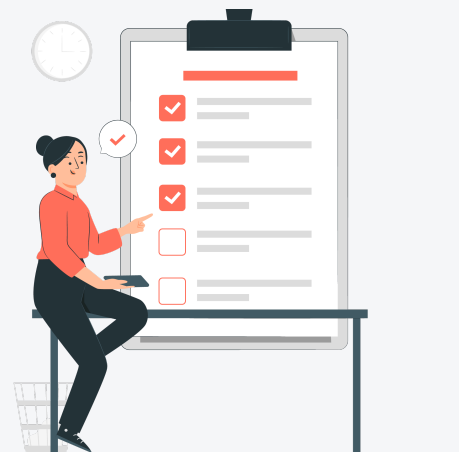
- Diagrama UML
- realizar el método `calcularCostoDeEnvio()` el cual calcula el costo de envío de cada tipo de paquete, teniendo en cuenta tarifas específicas para los paquetes frágiles y refrigerados.
- realizar el método `mostrarInformaciónDetalladaDeCadaPaquete()` que muestra por pantalla la información de cada uno de los paquetes.

Utiliza el concepto de polimorfismo para implementar las operaciones comunes de cálculo de costo de envío y mostrar información detallada de los paquetes. De esta manera, la compañía podrá ofrecer servicios más personalizados y eficientes para satisfacer las necesidades de sus clientes.



## Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "Polimorfismo" para que puedas cotejar con tu producción.



Descargá los documentos de la carpeta [Compañía de envíos llamada FastShip](#), para acceder a las claves de autocorrección.



## En un sistema de gestión de productos

- Tipo de actividad: Práctica formativa autogestionada con autocorrección
- Objetivo: Crear un diagrama UML

### Leé con atención el enunciado

Trabajamos en una tienda en línea y nos han pedido desarrollar un sistema de gestión de productos.

Nuestro objetivo es administrar diferentes tipos de productos y calcular el precio total de venta de cada uno de ellos.

Los productos en nuestro sistema tienen tres atributos comunes: nombre, precio y tipo.

Sin embargo, existen dos tipos específicos de productos: **los perecederos y los no perecederos**.

Los productos **perecederos** tienen un atributo adicional llamado "**días a caducar**", que indica cuántos días les quedan antes de que caduquen. Por otro lado, los productos **no perecederos** tienen un atributo llamado "**tipo**" que especifica la categoría a la que pertenecen.

### ¿Qué te proponemos realizar?

- Crear el diagrama UML acorde al enunciado.
- Codificar las clases detectadas.

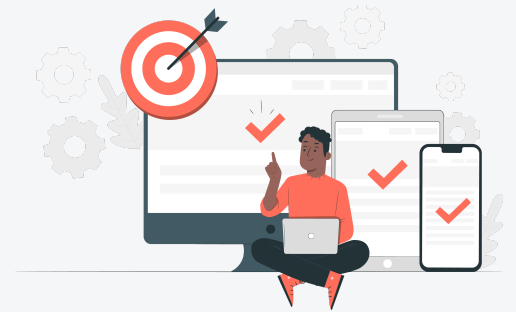
En la próxima semana le daremos funcionalidad al sistema. ¡Estemos preparados con nuestra base!



## **Claves de autocorrección**

Te dejamos una posible resolución de la práctica formativa "Herencia y UML" para que puedas cotejar con tu producción.

Descargá la carpeta de archivos [en este enlace](#) para acceder a las claves de autocorrección.







## Administrar información de empleados en una empresa

- **Tipo de actividad:** Práctica formativa autogestionada con **claves de autocorrección**.
- **Objetivo:** Aplicar los principios de abstracción, encapsulamiento y asociación simple.

### Descripción:

En el contexto de un sistema de gestión de empleados, se te ha encomendado diseñar y desarrollar una aplicación que administre la información de los empleados de una empresa.

### Para hacer:

1. Crea una clase llamada **Empleado** que represente a un empleado de la empresa. Esta clase debe tener las siguientes propiedades:
  - Nombre: el nombre del empleado (debe ser una cadena de caracteres).
  - Edad: la edad del empleado (debe ser un número entero).
  - Cargo: el cargo del empleado en la empresa (debe ser una cadena de caracteres).
2. Crea una clase llamada **Empresa** que represente a la empresa en su conjunto. Esta clase debe tener las siguientes propiedades:
  - Nombre: el nombre de la empresa (debe ser una cadena de caracteres).
  - Empleado: un objeto de la clase "Empleado" que representa al empleado principal de la empresa.
3. Implementa el encapsulamiento adecuado en ambas clases para garantizar el acceso controlado a las propiedades.
4. Crea un **método** en la clase **Empleado** llamado "**MostrarInformacion**" que imprima por consola el nombre, la edad y el cargo del empleado.
5. Crea una **instancia** de la clase **Empleado** y una **instancia** de la clase **Empresa**. Asigna el empleado creado a la propiedad **Empleado** de la empresa.
6. Utiliza el **método** "**MostrarInformacion**" del objeto **Empleado** dentro del objeto **Empresa** para mostrar por consola la información del empleado.

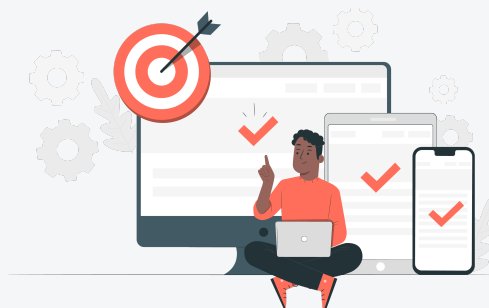
Recuerda aplicar los conceptos de **abstracción**, **encapsulamiento** y **asociación simple** en el diseño y desarrollo de tu solución. Presta atención a la forma en que las clases se relacionan entre sí y cómo se accede a las propiedades de cada objeto.



## **Claves de autocorrección**

Te dejamos una posible resolución de la práctica formativa "Abstraer y encapsular" para que puedas cotejar con tu producción.

Descargá la carpeta con los archivos [en este enlace](#), para acceder a las **claves de autocorrección**.





## Crear la Interfaces de usuario para el IFTS

- **Tipo de actividad:** Práctica formativa con claves de autocorrección.
- **Objetivo:** Crear interface de usuario con manejo de ventanas.

### Descripción:

Usando la Solución llamada **PrimerProyecto**, que venimos diseñando en las semanas anteriores, cuyo entorno hace referencia a:

“En un instituto terciario deben registrar el nombre, apellido y documento de los futuros postulantes para luego asignarles el curso y procesar la inscripción ...”

### ¿Qué te proponemos hacer?

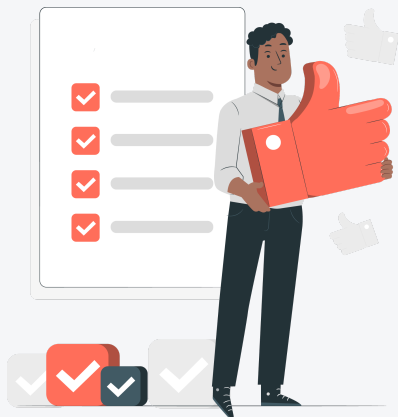
1. Agregar una ventana de Login con las siguientes características:
  - Debe tener en un PictureBox una imagen que haga referencia a un instituto terciario
  - TextBox que permita ingresar el nombre de usuario (txtUsuario)
  - TextBox que permita ingresar una contraseña camuflada (txtClave)
  - Button con la leyenda “Ingresar” (btnIngresar)
2. Leer el documento “ventana de inicio de una solución” para que el formulario que acabas de agregar sea el primero en la ejecución.
3. El **btnIngresar** debe validar el ingreso del usuario y contraseña, como no hay registro externo de datos vamos a considerar que la información válida es **Usuario = Administrador** y **contraseña = Admin1234**. Quiere decir que en el evento click del botón con el uso de un if se valida los datos.
  - **Respuesta = SI:** debe llamar de manera No Modal al formulario cuyo diseño es el formulario de la Solución PrimerProyecto
  - **Respuesta = NO** debe mostrar en un cuadro de diálogo un mensaje que muestre la leyenda usuario inexistente.
4. Al formulario de la Solución PrimerProyecto que tiene el siguiente diseño (recuerde que lo programaste en la semana 3)

5. Agregar un [DataGridView](#) con las columnas **Nombre**, **Apellido Tipo** y **Documento**. La carga de valores a las celdas las debe hacer usando la clase "**Postulante**" (creada en la semana 3)



## Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "Interface de usuario" para que puedas cotejar con tu producción.



Descargá la carpeta con los archivos [en este enlace](#) para acceder a las claves de autocorrección.



## Registro de postulantes en un IFTS

- **Tipo de actividad:** Práctica formativa con autocorrección
- **Objetivo:** Manipular clases en una Solución

**Descripción:** Usando la Solución llamada PrimerProyecto, diseñado en la semana la práctica formativa de la semana 2, cuyo entorno hace referencia a:

*"En un instituto terciario deben registrar el nombre, apellido y documento de los futuros postulantes para luego asignarles el curso y procesar la inscripción ..."*

Se pide resolver cada uno de los siguientes puntos

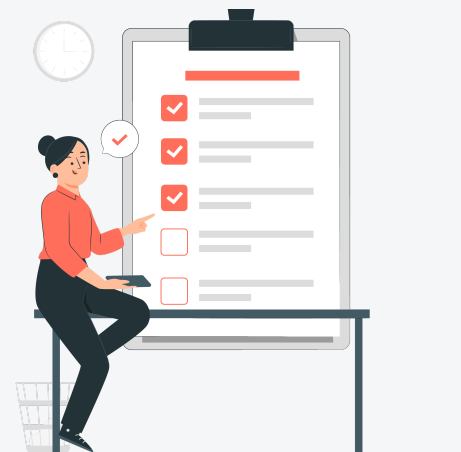
1. Agregar al diseño un control **comboBox** para indicar el tipo de documento del postulante. Los ítems deben ser DNI, Pasaporte, Extranjero.
1. Crear la clase **Postulante** cuyos atributos son los del siguiente diseño: Nombre, Apellido, Tipo, Documento.

3. Codificar en el botón **INGRESAR** la carga de datos en la clase. Para verificar si el ingreso es correcto mostrar en un cuadro de diálogo el contenido. Los contenidos de los controles **TextBox** y **ComboBox** se almacena en la propiedad text.



## Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "Manipular clases" para que puedas cotejar con tu producción.



Descargá la carpeta de archivos [en este enlace](#) para acceder a las claves de autocorrección.



## ¡A jugar con superhéroes!

- **Tipo de actividad:** Práctica formativa autogestionada con autocorrección
- **Objetivo:** Realizar un programa que te permita practicar la escritura de una clase y poder instanciar objetos de esa clase y hacerlos interactuar.

Leé con atención la siguiente situación.



En una galaxia lejana habitan muchos superhéroes con habilidades extraordinarias, estos han decidido realizar una competencia entre ellos comparando sus distintos atributos. Para ello, necesitamos crear la clase "SuperHeroe" que nos permita representar a estos poderosos personajes con los atributos nombre (String), fuerza (int), resistencia (int) y superpoderes (int).

Todos los atributos numéricos deberán aceptar valores entre 0 y 100; en caso que el setter correspondiente reciba un valor fuera de rango deberá setear el valor límite correspondiente (si recibe un valor negativo asignar cero, si recibe uno superior a cien, asignar cien).

El constructor de la clase recibirá todos los valores de sus atributos por parámetro y usará los setters (todos privados) para validar el rango correcto de los poderes del superhéroe.

Se deberá crear un método competir() el cual recibirá otro superhéroe como parámetro y, comparando los poderes de él mismo contra el otro recibido por parámetro, devolverá TRIUNFO, EMPATE o DERROTA, dependiendo del resultado.



Para triunfar un superhéroe debe superar al otro en al menos 2 de los 3 ítems.

### ¿Qué te proponemos realizar?

- Escribir la clase Test que contenga el método main() para probar el correcto funcionamiento de la clase previamente realizada con el siguiente ejemplo:

superHeroe1: Nombre: "Batman", Fuerza: 90, Resistencia: 70, Superpoderes: 0

superHeroe2: Nombre: "Superman", Fuerza: 95, Resistencia: 60, Superpoderes: 70

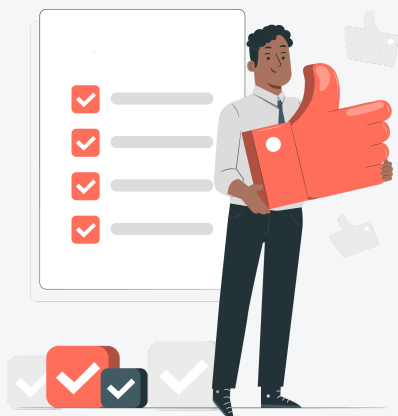
- Hacer jugar al superheroe1 pasándole el objeto superheroe2 y mostrar el resultado por pantalla.
- Chequear el resultado (debería ser DERROTA).
- Hacer jugar al superheroe2 contra el superheroe1 y mostrar el resultado por pantalla.
- Chequear el resultado (debería ser TRIUNFO).
- Crear más superhéroes con distintos valores y ... ¡¡A jugar!!





## Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "¡A jugar con superhéroes!" para que puedas cotejar con tu producción.



Descargá la carpeta de archivos [en este enlace](#) para acceder a las claves de autocorrección.



## Creación de la primera solución en el entorno visual

- **Tipo de actividad:** Práctica formativa autogestionada con claves de autocorrección.
- **Objetivo:** Manipular los controles básicos en un formulario.

**Descripción :** teniendo en cuenta la tabla que se detalla a continuación, construí la primera Solución en el entorno visual. El nombre de la Solución es PrimerProyecto

Sobre este diseño trabajaremos la próxima semana.

Nombre	-----	INGRESAR
Apellido	-----	LIMPIAR
Documento	-----	

Control	Prefijo	Propiedad	Valor
Formulario	frm	Name	frmPrincipal
		Text	PRIMER PROYECTO
		StartPosition	CenterScreen
		BackColor	192; 255; 192 (verde claro)
Label1	lbl	Name	lblNombre
		Text	Nombre
Label2	lbl	Name	lblApellido
		Text	Apellido
Label3	lbl	Name	lblDocumento
		Text	Documento
TextBox1	txt	Name.	txtNombre
		text	vacío
TextBox2	txt	Name.	txtApellido
		text	vacío
TextBox3	txt	Name.	txtDocumento
		text	vacío
Button1	btn	Name	btnIngresar
		text	INGRESAR

Button2	btn	Name	btnLimpiar
		text	LIMPIAR



## Claves de autocorrección

Te dejamos una posible resolución de la práctica formativa "Primera solución en el entorno visual" para que puedas cotejar con tu producción.

**Video tutorial:**  
Primera solución  
en el entorno.

DS00 S2 "Primera solución en el entorno visual"





- **Tipo de actividad:** Práctica formativa autogestionada con claves de autocorrección.
- **Objetivo:** Manipular los datos de entrada para mostrar por cuadro de diálogo los resultados.

**Descripción :**

Partiendo del diseño que se detalla a continuación, modificando las características que consideres necesarias y sin olvidar la regla del uso de prefijos, se pide resolver cada punto-

**Para hacer:**

1. Ingresar los datos desde los textBox y mostrarlos en cuadro de dialogo. Usar el botón "Aceptar".
2. Tomar el ejercicio 1, pero los datos deben ser ingresados también en un cuadro de diálogo.
3. Usando el botón "Cálculo" solicitar con un cuadro de diálogo el ingreso de dos números (es un cuadro de dialogo para cada dato) y mostrar cuál es el número más grande. La forma de declaración, lectura de datos numéricos y uso de condicional lo viste en la semana 1 y lo ejercitaste desde el modo consola. Simplemente debes adaptarlo a este nuevo modo.

INGRESO Y MUESTRA DE DATOS

DATO 1

DATO 2

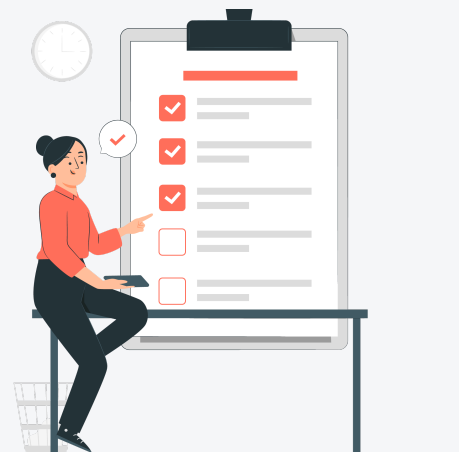
ACEPTAR

CALCULO



## **Claves de autocorrección**

Te dejamos una posible resolución de la práctica formativa "Ingresando datos" para que puedas cotejar con tu producción.



Descargá la carpeta de archivos [en este enlace](#) para acceder a las **claves de autocorrección**.



- **Tipo de actividad:** Práctica formativa autogestionada con autocorrección.
- **Objetivo:** Escribir en C# para comprender distintas estructuras de control.

#### Leé el enunciado para resolver la práctica

Tres personas aportan diferente capital a una sociedad y desean saber el valor total aportado y qué porcentaje aportó cada una (indicando nombre y porcentaje). Para ello, solicitar la carga por teclado del nombre de cada socio, su capital aportado y a partir de esto calcular e informar lo requerido previamente .

#### ¿Qué tenés que realizar?

1. Instalar Visual Studio 2022 con C# ([hacé clic aquí](#) para acceder al manual de instalación).
2. Releer el enunciado de más arriba
3. Crear las variables,
4. Pedir datos al usuario/a y,
5. Mostrar por pantalla.
6. Comparar con la resolución posible que se encuentra en...



## **Claves de autocorrección**

Te dejamos una posible resolución de la práctica formativa "Escribir en C#" para que puedas cotejar con tu producción.

Descargá la carpeta de archivos [en este enlace](#) para acceder a las **claves de autocorrección**.

