Dependency injection

Conceptos previos

Clase

- Plano técnico que especifica lo que el tipo de objeto puede hacer.

Objeto

- Bloque de memoria que se ha asignado y configurado de acuerdo con el plano.

Variables de instancia

- Se relaciona con una única instancia de una clase (No con la clase estática).
- Son declaradas fuera del cuerpo de los métodos y dentro de la clase, por lo tanto son de tipo global.
- Se crean cuando se instancia el objeto.
- Pueden ser utilizadas por cualquier método no estático de dicha clase.

Abstracción

- Modelar los atributos e interacciones de las entidades que se relacionan para cumplir un objetivo común, ocultando los detalles internos de implementación.

Encapsulamiento

- Ocultar el estado interno y la funcionalidad de un objeto y permitir solo el acceso a través de un conjunto público de funciones.

Herencia

- Capacidad de crear nuevas abstracciones basadas en abstracciones existentes.

Polimorfismo

- Capacidad de implementar propiedades o métodos heredados de maneras diferentes en varias abstracciones.

Dependencia

- Una dependencia es un objeto del que depende otro objeto.

Interface

- Una interfaz define un contrato. Cualquier clase o estructura que implemente ese contrato debe proporcionar una implementación de los miembros definidos en la interfaz.

Inversión de control (Inversion of Control IoC)

- La dirección de la dependencia dentro de la aplicación, debe in en la dirección de las abstracciones y no de los detalles de implementación.

- Los detalles de implementación se escriben para depender e implementar abstracciones de alto nivel, en lugar de hacerlo al revés.

"Pienso que necesitamos un nombre más específico para este patrón. Inversión de control es un término muy genérico, y por eso, a la gente le parece confuso. Como resultado de una larga discusión con varios defensores de IoC, nos decidimos por el nombre Inyección de dependencia"

Fuente: Martin Fowler - Inversion of Control Containers and the Dependency Injection pattern

Inyección de dependencias

La definición más simple posible

- Significa darle a un objeto sus variables de instancia.
Fuente: James Shore - Dependency Injection Demystified

- Formas de darle a un objeto sus variables de instancia
    - Constructor Injection.
    - Setter Injection.

Ejemplo sin usar contenedores

Continuando con el tutorial de POO, donde se crearon las cuentas bancarias.
Aparece un nuevo requerimiento:
- Los retiros de dinero de las cuentas de crédito, deben ser validados por un sistema de IA.
Se plantea agregar un servicio para validar los retiros generados en dichas cuentas.

Ver ejemplo de código en Github

Analogía

Los autos se pueden pensar como servicios que sirven para transportar gente de un lugar a otro. Los motores de los autos requieren nafta, diesel o electricidad, pero esos detalles no son importantes para el conductor, que solo le importa si puede llegar a destino.

Los autos presentan una interface uniforme a través de los pedales, volante y otros controles.
Con que motores fueron "inyectados" los autos en la fábrica, deja de importar, y los conductores pueden intercambiar de auto como sea necesario.

Fuente: https://en.wikipedia.org/wiki/Dependency_injection

.NET dependency injection and service containers

.NET soporta el patrón de diseño Inyección de dependencias, que es una técnica para alcanzar la Inversión de Control entre clases y sus dependencias.

La inyección de dependencia está incorporada en el framework .NET, al igual que los patrones de diseño como Configuration, Loggin y Options.

Como implementa la inyección de dependencia .NET

- Usando interfaces o clases base para abstraer la implementación de dependencias.

- Registrando las dependencias en un service container. NET tiene incorporado un service container, IServiceProveder. Los servicios se suelen registrar cuando inicia la aplicación, agregándo a los mismos a una IServiceCollection. Una vez que todos los servicios fueron agregados, se usa BuildServiceProvider para crear el service container.

- Inyectando los servicios en el constructor de las clases donde son usados. El framework toma la responsabilidad de crear las dependencias y de desecharlas cuando no se las requiere más.

ver Ejemplo de código en Github

Extra

"OO evita que el software desarrollado sea rígido, frágil y no reusable."
Uncle Bob

Si una clase tiene inyectada demasiadas dependencias, puede ser un signo de que esa clase tiene muchas responsabilidades y está violando el Single Responsability Principle (SRP). Intente refactorizarla moviendo algunas de esas responsabilidades a nuevas clases.
Fuente: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection-guidelines

Autores software development y OOP

Martin Fowler (1953 -) British.
Alistair Cockburn (1953 -) American.
Craig Larman (1958 -) Canadian.
Robert C. Martin (Uncle Bob) (1952 -) American.

UML principalmente
Ivar Jacobson (1929 -) Swedish, American.
Grady Booch (1955 -) American.
James Rumbaugh (1947 -) American.