UNIVERSIDAD DE MANIZALES
FACULTAD DE CIENCIAS E INGENIERÍAS

Asignatura: Programación I

Actividad colaborativa: taller práctico

Orientaciones de desarrollo: para el desarrollo de la presenta actividad los estudiantes conformarán equipos de trabajo de máximo 5 integrantes. Luego descargarán el archivo anexo en el cual se presentan casos reales de modelamiento de clases e instanciamiento de objetos. Como resultado de la actividad los estudiantes subirán a la plataforma los archivos de Python completamente documentados con su extensión .py.

Ejercicio 1: Creación de una Clase Básica

- Objetivo: Comprender la estructura básica de una clase y la creación de objetos.
- Descripción: Crea una clase llamada Coche con atributos como marca, modelo y año. Define un método describir() que imprima la información del coche. Luego, crea varios objetos Coche y utiliza el método describir() para mostrar la información.
- Conceptos: Clases, objetos, métodos.

Ejercicio 2: Encapsulamiento y Modificadores de Acceso

- Objetivo: Practicar el uso de atributos privados y métodos públicos (getter y setter).
- Descripción: Modifica la clase Coche para que los atributos sean privados y crea los métodos getMarca(), setMarca(), getModelo(), setModelo(), etc. que permitan acceder y modificar los atributos.
- Conceptos: Encapsulamiento, modificadores de acceso.

Ejercicio 3: Constructores

- Objetivo: Utilizar constructores para inicializar objetos.
- Descripción: Agrega un constructor a la clase Coche que permita inicializar los atributos marca, modelo, y año al crear un nuevo objeto. Crea varios objetos utilizando diferentes parámetros en el constructor.
- Conceptos: Constructores, inicialización de objetos.

Ejercicio 4: Herencia

- Objetivo: Aplicar el concepto de herencia para extender una clase.
- Descripción: Crea una clase Vehículo con atributos comunes como velocidad y métodos como acelerar(). Luego, crea dos clases hijas: Coche y Bicicleta, que hereden de Vehículo. Añade atributos y métodos específicos para cada clase hija.
- Conceptos: Herencia, clases padre e hijo.

Ejercicio 5: Sobreescritura de Métodos

- Objetivo: Implementar la sobreescritura de métodos en clases derivadas.
- Descripción: En la clase Coche, sobreescribe el método acelerar() para que acelere de manera diferente a la clase Bicicleta. Muestra cómo se comporta cada clase con su propia versión de acelerar().
- Conceptos: Sobreescritura de métodos, polimorfismo.

Ejercicio 6: Polimorfismo

- Objetivo: Aplicar polimorfismo en el uso de clases y métodos.
- Descripción: Crea una lista de objetos que contenga tanto Coche como Bicicleta. Recorre la lista e invoca el método acelerar() en cada uno, mostrando cómo cada objeto utiliza su propia versión del método, aunque todos son del tipo Vehículo.
- Conceptos: Polimorfismo, herencia, listas de objetos.

Ejercicio 7: Clases Abstractas

- Objetivo: Entender la abstracción y las clases abstractas.
- Descripción: Crea una clase abstracta Animal con un método abstracto hacerSonido(). Luego, crea clases concretas como Perro y Gato que implementen el método hacerSonido(). Crea una lista de animales y llama al método hacerSonido() en cada uno.
- Conceptos: Clases abstractas, métodos abstractos.

Ejercicio 8: Interfaces

- Objetivo: Implementar el uso de interfaces para definir comportamiento común.
- Descripción: Define una interfaz Volador con un método volar(). Crea clases como Pájaro y Avión que implementen la interfaz Volador. Implementa el método volar() de manera diferente en cada clase.
- Conceptos: Interfaces, implementación de métodos.

Ejercicio 9: Composición

- Objetivo: Comprender cómo una clase puede estar compuesta por otras clases.

- Descripción: Crea una clase Motor con atributos como potencia y tipo. Luego, modifica la clase Coche para que contenga un objeto de la clase Motor. Implementa un método que describa el coche incluyendo los detalles del motor.
- Conceptos: Composición, clases dentro de clases.

Ejercicio 10: Manejo de Excepciones y Clases Personalizadas

- Objetivo: Introducir el manejo de excepciones y la creación de excepciones personalizadas.
- Descripción: Añade un método en la clase Coche llamado incrementarVelocidad(int velocidad), que aumente la velocidad. Si la velocidad es mayor que un valor permitido (por ejemplo, 200 km/h), lanza una excepción personalizada ExcesoVelocidadException. Implementa el manejo de esta excepción en el programa principal.
- Conceptos: Manejo de excepciones, clases de excepción personalizadas.