TRABAJO PRÁCTICO 7

Tema: Encapsulamiento de la información en Java.

Objetivos:

Que el estudiante logre…

- Crear clases aplicando la POO (abstracción y encapsulamiento) usando el lenguaje Java y el IDE NetBeans para organizar las clases en un proyecto.
- Aplicar el concepto de encapsulamiento en la definición de clases en lenguaje Java mediante el uso del especificador private en la definición de variables miembro de la clase.

Condiciones de desarrollo y autoevaluación del alumno.

- Este trabajo práctico debe realizarse en forma individual.
- La codificación debe llevarse a cabo de manera repetitiva y analítica tantas veces como sea necesario, revisando el significado de cada sentencia hasta que el alumno adquiera seguridad conceptual y práctica.
- La resolución completa de este trabajo práctico, incluyendo diagramas de flujo, deberá pasar a integrar la carpeta de práctica y autoevaluación del alumno.

Recursos Bibliográficos

- Programación en lenguaje Java. Ángel Esteban. Grupo Eidos.
- Java™ Platform Standard Ed. 7. https://docs.oracle.com/javase/8/docs/index.html
- Materiales disponibles en el aula virtual, tales como videos, lecciones, etc.

Materiales

- Video de revisión del proyecto Cuadrado.
- Bibliografía disponible en el CUV.
- IDE NetBeans o similar.

Tareas a desarrollar para cada enunciado

- ❖ Desarrolle la diagramación de flujo y la codificación en lenguaje Java de los siguientes enunciados.
- ❖ Para cada problema implemente una solución en lenguaje de programación Java aplicando POO.
- ❖ Identifique en cada caso las clases que se requieren, atributos y métodos.
- ❖ Construya un solo proyecto en el entorno NetBeans en el cual incluya la/s clase/s necesarias para resolver cada problema.
- ❖ La resolución de todos los enunciados deberá estar desarrollada en la carpeta de práctica de la asignatura y los programas en Java correspondientes a cada uno de los enunciados deberán ser completamente desarrollados por el alumno.

Notas a tener en cuenta

- ❖ Utilizar la clase Scanner para las entradas de datos.
- ❖ Aplicar encapsulamiento y modularidad.
- ❖ Todos los datos de entrada deben ser verificados, el programa no puede avanzar hasta que los datos de entrada sean correctos.

TRABAJO PRÁCTICO 7

Enunciado 1

**** Para completar esta tarea se recomienda seguir paso a paso el "Video de revisión del proyecto Cuadrado" que se encuentra en el aula virtual ****

- Diseñar una clase Cuadrado que modele un cuadrado utilizando el principio de encapsulamiento.
- La clase debe permitir calcular el perímetro a partir de la longitud de su lado.
- El proyecto estará formado por dos clases: Cuadrado y Principal.
- La ejecución del programa comenzará en la clase Principal.
- La clase Cuadrado debe contar con un atributo privado:
  `private int longLado;`
- Crear un constructor para inicializar la variable miembro en la clase Cuadrado.
  `public Cuadrado()` → `Inicializa el objeto de tipo Cuadrado asignando cero a longLado.`
- Incluir los siguientes métodos para representar el comportamiento del cuadrado.
  `public void cargarCuadrado()` → `Debe validar que el valor sea mayor que cero.`
  `public int calcularPerimetro()` → `Calcula y devuelve el perímetro del cuadrado, utilizando la fórmula 4 * lado.`
  `private void setLongLado(int longLado)` → `Convocado por cargarCuadrado, asigna el valor a la variable miembro.`
  `public int getLongLado()` → `Devuelve el valor del lado (método de acceso).`
- Ejemplo de uso (Clase Principal):

```
public static void main(String[] args) {
      Cuadrado c1 = new Cuadrado();
      c1.cargarCuadrado();

      Cuadrado c2 = new Cuadrado();
      c2.cargarCuadrado();

      Cuadrado c3 = new Cuadrado();
      c3.cargarCuadrado();

      System.out.println("Cálculo de los perímetros:");
      System.out.println(c1.calcularPerimetro());
      System.out.println(c2.calcularPerimetro());
      System.out.println(c3.calcularPerimetro());
}
```

- Salida esperada:

```
Cálculo de los perímetros:
16
80
44
```

Para esta salida se supone que el valor de los lados ingresados fue 4, 20, y 11.

Enunciado 2

- Crear un proyecto que estará formado por dos archivos de clases: la clase Rectangulo y la clase Principal.
- La clase Rectangulo debe modelar un rectángulo utilizando el principio de encapsulamiento. Los atributos deben ser privados y accederse mediante métodos públicos.
- La clase Rectangulo debe contar con los atributos privados que la describen:
  `private double base;`
  `private double altura;`
- Crear un constructor para inicializar esas variables miembro con valores iniciales (valores nulos).
  `public Rectangulo()`

- Incluir los siguientes métodos para representar el comportamiento del rectángulo.

```
public void cargarRectangulo()
public void mostrarRectangulo()
public double getBase()
public double getAltura()
private void setBase()
private void setAltura()
public double calcularArea()
public double calcularPerimetro()
```

- Desarrollar una clase Principal en la que se realice la creación de 2 instancias de la clase Rectangulo y se muestren sus áreas y perímetros.
- Los datos de los rectángulos deberán cargarse usando el método correspondiente. Toda entrada debe ser verificada adecuadamente. Estas tareas se realizan en clase Rectangulo, aplicando encapsulamiento.
- Ejemplo de uso:

```
Rectangulo r1 = new Rectangulo();
Rectangulo r2 = new Rectangulo();
r1.cargarRectangulo();
r2.cargarRectangulo();
r1.mostrarRectangulo();
r2.mostrarRectangulo();
```

- Salida esperada:

```
Rectángulo 1     Área: 10.0 ... Perímetro: 13.0
Rectángulo 2     Área: 10.5 ... Perímetro: 13.0
```

Enunciado 3 INVESTIGAR: Qué es sobrecarga de métodos?

Desarrolle una clase Numero, descripta mediante una variable miembro llamada valor, de tipo entero. Aplicar encapsulamiento definiendo a la variable valor como private en la clase Numero.

Incluir en su comportamiento tres métodos que cumplan las siguientes tareas:

- Sumar a la variable valor un entero recibido como parámetro.
- Sumar a la variable valor el valor de otro objeto Numero recibido como parámetro.
- Mostrar el valor del objeto Numero.

Desarrollar una clase Principal en la que se realice la creación de 2 instancias de la clase Numero, luego sumar y mostrar sus valores.

Ejemplos de ejecución

```
CASO UNO                                          CASO DOS
Entrada                                           Entrada
78                                                -7
25                                                -35
Salida                                            Salida
El resultado de la suma de los números es = 103   El resultado de la suma de los números es = -42
```

Enunciado 4

- Crear un proyecto que estará formado por dos archivos de clases: la clase Persona y la clase Principal.
- La clase Persona debe aplicar encapsulamiento para representar información básica y calcular la edad. Los atributos deben ser privados y accederse mediante métodos públicos.
- Atributos privados de la clase Persona:

```
private String nombre;
private int anioNacimiento;
```

- Crear un constructor para inicializar esas variables miembro con valores iniciales.

```
public Persona()
```

- Incluir los siguientes métodos para representar el comportamiento del objeto Persona.

```
public void cargarPersona()
public void mostrarPersona()
public int getEdad()
public int getAnioNacimiento()
private void setEdad()
private void setAnioNacimiento()
```

- Desarrollar una clase Principal en la que se realice la creación de 3 instancias de la clase Persona y se muestren sus datos, calculando la edad.
- Los datos de los objetos Persona deberán cargarse usando el método correspondiente. Toda entrada debe ser verificada adecuadamente. Se aplica encapsulamiento, los datos solo se acceden en clase Persona.
- Ejemplo de uso:

```
Persona p1 = new Persona();
Persona p2 = new Persona();
Persona p3 = new Persona();

p1.cargarPersona();
p2.cargarPersona();
p3.cargarPersona();

p1.mostrarPersona();
p2.mostrarPersona();
p3.mostrarPersona();
```

- Salida esperada:

```
Nombre: Ana ... Edad: 25 años
Nombre: Carlos ... Edad: 30 años
Nombre: Lucía ... Edad: 45 años
```

Enunciado 5

- Crear un proyecto que estará formado por dos archivos de clases: la clase Saludo y la clase Principal.
- La clase Saludo debe utilizar sobrecarga de métodos.
- Utilizaremos el constructor por default para crear objetos de tipo Saludo. Este constructor NO debe ser incluído en el código, Java lo crea automáticamente. [1]

```
public Saludo()
```

- Incluir los siguientes métodos para representar el comportamiento del objeto Saludo utilizando sobrecarga de métodos.

```
public void saludar()
public void saludar(String nombre)
public void saludar(String nombre, int edad)
```

- Ejemplo de uso:

```
Saludo s = new Saludo();
s.saludar();
s.saludar("María");
s.saludar("Pedro", 17);
s.saludar("Lucía", 25);
```

- Salida esperada:

```
¡Hola! Bienvenido al programa.
¡Hola María! Qué gusto verte.
Hola Pedro, eres menor de edad.
Hola Lucía, eres mayor de edad.
```

[1] *Podemos usar un método constructor default cada vez que los datos del objeto creado deban ser inicializados con valores básicos como cero o null.*

Enunciado 6

- Crear un proyecto que estará formado por dos archivos de clases: la clase Libro y la clase Principal.
- La clase Libro debe modelar un libro utilizando el principio de encapsulamiento. Los atributos deben ser privados y accederse mediante métodos públicos.
- Atributos privados de la clase Libro:

```
private String titulo;
private String autor;
private int anioPublicacion;
```

- Crear un constructor para inicializar esas variables miembro con valores iniciales.

```
public Libro()
```

- Incluir los siguientes métodos para representar el comportamiento asociado a un libro e identificar si el libro es antiguo. Un libro es antiguo si su año de publicación es anterior al año 2000.

```
public void cargarLibro()

public String getTitulo()
public String getAutor()
public int getAnioPublicacion()

private void setTitulo()
private void setAutor()
public void setAnioPublicacion()

public void mostrarLibro()
private boolean esAntiguo() → (devuelve true si el libro es anterior al año 2000)
public void muestraAntiguo() → (convoca a esAntiguo y traduce el booleano a una expresión
                                 String)
```

- Desarrollar una clase Principal en la que se realice la creación de 2 instancias de la clase Libro y se carguen sus datos para luego mostrarlos.
- Los datos de un libro deberán cargarse usando el método correspondiente. Toda entrada debe ser verificada adecuadamente. Aplicar encapsulamiento.
- Ejemplo de uso:

```
Libro r1 = new Libro();
Libro r2 = new Libro();

r1.cargarLibro();
r2.cargarLibro();

r1.mostrarLibro();
r2.mostrarLibro();
```

- Salida esperada:

```
Título: El Principito | Autor: Antoine de Saint-Exupéry | Año: 1943
¿Es antiguo? Si, es antiguo

Título: Harry Potter y la piedra filosofal | Autor: J.K. Rowling | Año: 1997
¿Es antiguo? Si, es antiguo
```