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 de enero 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 un 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 (y también en la sección de recursos de esta actividad).

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: la orientación a objetos".

Visualiza los videos propuestos y responde a las siguientes preguntas. Como resultado de la actividad práctica deberás entregar un documento de WORD (RESPUESTAS.doc) con tus respuestas a las preguntas formuladas.

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.

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

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?
2. Los objetos guardan datos en su interior (no visibles para el exterior). ¿Cómo se denominan dichos datos?
3. ¿Qué es lo que se conoce como el estado de un objeto?
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?.
5. Los objetos, además de atributos, van a tener comportamientos. ¿Qué son dichos comportamientos? ¿Dónde se programan los comportamientos de un objeto?
6. ¿Qué son los mensajes en POO?
7. Para realizar o enviar un mensaje (invocación de un método dentro de un objeto), ¿qué se necesita conocer?

Video3. Entendiendo las clases y la encapsulación.

1. ¿Qué es una clase y de qué elementos se compone?
2. ¿Qué se conoce como instanciación? ¿Es lo mismo un objeto que una instancia de una clase?
3. ¿Qué se conoce como encapsulación?
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?
5. Teniendo en cuenta el principio de la encapsulación, ¿cómo logramos hacer visibles o invisibles los atributos y/o comportamientos de un objeto?
6. ¿Qué es el Data Hiding?
7. El concepto de Data Hiding está muy relacionado el de Encapsulación. ¿En qué sentido?
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.
9. Indica las dos formas básicas para poder implementar los métodos de acceso (Accesor).
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?

Video4. Conceptos de Interface, Herencia y Polimorfismo.

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

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?
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.
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?
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?
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?

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.
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?

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

Video8. Concepto de Constructor.

1. Define qué es un Constructor y cuáles son sus características principales.
2. Dada la Clase FIGURA Cuando ejecutamos la siguiente instrucción: Figura cuadrado = new Figura(); ¿Qué se está produciendo?.
3. ¿Qué ventajas nos proporciona el uso de un constructor?
4. ¿Qué es la sobrecarga de constructor?

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

Video10. Ámbito o Scope.

1. ¿Qué es el Ámbito en el paradigma de POO y qué niveles de ámbito existen?
2. Describe cómo funciona el ámbito a nivel LOCAL.
3. Describe cómo funciona el ámbito a nivel de OBJETO. Indica donde se declararían los atributos con un ámbito a nivel de OBJETO.
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?
5. Describe cómo funciona el ámbito a nivel de CLASE.
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.

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.