ITSSAT

LISTA DE COTEJO PARA INVESTIGACION

INTITUTO TECNOLOGICO SUPERIOR DE SAN ANDRES TUXTLA CARRERA: INGENIERIA INFORMÁTICA		
DATOS GENERALES DEL PROCESO DE EVALUACION		
Nombre(s) del alumno(s): PEREZ DOLORES ANGEL EMMANUEL		Firma del alumno(s):
Producto: Investigación Unidad I	Nombre del Proyecto: Investigación:Características de la programación orientada a objetos	Fecha:
Asignatura: Programación Avanzada	Grupo: 311- A	Semestre: Tercero
Nombre del Docente: MTI Lorenzo de Jesús Organista Oliveros		Firma del Docente:

INSTRUCCIONES				
Revisar las actividades que se solicitan y marque en los apartados "SI" cuando la evidencia se cumple; en caso contrario marque "NO". En la columna OBSERVACIONES indicaciones que puedan ayudar al alumno a saber cuáles son las condiciones no cumplidas, si fuese necesario.				
Valor del reactivo	*Características a cumplir (Reactivo)*	*CUMPLE*		*OBSERVACIONES*
		SI	*NO*	
1%	**Presentación.** El trabajo cumple con los requisitos de : a. Buena presentación	X		
1%	b. No tiene faltas de ortografía	X		
1%	c. Mismo formato (letra arial 14, títulos con negritas)	X		
1%	d. Misma calidad de hoja e impresión	X		
1%	e. Maneja el lenguaje técnico apropiado	X		
2%	**Introducción y Objetivo.** La introducción y el objetivo dan una idea clara del contenido del trabajo, motivando al lector a continuar con su lectura y revisión.	X		
5%	**Sustento Teórico.** Presenta un panorama general del tema a desarrollar y lo sustenta con referencias bibliográficas y cita correctamente a los autores.	X		
2%	**Desarrollo.** Sigue una metodología y sustenta todos los pasos que se realizaron al aplicar los conocimientos obtenidos, es analítico y bien ordenado.	X		
2%	**Resultados.** Cumplió totalmente con el objetivo esperado, tiene aplicaciones concretas.	X		
2%	**Conclusiones.** Las conclusiones son claras y acordes con el objetivo esperado.	X		
2%	**Responsabilidad.** Entregó el reporte en la fecha y hora señalada.	X		
20%	**CALIFICACIÓN:**	**20%**		

Instituto Tecnológico Superior de San Andrés Tuxtla

Ingeniería Mecatrónica

Grupo: 311-B.

Materia:
Programación Avanzada.

Investigación - Unidad I

Docente:
M.T.I. Lorenzo de Jesús Organista Oliveros.

Alumno:
Ángel Emmanuel Pérez Dolores

Sán Andrés Tuxtla Ver, a 19 de Septiembre del 2025.

Introducción.

La Programación Orientada a Objetos (POO) es uno de los paradigmas más utilizados en la informática moderna. Se originó como una alternativa a la programación estructurada, ofreciendo un modelo más cercano a la manera en la que los seres humanos entendemos el mundo: mediante objetos que poseen características y realizan acciones. Este paradigma no solo facilita la construcción de software complejo, sino que también promueve la reutilización, la escalabilidad y el mantenimiento del código.

Programación Orientada a Objetos (POO)

Es un modelo de programación en el que el diseño de software se organiza alrededor de datos u objetos, en vez de usar funciones y lógica.
Un objeto es una entidad que combina datos (atributos o propiedades) y comportamientos (métodos o funciones). Se puede definir como un campo de datos con atributos y comportamientos únicos.
La principal característica de este tipo de programación es que soporta objetos, que tienen un tipo o clase asociado. Esas clases pueden heredar atributos de una clase superior o superclase. Por esa razón, este enfoque de programación se utiliza en programas grandes y complejos que se deben actualizar con cierta regularidad. Esta forma de programar se inspira en la manera en que pensamos sobre el mundo real: los objetos tienen características y realizan acciones.

Características de la programación orientada a objetos

Las características de la Programación Orientada a Objetos (POO) son los principios fundamentales que permiten organizar, reutilizar y mantener el código de manera eficiente.

Usos de clases.

Una clase en POO funciona como una plantilla o un modelo que define la estructura y el comportamiento que serán comunes a un conjunto de objetos. Esta plantilla especifica qué atributos (datos) y qué métodos (funciones o comportamientos) deberán tener los objetos que se creen a partir de ella.
La clase determina las características esenciales de los objetos, como sus propiedades y capacidades, sin materializar ninguna entidad concreta. Por ejemplo, Por ejemplo, una clase “Coche” podría definir atributos como “marca”, “modelo”, y “color”, y métodos como “arrancar” y “detener”.

Objetos

Con la clase pueden crearse instancias de un objeto, es una **instancia concreta de una clase**, es decir, algo creado a partir del molde, Cada objeto creado de la misma clase puede tener características diferentes. Cada objeto creado desde una misma clase puede tener valores diferentes para sus atributos, pero todos compartirán la misma estructura y comportamientos definidos por la clase.

Ambos conceptos están relacionado ya que, la clase actúa como un molde, mientras que los objetos son las entidades reales creadas utilizando ese molde. Cada objeto mantiene su propio estado, pero todos ellos siguen las directrices establecidas por la clase de la que derivan.

4 principios de la Programación Orientada a Objetos

La programación orientada a objetos es más compleja de estructurar, es por eso que utiliza principios que pueden ayudar a escribir un código mejor, más limpio y reutilizable. Los principios de la Programación Orientada a Objetos (POO) son las bases que guían cómo se organiza y estructura el código en este paradigma. Generalmente se consideran cuatro principios fundamentales (conocidos como los pilares de la POO).

Encapsulación.

La encapsulación presenta toda la información importante de un objeto dentro del mismo y solo expone la información elegida al mundo exterior. Significa ocultar los detalles internos de un objeto y permitir el acceso solo mediante métodos específicos. Esta propiedad nos permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una clase las características o atributos que tienen un acceso privado, y los comportamientos o métodos que cuenta con un acceso público.
El único modo en la que esta se puede modificar es a través los propios métodos del objeto. De esta manera, los atributos internos de un objeto son ser inaccesibles desde fuera, pudiéndose modificar sólo llamando a las funciones correspondientes.

Abstracción.

Este principio consiste en simplificar la complejidad del mundo real modelando solo los aspectos relevantes para un propósito particular, mientras se ocultan los detalles innecesarios. La abstracción permite a los desarrolladores concentrarse en lo que es importante para la aplicación, sin necesidad de entender cada detalle del comportamiento o estado interno de los objetos. Esto reduce la complejidad y mejora la comprensibilidad del código.

Herencia.

a herencia permite que una clase derive o herede propiedades y comportamientos de otra clase, conocida como su clase padre. Esto facilita la reutilización del código y la extensión de las funcionalidades existentes sin tener que reescribir mucho código. Además, la herencia promueve la creación de una jerarquía de clases que puede reflejar relaciones naturales entre objetos más amplios y específicos, mejorando la organización del código. Mediante la definición en una clase de los atributos y comportamientos básicos, pueden crearse clases secundarias, ampliando la funcionalidad de la clase principal y añadiendo atributos y comportamientos extra. Es una de las claves de la Programación Orientada a Objetos.

Poliformismo

Se refiere a la capacidad de un objeto para tomar varias formas y comportarse de manera diferente en diferentes contextos, usando la misma interfaz. Esto se logra mediante el uso de la herencia y la sobreescritura de métodos, permitiendo que un mismo método tenga diferentes implementaciones en distintas clases. Gracias al polimorfismo permite tratar objetos de diferentes clases de manera uniforme.

Ventajas de la Programación Orientada a Objetos

Reutilización del Código:

Como se había mencionado, cuando se diseñan correctamente las clases, se pueden usar en distintas partes del programa y en diferentes proyectos. La técnica de herencia ahorra tiempo porque permite crear una clase genérica y luego definir las subclases que heredarán los rasgos de la misma, de manera que no es necesario escribir esas funciones de nuevo. Además, al aplicar un cambio en la clase, todas las subclases lo adoptarán automáticamente.

Modularidad.

Una de las características de la programación orientada a objetos más interesantes es la modularidad ya que así un equipo puede trabajar en múltiples objetos a la vez mientras se minimizan las posibilidades de que un programador duplique la funcionalidad de otro. El trabajo modular también permite dividir los problemas en partes más pequeñas que se pueden probar de manera independiente

Conclusión

La Programación Orientada a Objetos representa un avance significativo en la historia de la programación, al permitir un modelo más natural, escalable y reutilizable. Sus principios fundamentales, clases, objetos, encapsulamiento, abstracción, herencia y polimorfismo, han demostrado ser la base para el desarrollo de sistemas complejos en múltiples áreas de la tecnología. Por ello, la POO no solo es un paradigma vigente, sino una herramienta indispensable en la formación y práctica de todo programador actual.

Fuentes Consultadas.

Darias Perez, Sergio. (s.f.). Qué es la Programación Orientada a Objetos. [Página Web]. Intelequia. https://intelequia.com/es/blog/post/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos

Universidad Europea. (2022, 24 de Agosto). Programación orientada a objetos. [Página Web]. https://universidadeuropea.com/blog/programacion-orientada-objetos/

Valencia, Alonso. (2024, 9 de Mayo). ¿Qué es la Programación Orientada a Objetos (POO) y cuáles son sus principios fundamentales?. [Página Web]. CodersLink. https://coderslink.com/talento/blog/que-es-la-programacion-orientada-a-objetos-poo-y-cuales-son-sus-principios-fundamentales/

ITSSAT

GUIA DE OBSERVACIÓN PARA RESOLUCIÒN DE EJERCICIOS PRACTICOS

NOMBRE DE LA ASIGNATURA: Programación Avanzada

NOMBRE DE LA UNIDAD: Introducción

ALUMNO: PEREZ DOLORES ANGEL EMMANUEL

INSTRUCCIONES

Revisar los documentos o actividades que se solicitan y marque en los apartados "SI" cuando la evidencia a evaluar se cumple; en caso contrario marque "NO". En la columna "OBSERVACIONES" ocúpela cuando tenga que hacer comentarios referentes a lo observado.

Valor del reactivo	***Características a cumplir (Reactivo)***	***CUMPLE***		***OBSERVACIONES***
		Si	***NO***	
8%	¿Identifico el problema planteado?	X		
4%	¿Identifico los datos de entrada del problema?	X		
4%	¿Identifico los datos de salida del problema?	X		
8%	¿Generó la solución del problema en forma clara y comprensible (orden)?	X		
12%	¿Elaboró el programa respetando la sintaxis del lenguaje de programación (orden)?	X		
4%	Comprobó los resultados esperados a través de pruebas de escritorio?	X		
40%	***CALIFICACIÓN:***	**40%**		

```
/* PROGRAMACION AVANZADA
UNIDAD 1
ING.MECATRONICA. GRUPO:311-A
ALUMNO: PEREZ DOLORES ANGEL EMMANUEL
PROGRAMA 7
Utiliza vectores y metodos para el llenado de datos
*/



#include <iostream>
using namespace std;

int vec[10];

void LlenaVector()
{
for (int x = 0; x < 10 ; x++)
    {
        cout <<"TECLEA LOS NUMERO \n";
        cin >> vec[x];
    }
}

void ImprimeVector()
{
for (int x = 0; x < 10; x++)
    {
        cout << "Numero [" << x << "]" << vec[x] << "\n";
    }
}



int main()
{
LlenaVector();

ImprimeVector();

}
```

Título*

Evaluación - Unidad I

*Obligatorio

Instrucciones (opcional)

-Realizar un programa que cree 5 métodos, cada uno tendrá una función diferente [Los métodos serán indicados por el docente] con las siguientes indicaciones:

- El programa incluirá ciclo Do-While y Switch [Incluir el menú de los métodos] []

1.- Subir el código fuente [EvaluacionUI.cpp]

2.- Una imagen [captura de pantalla del funcionamiento del programa]

Adjuntar

Drive YouTube Crear Subir Enlace NotebookLM Gem

```
/*
EVALUACI�N UNIDAD 1

Creado por : ANGEL EMMANUEL PEREZ DOLORES
Programa que cree 5 m�todos, cada uno tendr� una funci�n diferentes [Los metodos
ser�n indicados por el docente] con las siguientes indicaciones: El programa
incluir� ciclo Do-While y Switch
*/

#include <iostream>
using namespace std;

int vec[10];

void rectangulo()
{
      cout<<"* * * * * * * *\n";
      cout<<"*             *\n";
      cout<<"*             *\n";
      cout<<"*             *\n";
      cout<<"* * * * * * * *\n";
}


void parimpar()
{
      int a,b;
      cout<<"ingrese un numero\n";
      cin>>a;
      b=a%2;
      if (b==0){
            cout<< "numero es par\n";
      }
      else{
            cout<< "numero es impar\n";
      }
}

void tabla()
{
      int num;
      cout<<"que tabla de multiplicar deseas";
      cin>>num;
      for(int x=0; x<=10;x++)
      {
            cout<<x<<"*"<<num<<"= "<<num*x<<endl;
      }
}

void promedio()
{
      int num, calif, acumulador=0;
      cout<<"Numero de unidades de la materia:\n";
      cin>>num;
      for(int x=1; x<=num;x++)
      {
            cout<<"teclee la calificacion"<<x<<"=";
            cin>>calif;
            if((calif>=0)&&(calif<=100))
```

```
            {
                  acumulador=acumulador+calif;
            }
            else
            {
                  cout<<"calificacion NO VALIDA\n";
                  x--;
            }
      }
      cout<<"promedio es:"<<(acumulador/num)<<endl;
      if(((acumulador/num)>=70)&&((acumulador/num)<=100))
      {
            cout<<"APROBADO\n";
      }
      else
      {
            cout<<" No aprobado\n";
      }
}

void cinconum()
{
      int num, contador=0;
      do
      {
            cout<<"teclee un numero";
            cin>> num;
            if(num>=1000)
            {
                  contador++;
            }

      }while(contador<5);
}

void llenarvector()
{
      for(int x=0;x<10;x++){
            cout<<"teclee valor["<<x<<"]->";
            cin>>vec[x];
      }
}

void imprimevector()
{
            for(int x=0;x<10;x++){
            cout<<"valor: "<<vec[x]<<"\n";
      }
}

int main()
{
      int OP;
      char resp;
      do
      {
            cout<<"*****************************************\n";
            cout<<"*                  Menu                 *\n";
            cout<<"*                                       *\n";
```

```
            cout<<"*    1.- presente un rectangulo          *\n";
            cout<<"*                                        *\n";
            cout<<"*    2.-      Par/Impar                  *\n";
            cout<<"*                                        *\n";
            cout<<"*    3.-  Tabla de multiplicar          *\n";
            cout<<"*                                        *\n";
            cout<<"*    4.-Promedio(aprobado/reprobado)     *\n";
            cout<<"*                                        *\n";
            cout<<"*    5.-  Imprimir solo 5 numeros        *\n";
            cout<<"*                                        *\n";
            cout<<"*     6.- Salir                          *\n";
            cout<<"*                                        *\n";
            cout<<"******************************************\n";

            cout<<"Elija una opcion  ";
            cin>>OP;
            switch(OP)
            {
                  case 1:
                        {
                              cout<<"Eligio presentar un rectangulo\n";
                              rectangulo();
                              break;
                        }
                  case 2:
                        {
                              cout<<"Eligio Par/Impar\n";
                              parimpar();
                              break;
                        }
                  case 3:
                        {
                              cout<<"Eligio Tabla de multiplicar\n";
                              tabla();
                              break;
                        }
                  case 4:
                        {
                              cout<<"Eligio Promedio\n";
                              promedio();
                              break;
                        }
                  case 5:
                        {
                              cout<<"Eligio Imprimir 5 Numeros\n";
                              cinconum();
                              break;
                        }
                  case 6:
                        {
                              cout<<"eligio salir\n";
                              break;
                        }
                  default:
                        {
                              cout<<"No existe esa opci□n--elija una opci□n del
menu"<<endl;
                        }
            }
```

```
        }while(OP!=6);


}
```