ACTIVIDAD PRÁCTICA 8 (AP8)

Título

Entendiendo la Programación Orientada a Objetos (POO) – Parte 1

Objetivos

- Conocer y aprender los conocimientos básicos teóricos necesarios para iniciarse en el uso del Paradigma de programación orientado a objetos.

Temporalización

Previsto: Una sesión de clase presencial + trabajo en casa (período del 11 al 18 de enero) – una semana.

Proceso de desarrollo

Tras haber trabajado durante la 1ª evaluación la programación estructurada y modular con el lenguaje JavaScript (bloque temático 1), y habernos introducido en la sintaxis y características del lenguaje de programación C# (bloque temático 2) que utilizaremos durante el resto de las evaluaciones (2ª y 3ª), iniciamos el boque temático 3, en el cual vamos a **conocer y utilizar el paradigma de la programación orientada a objetos (POO)**.

Dicho paradigma requiere de unos **conocimientos teóricos de base importantes y necesarios** para poder desarrollar de forma práctica aplicaciones que utilicen correctamente dicho paradigma orientado a objetos.

Esta **Actividad Práctica (AP8),** persigue la obtención de dichos conocimientos teóricos básicos que deberán trabajarse de forma simultánea junto con ejercicios prácticos de desarrollo en los cuales se vayan utilizando y aplicando los conceptos teóricos en los que se basa la POO.

Esta actividad práctica se basa en el curso de libre acceso disponible en internet denominado (ENTENDIENDO LA PROGRAMACIÓN ORIENTADA A OBJETOS) y cuya referencia WEB tienes disponible en la sección de <material de apoyo/consulta> de los recursos didácticos del bloque.

Deberás realizar la visualización de cada uno de los videotutoriales propuestos en la siguiente tabla y a continuación, responder de forma detallada a las preguntas/cuestiones formuladas para cada uno de los videos visualizados.

Ten en cuenta que el aprendizaje de los conceptos explicados en los videos es fundamental y totalmente necesario para poder utilizar correctamente de forma práctica (en ejercicios y proyectos prácticos) el paradigma de la programación orientada a objetos. Nuestra forma de programar debe ajustarse y variar hacia esta "nueva forma de pensamiento computacional orientada a objetos".

Visualiza los videos propuestos y responde a las siguientes preguntas:

Video1. POO. Conceptos Fundamentales.

1. ¿Cuáles son los conceptos fundamentales en los que se basa la POO?. Existe otro concepto, no presentado en el video que es la Abstracción. Investiga cual sería el significado de dicho término en el ámbito de la POO.

   Encapsulación, Herencia y polimorfismo. De forma añadida la Abstracción

2. Enumera y razona las principales diferencias existentes entre los dos paradigmas de programación tratados en el video (Orientada a Objetos y Estructurada/Procedural).

En la POO los datos y comportamientos están juntos (dentro de clases), mientras que en la programación estructurada van por separado (por una parte tenemos datos y por otra funciones). También es diferente la forma de ejecución de las aplicaciones, ya que en la estructurada sigue una forma secuencial con llamadas a funciones y en la POO se realiza a través de la interacción entre los objetos y mensajes entre ellos.

Video2. POO. Entiendo los objetos

1. Un objeto en la POO se puede entender como un componente de un sistema. Teniendo en cuenta esta definición ¿cómo definirías una aplicación programada con el paradigma orientada a objetos?

Una colección de objetos que forman un sistema y que interactúan entre sí teniendo cada uno de los objetos / componentes una responsabilidad dentro del sistema. Los objetos interactúan entre ellos y se comunican para lograr las funcionalidades deseadas en el programa.

2. Los objetos guardan datos en su interior (no visibles para el exterior). ¿Cómo se denominan dichos datos?

Atributos (características)

3. ¿Qué es lo que se conoce como el estado de un objeto?

Es la información que tiene el objeto en un momento particular en el tiempo. Cada vez que cambia alguno de los valores de los atributos del objeto cambia el estado del objeto. El estado del objeto cambia a través del tiempo.

4. Los objetos, en su interior tienen atributos y comportamientos, que podrán ser invocados (utilizados desde diversos lugares) dependiendo del método de acceso que tengan asignado. ¿Cuáles son los métodos de acceso básicos existentes en la POO? ¿Desde donde se tiene acceso al atributo o comportamiento del objeto según cada método de acceso?

PUBLICO → Acceso desde el exterior del objeto / PRIVADO → Acceso sólo desde el interior del objeto. / PROTEGIDO → Acceso desde dentro de la cadena de herencia establecida.

5. Los objetos, además de atributos, van a tener comportamientos. ¿Qué son dichos comportamientos? ¿Dónde se programan los comportamientos de un objeto?

El comportamiento es lo que el objeto puede realizar, hacer, o llevar a cabo, programándose y codificándose en métodos (funciones). El conjunto de todos estos métodos nos indica el comportamiento del objeto.

6. ¿Qué son los mensajes en POO?

Es el mecanismo de comunicación entre objetos, de tal forma que se utiliza un mensaje cuando dentro de un objeto se invoca el método de otro objeto (se manda un mensaje).

7. Para realizar o enviar un mensaje (invocación de un método dentro de un objeto), ¿qué se necesita conocer?

El nombre del método que se desea invocar del objeto, los parámetros o argumentos a pasar al método y el tipo de retorno/devolución realizada por el método.

Video3. Entendiendo las clases y la encapsulación.

1. ¿Qué es una clase y de qué elementos se compone?

Especie del molde a partir del cual vamos a crear un objeto. Es posible pensar en las clases como tipos de alto nivel definidos por el programador. Las clases tienen atributos y comportamientos, que son los que tendrán los objetos que se creen a partir de dicha clase.

2. ¿Qué se conoce como instanciación? ¿Es lo mismo un objeto que una instancia de una clase?

Es la acción de creación de un objeto a partir de una clase. A dicho objeto se le denomina también instancia de la clase a partir de la cual se ha creado. Un objeto es, por tanto, una instancia concreta de una clase a partir de la cual se ha creado. (SI).

3. ¿Qué se conoce como encapsulación?

La combinación de los atributos y comportamientos dentro de un objeto es lo que se conoce como encapsulación. Percibimos el objeto como una cápsula (caja negra) que contiene datos -información- y comportamiento -acciones sobre esa información-. Es un pilar básico de la POO.

4. La encapsulación protege del exterior tanto a los atributos como a los comportamientos que tiene el objeto en su interior. Teniendo en cuenta lo anterior, ¿qué atributos/comportamientos se deben mostrar/revelar al exterior del objeto?

Solamente mostraremos al exterior aquellos atributos/comportamientos (métodos) con los que otros objetos deban interactuar, debiendo dejar oculto aquellos atributos/comportamientos que no deban ser utilizados desde el exterior, para el cual estos elementos no serán conocidos.

5. Teniendo en cuenta el principio de la encapsulación, ¿cómo logramos hacer visibles o invisibles los atributos y/o comportamientos de un objeto?

Haciendo uso de los métodos de acceso (PUBLICO / PRIVADO / PROTEGIDO). El acceso public permite al exterior conocer aquello que nos interese y utilizaremos el acceso private para ocultar o bloquear al exterior aquello a lo que no necesita acceder para invocarlo, leerlo o modificarlo.

6. ¿Qué es el Data Hiding?

Es la técnica utilizada para esconder o restringir el acceso desde el exterior a los atributos (datos o información) de un objeto. Un buen Data Hiding implica mostrar al exterior únicamente la información que necesite para trabajar y esconder todo lo demás. Esto eleva la seguridad sobre los datos del objeto.

7. El concepto de Data Hiding está muy relacionado el de Encapsulación. ¿En qué sentido?

Un buen Data Hiding permite una encapsulación correcta, La encapsulación engloba a los datos y comportamientos en una especie de “cápsula” y el principio de Data Hiding protege a dichos elementos del exterior, mostrando o haciendo visibles aquellos que únicamente sean necesarios, convirtiendo esa “cápsula” en una especie de “caja negra” donde muchos aspectos del objeto quedan ocultos.

8. Para lograr un Data Hiding y por tanto una encapsulación correcta es primordial mantener los atributos de un objeto con acceso privado (private). Esto se logra haciendo uno de unos métodos especiales denominados Accesor (métodos de acceso). Describe cuál sería su funcionamiento.

Al ser privados los datos/información del objeto, sólo se pueden acceder desde el interior del objeto. Para poder manipularlos se crean métodos de acceso (Accesor) públicos que se puedan invocar desde el exterior. Dentro de dichos métodos se puede incluir código de seguridad y/o reglas para poder modificar

los datos del interior del objeto o para poder suministrar datos al exterior del objeto logrando una mayor seguridad.

9. Indica las dos formas básicas para poder implementar los métodos de acceso (Accesor).

Mediante funciones de interfaces denominados métodos getter y setter (get y set). Estas son generales a todos los lenguajes de POO. En C# existen otro mecanismo utilizable que son las Propiedades.

10. Un objeto tiene un comportamiento implementado a través de diversos métodos o acciones. ¿Cuándo se define como Mutator (Mutadores) a un método perteneciente al comportamiento del objeto?

Cuando su ejecución provoca un cambio de estado del objeto, es decir, modifica la información (valores de los atributos) que tiene el objeto. Son métodos invocados desde el exterior que provocan un cambio de estado del objeto.

Video4. Conceptos de Interface, Herencia y Polimorfismo.

1. ¿A que denominamos Interface de un objeto (o de una clase)?

Son aquellos elementos (atributos y/o métodos) que se muestran al exterior y por tanto son públicos (medio de comunicación entre los objetos). La interface de un objeto muestra la forma en la que los usuarios del objeto pueden comunicarse con dicho objeto.

2. Todo atributo o método que tenga acceso público (public) formará parte de la interface de un objeto, no obstante, y teniendo en cuenta el principio del Data Hiding en la POO. ¿Qué elementos deberán formar parte de la interface de un objeto?

Aquellos métodos públicos que exclusivamente se considere necesarios mostrar al exterior para que puedan ser invocados por los usuarios del objeto. El acceso para consultar o modificar los atributos del objeto (que deben ser privados para lograr un correcto encapsulamiento) deberá realizarse a través de métodos de acceso públicos (Acceso del tipo getter o setter o mediante el uso de Propiedades).

3. El principio de la Herencia nos permite una gran ventaja de la POO como es la reutilización de código. Define en qué consiste el mecanismo de la Herencia. Puedes ayudarte de un ejemplo demostrativo.

La herencia es un mecanismo que nos permite reutilizar el código de una Clase A para crear una nueva Clase B que aprovecha todo el código (datos y comportamientos) heredados de la Clase A. Podemos tomar como ejemplo una clase EMPLEADO, que herede los atributos y métodos de una clase PERSONA.

4. El mecanismo de herencia involucra a dos clases. Una clase más abstracta a partir de la cual existe otra clase más concreta que hereda (atributos y métodos) de la clase más abstracta. ¿Qué denominaciones pueden recibir dichas clases involucradas en la herencia?

La clase de la cual se hereda (más abstracta) se denomina SUPERCLASE / CLASE PADRE o CLASE BASE. La clase que recibe la herencia (más concreta) se denomina SUBCLASE / CLASE HIJA o CLASE DERIVADA.

5. El uso de la herencia en POO conlleva la aparición de una relación ES-UN entre las dos clases involucradas (padre e hija). Ejemplos serían: 1) clase EMPLEADO que hereda de clase PERSONA (Relación: empleado es una persona). 2) clase AUTOMOVIL que hereda de clase MEDIOTRANSPORTE (Relación: automóvil es un mediotransporte). Piensa y enumera cinco ejemplos más de objetos de la vida real que puedas abstraer en clases y que puedan tener una relación de herencia (ES-UN) entre ellos.

Clase CIRCULO que hereda de clase FIGURA (Relación: circulo es una figura). 2) Clase MESA que hereda de clase MUEBLE (Relación: mesa es un mueble). 3) Clase APARTAMENTO que hereda de clase VIVIENDA

(Relación: apartamento es una vivienda). 4) Clase CHALET que hereda de clase VIVIENDA (Relación: chalet es una vivienda). 5) Clase CUADRADO que hereda de clase FIGURA (Relación: cuadrado es una figura).

6. El Polimorfismo es otro pilar básico de la POO muy relacionado con la Herencia. Polimorfismo quiere decir "muchas formas". Suponiendo el siguiente ejemplo: Tenemos una clase ANIMAL con al menos un método implementado denominado "Gritar", si creamos dos clases que derivan o heredan de ANIMAL y que denominamos PERRO y GATO, estas dos clases hijas de la clase ANIMAL dispondrán también del método "Gritar". ¿La técnica del Override qué nos permitiría hacer sobre el método "Gritar" de las clases hijas?

Permite implementar de forma diferente y sobrescribir/reemplazar el código heredado de la clase padre en el método "Gritar" para lograr un comportamiento más adecuado en el objeto concreto. En el caso de un objeto de la clase PERRO podemos reemplazar el método "Gritar" para que su grito sea "GUAU" mientras que la clase GATO lo haremos para que su grito sea "MIAU".

7. La Composición es otro concepto importante de la POO (aunque no forma parte de los cuatro pilares de la POO (Abstracción, Herencia, Encapsulación y Polimorfismo). ¿En qué consiste o se basa el principio de Composición?

Permite la reutilización de código al permitir que un objeto pueda estar formado por otros objetos. Los atributos de un objeto pueden ser a su vez otros objetos. Ejemplos serían: 1) un objeto COCHE puede contener a su vez un objeto MOTOR. 2) un objeto HOTEL puede contener a su vez una lista de objetos HABITACIÓN.

8. La Herencia establece una relación ES-UN entra la clase hija que hereda de la clase padre. En el caso de la Composición ¿Qué tipo de relación se establece entre las dos clases involucradas?

Una relación TIENE-UN. En los ejemplos anteriores: 1) un objeto de la clase COCHE tiene-un objeto de la clase MOTOR. 2) un objeto de la clase HOTEL tiene una lista de objetos de la clase HABITACIÓN.

Video5. Diferencia entre Interface e implementación en un objeto.

1. Tal como se han presentado en el video anterior (video4) la interface de una clase son los elementos (principalmente métodos públicos) que son visibles para los usuarios de la clase (quienes utilicen objetos de la clase). En el diseño de la clase deberemos de pensar qué debe formar parte de la interface. Describe las principales recomendaciones para crear una buena interface de una clase.

Debe ser lo más mínima posible, empezando por no incluir nada en la misma e ir agregando métodos a la interface según vamos viendo qué se necesitan el objeto para interactuar con el sistema (aplicación). Según la vamos creando la interface de la clase debemos ir preguntándonos si realmente las acciones públicas que ofrecer el objeto en su interface contribuye correctamente a su funcionamiento deseado y a la comunicación que ofrece a los objetos con los que interactúa.

2. Debemos tener en cuenta que una buena técnica de programación orientada a objetos requiere que haya una separación clara entre Interface e Implementación. ¿Qué sería la implementación en una clase/objeto? ¿Cuál es el beneficio de tener una buena separación entre ambas partes?

La implementación en una clase/objeto es cualquier elemento (atributo/método) que no forme parte de la interface. Son principalmente los métodos ocultos (privados) a los usuarios de la clase y que se invocan internamente para el uso de la clase.

Al separar la interface de la implementación de un objeto hace que cualquier cambio interno en la parte de implementación no provoque la necesidad de ningún cambio en otras clases u objetos que usan el objeto, ya que estos se comunican con la Interface y esta no varía al ser totalmente independiente de la parte de implementación.

Video6. Beneficios de la Programación orientada a objetos (POO) – 1

Video7. Beneficios de la Programación orientada a objetos (POO) – 2

1. ¿Qué significa realizar un cambio de paradigma a la POO?

Usar correctamente las técnicas de programación orientada a objetos en lenguajes que permiten dicha POO. Un problema común es programar utilizando el paradigma estructurado/procedural con un lenguaje orientado a objetos.

2. Enumera los principales beneficios que proporciona la utilización de la POO.

Es natural (modela aquello que estamos viendo en el mundo real) de tal forma que se programa en términos del problema y la aplicación se concibe como un sistema de objetos que interactúan y trabajan entre ellos y para el cual debemos construir un modelo funcional (más abstracto) que resuelva el problema real planteado.

Es confiable (La POO permite modificar una parte del programa -especialmente la parte de implementación- sin que afecte a otras partes. Una vez se ha testeado y comprobado la validez de un componente/clase se puede reutilizar con confianza).

Es reutilizable (Las clases se pueden reutilizar en muchos programas o se puede reutilizar código por medio del mecanismo de la herencia).

Es fácil de mantener (Al seguir el principio de la encapsulación, los errores se detectan y corrigen en una sola parte).

Es extensible (permite que el software sea escalable -pueda crecer y cambiar de forma dinámica- añadiendo nuevas funcionalidades al software existente a través de nuevas clases o componentes sin tener que modificar el sistema ya existente.

Es oportuno (se reduce el tiempo de desarrollo en comparación con la Programación estructurada, pudiendo desarrollar clases en paralelo dentro del sistema que forma la aplicación, siempre atendiendo a un diseño previo y correcto).

Video8. Concepto de Constructor.

1. Define qué es un Constructor y cuáles son sus características principales.

Es el método que se invoca automáticamente en el momento que se instancia un objeto de la clase. Debe tener siempre el mismo nombre que tiene la clase (así se distingue) y no devuelve/retorna nada. Como Constructor no tiene tipo (ni void) y tiene acceso público (public) para que puede utilizarse desde fuera de la clase y realizar la instanciación de un objeto. En caso de no programar un Constructor para la clase no significa que no disponga del mismo para instanciar objetos (existe siempre un constructor por defecto que se utiliza para instanciar objetos de la clase -se invoca al constructor de la clase base-).

2. Dada la Clase FIGURA Cuando ejecutamos la siguiente instrucción: Figura cuadrado = new Figura(); ¿Qué se está produciendo?.

En primer lugar (en la parte izquierda) se crea una variable llamada cuadrado que es de tipo Figura. En segundo lugar, una vez creada la variable cuadrado, se crea una instancia de un objeto de la clase Figura que es asignada a la variable cuadrado, que apuntará a la zona de memoria donde está la nueva instancia creada.

3. ¿Qué ventajas nos proporciona el uso de un constructor?

Al ser un método que se invoca automáticamente cuando se construye el objeto, es un buen lugar para inicializar los valores de los atributos del objeto (podemos crear un estado inicial del objeto que sea correcto, estable y seguro).

4. ¿Qué es la sobrecarga de constructor?

Es la posibilidad de disponer o implementar varias versiones diferentes del método Constructor que tengan diferentes parámetros para inicializar los valores/datos del objeto.

Video9. **Manejo de errores, sobrecargas de métodos y herencias múltiples**

1. En el paradigma de POO el manejo de excepciones es uno de los mecanismos más utilizados para tratar de evitar y gestionar errores que se produzcan en tiempo de ejecución. Describe de forma básica cómo funciona su mecanismo de uso.

Se pueden lanzar excepciones en aquellas zonas de código que podamos detectar que pueden tener algún posible error o evento inesperado en tiempo de ejecución. Esa zona de código es englobada dentro de un bloque <try> y en caso de que se produzca un evento inesperado o error la ejecución no se interrumpe de forma anormal, sino que se deriva al boque de código <catch> para que se trate el error producido de una forma global o específica atendiendo a la excepción que se haya producido.

2. ¿Qué es la sobrecarga de métodos en un objeto/clase?

Sobrecargar un método significa disponer de diferentes versiones de un mismo método de un objeto que puede utilizarse de diferentes formas (con un número de parámetros distintos, etc…). Dependiendo de los parámetros que utilicemos, el lenguaje de programación detectará qué versión del método sobrecargado necesitaremos utilizar.

3. La herencia múltiple permite que una clase herede de diferentes clases distintas. Normalmente una clase hija deriva de una única clase padre, pero la herencia múltiple permitiría que una subclase pudiese derivar de varias superclases. ¿El lenguaje C# permite el uso de herencia múltiple?

NO

Video10. Ámbito o Scope.

1. ¿Qué es el Ámbito en el paradigma de POO y qué niveles de ámbito existen?

Es la región o zona de la clase donde un atributo o variable es conocido y se puede usar. Existen tres niveles de ámbito (nivel LOCAL, nivel de OBJETO y nivel de CLASE).

2. Describe cómo funciona el ámbito a nivel LOCAL.

El ámbito local funciona cuando declaramos un atributo o variable dentro de un método. Dicho atributo y/o variable sólo será conocida y utilizable dentro de dicho método, no siendo reconocidos fuera del bloque del método donde han sido declarados. Esto permite tener declaradas variables con el mismo nombre en diferentes métodos (serán distintas) y si queremos comunicar los valores entre métodos habrá que realizarlo mediante el paso de parámetros, lo cual dota de mayor seguridad al uso de los métodos.

3. Describe cómo funciona el ámbito a nivel de OBJETO. Indica donde se declararían los atributos con un ámbito a nivel de OBJETO.

El ámbito a nivel de OBJETO hace que los atributos que declaremos a dicho nivel sean conocidos en todos los métodos (públicos o privados) que contenga la clase. Podrán ser utilizados (leerlos y/o modificarlos), por tanto, por dichos métodos sin necesidad de ser pasados como parámetros.

Se declararían dentro de la clase (normalmente al principio de la misma y antes de crear los métodos) y fuera de los métodos de la clase.

4. El ámbito a nivel de OBJETO puede provocar un conflicto cuando tenemos un atributo (declarado a nivel de OBJETO) y una variable declarada dentro de un método (a nivel LOCAL) y donde ambos tienen en mismo nombre. ¿Qué se utiliza para diferenciar ambos elementos?

Se utiliza la palabra reservada <this>, que se antepone con un punto al nombre del atributo declarado a nivel de OBJETO para reconocerlo. Por ejemplo: this.salario (se referiría al atributo definido con ámbito a nivel de OBJETO).

5. Describe cómo funciona el ámbito a nivel de CLASE.

Cuando declaramos un atributo a nivel de clase, este atributo es compartido por todos los objetos instanciados de la clase o que pertenecen a la clase. En caso de que un objeto cambie el valor de dicho atributo, dicho cambio se verá reflejado en el resto de los objetos de la clase y por tanto será reconocido/compartido por todos.

6. Para declarar un atributo con un ámbito a nivel de CLASE es necesario hacer uso de un modificador especial. Indica cual sería este modificador y dónde se declararía.

El modificador sería <static> y se declararía en la misma zona que los atributos a nivel de OBJETO (dentro de la clase, pero fuera de los métodos).

Evaluación

Esta actividad práctica no es una actividad de evaluación. Su realización forma parte de la evaluación correspondiente al factor multiplicador (seguimiento e interés de la asignatura por parte del alumnado). Supone un refuerzo / estudio necesario para complementarlo con los ejercicios prácticos a desarrollar en el bloque temático.

Recursos

Disponibles en plataforma (Recursos didácticos del Bloque3):

- VideoTutoriales (Curso: Entendiendo la Programación orientada a objetos - POO).
- Libro2. Lenguaje C#.
- Documentación de ayuda .NET de Microsoft.