CLASES Y OBJETOS

Objetivos

- Comprender el proceso de diseño basado en objetos para la solución de problemas
- Identificar los procesos para crear y usar objetos en un lenguaje de programación de alto nivel

Recursos requeridos

- PC con NetBeans instalado
- PC con Visual Studio Code

Actividades preliminares al laboratorio

- Lectura de la guía

Marco teórico

Programación orientada a objetos: La programación orientada a objetos (POO) es un paradigma de programación, es decir, un estilo o forma de estructurar un programa. En POO, los objetos manipulan los datos de entrada para la obtención de los datos de salida, cada objeto tiene una funcionalidad especifica.

¿Qué es un objeto?

Los objetos representan cosas, que pueden ser simples o complejas, pueden ser reales o imaginarios. Cada objeto tiene unas características o valores denominados atributos, los cuales permiten definir el estado del objeto. Adicionalmente, cada objeto tiene asociadas las acciones que puede realizar definidas a través de métodos, los cuales se ejecutan de acuerdo con los parámetros o argumentos de entrada y pueden devolver un valor al acabar su ejecución.

Ejemplo:

• Atributos contantes: Marca, color, potencia, velocidad máxima.

• Atributos variables: velocidad, aceleración
• Métodos: arrancar, parar, acelerar, frenar, girar a la derecha, girar a la izquierda

¿Qué es una clase?

Las clases representan un tipo particular de objetos, agrupando objetos con características y comportamiento similares. Ejemplo clase carro:

Cada clase tiene asociado una definición que determina:
• Los atributos que tienen los objetos de la clase
• Los métodos que pueden ejecutar los objetos de la clase y como lo hacen

La POO consiste en diseñar e implementar las clases de objetos que permiten la solución de un problema. Un programa en POO consta de un conjunto de instancias de una clase (objetos) y un flujo de control principal (main). Durante la ejecución del programa los objetos se crean y se destruyen, y se solicita a los objetos que ejecuten métodos.

Metas de POO

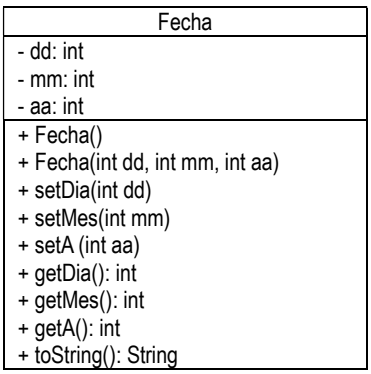
• Robustez: un programa se considera robusto cuando es capaz de manejar una entrada inesperada. En general, un programa funciona correctamente, produciendo la salida correcta dada una entrada esperada. Sin embargo, en el caso de entradas no esperadas (por ejemplo, ingresan caracteres en vez de enteros), el programa debe dar una respuesta.
• Adaptabilidad: un programa debe ser capaz de adaptarse en el tiempo al cambio de condiciones. Un concepto relacionado a la adaptabilidad es la portabilidad, la cual es la habilidad de un programa para correr con cambios mínimos en diferentes hardware o sistemas operativos.
• Reusabilidad: se espera que le mismo código pueda ser usando como un componente de diferentes sistemas en diversas aplicaciones.

Principios de POO

• Abstracción: se refiere a dividir un sistema complejo en sus partes más fundamentales. Definir las partes de un sistema incluye nombrarlas y explicar su funcionalidad.
• Encapsulación: los componentes de un sistema no deben revelar detalles internos de sus respectivas implementaciones. Una de las principales ventajas de la encapsulación es que un programados tiene la libertad de implementar un componente, solo debe mantener la interfaz (como se relaciona con otros componentes). Esto permite la robustez y adaptabilidad de los programas.
• Modularidad: principio de organización en el cual los diferentes componentes de un programa se dividen en diferentes unidades funcionales.

Actividades a desarrollar durante el laboratorio

Vamos a realizar la implementación de una clase Fecha en JAVA y Python, para identificar la estructura de ambos lenguajes de programación. La clase Fecha se encuentra detallada en el siguiente diagrama UML.



- Implementación de la clase Fecha

- Instanciación de objetos tipo Fecha
- Prueba de los settings y gettings
- Implementación del método toString()

Nota: Una vez finalizada la práctica, copia de los códigos realizados en clase estarán disponibles en el Moodle.

Actividades a desarrollar por los estudiantes:

Parte 1. Implementación de las clases Direccion y Usuario en Java – Ejercicio practico

Usuario	
- nombre: String - id: long - fecha_nacimiento: Fecha - ciudad_nacimiento: String	- tel: long - email: String - dir: Direccion
+ Usuario() + Usuario(String n, long id) + setNombre(String n) + setId(long id) + setFecha_nacimiento(Fecha f) + setCiudad_nacimiento(String c) + setTel(long t) + setEmail(String e) + setDir(Direccion d)	+ getNombre():String + getId():long + getFecha_nacimiento():Fecha + getCiudad_nacimiento():String + getTel():long + getEmail():String + getDir():Direccion + toString(): String

Direccion	
- calle: String - nomenclatura: String - barrio: String	- ciudad: String - edificio: String - apto: String
+ Direccion() + setCalle(String c) + setNomenclatura(String n) + setBarrio(String b) + setCiudad(String ci) + setEdificio(String e) + setApto(String a)	+ getCalle():String + getNomenclatura():String + getBarrio():String + getCiudad():String + getEdificio():String + getApto():String + toString():String

Vamos a crear las clases Direccion y Usuario de acuerdo con los diagramas de clase. Una vez implementados:

A. En un programa principal:
- Crear un objeto tipo fecha con la información de su fecha de nacimiento e imprimir en pantalla los datos en el formato dd – mm – aa.
- Crear un objeto tipo dirección con su dirección de residencia e imprimir en pantalla los datos en una sola cadena.
- Crear un objeto tipo Usuario con su información personal y de contacto, imprimir en pantalla todos los atributos del objeto creado.

B. En un programa principal:

- Solicite por consola la información personal y de contacto requerida para construir un Usuario
- Cree un objeto empleando la información ingresada por consola, e imprima en pantalla todos los atributos del objeto.

Parte 2. Implementación de las clases Direccion y Usuario en Python – Ejercicio practico

Repita el ejercicio de la parte 1, implementando la solución en Python. Se recomienda el uso de Visual Studio Code para esta segunda parte.

Entrega de actividades parte 1 y parte 2: los programas funcionales tanto en JAVA como en Python deben ser sustentados durante la sesión del 8 de mayo.