TEMÁTICA DE LAS UNIDADES DE LOGICA COMPUTACIONAL

UNIDAD I
ELEMENTOS DE PROGRAMACIÓN

1. Solución de problemas con programación
   - 1.1 Concepto de programación
   - 1.2 Concepto de problema
2. Concepto de algoritmo
   - 2.1 Elementos de un algoritmo
   - 2.2 Características de un algoritmo
   - 2.3 Ejemplos y ejercicios
3. Tipos de datos
   - 3.1 Concepto de variable
   - 3.2 Concepto de constante
   - 3.3 Ejemplos y ejercicios
4. Diagrama de flujo
   - 4.1 Prueba de escritorio
   - 4.2 Ejemplos y ejercicios
5. Pseudo código
6. Metodología de programación (lineal, estructurada, eventos y orientada a objetos)
7. Estructuras de control de flujo programático
8. Programación Orientada a Objetos
   - 8.1 Características
   - 8.2 Elementos

UNIDAD II
VISUAL BASIC .NET Y LA PLATAFORMA .NET FRAMEWORK

1. .Net FrameWork
2. Visual Studio . NET
3. Visual Basic . NET
   - 3.1 Ambiente
   - 3.2 Aplicaciones de consola
   - 3.3 Formularios
   - 3.4 Eventos
   - 3.5 Depurador

UNIDAD III
APLICACIONES WINDOWS CON VISUAL BASIC .NET

1. Conversiones de tipos de datos
2. Arreglos y Estructuras
3. Módulos
   - 3.1 Funciones
   - 3.2 Subs
4. Ventanas
5. Validaciones
6. Clases

UNIDAD IV
PROYECTOS DE INSTALACIÓN

1. Distribución de Aplicaciones.
2. Distribuir el Common Lenguaje Runtime
3. Proyecto de instalación
4. Asistente de Proyectos de instalación

INTENCIÓN.

Qué el estudiante genere aplicaciones Windows en Visual Basic . NET, mediante la aplicación de la metodología de programación orientada a objetos, el uso de las librerías del FrameWork, las estructuras de datos, módulos, herramientas de programación y distribución de aplicaciones; con la intención de generar soluciones a problemas informáticos con programación.

UNIDAD I

ELEMENTOS DE PROGRAMACIÓN

1. Solución de problemas con programación
   1.1 **Concepto de programación** : Es un método para la solución de problemas.
   1.2 **Concepto de problema**: Es una situación que surge a raíz de la necesidad de algo.

2. Concepto de algoritmo: Es una serie de pasos lógicos que resuelven un problema.
   2.1 **Elementos de un algoritmo**:

Entrada de datos: Acción u operación que permite el ingreso de los datos del problema.

Proceso: Operación u operaciones secuenciales, lógica y organizadas, cuyo objetivo es obtener la solución al problema.

Salida de datos: Operación o conjunto de operaciones que permiten comunicar al exterior los resultados alcanzados.

2.2 **Características de un algoritmo**:

Finito: En esta característica un algoritmo siempre debe tener un fin sin importar si es simple o complejo.
Determínistico: En esta característica, dados un conjunto de datos de entrada, deberán arrojar los mismos resultados siempre
Verificación del algoritmo: En esta etapa se ejecuta y se valida el algoritmo o el programa por la computadora al exterior los resultados alcanzados.

2.3 Ejemplos y ejercicios: QUE PASOS TENGO QUE HACER PARA HACER UN PASTEL, QUE PASOS TENGO QUE HACER PARA HACER UNAAGUA DE LIMÓN

3. **Tipos de datos**

- **Entero (Integer):** Son valores enteros como por ejemplo: 1, 5 10, 50, etc
- **Booleano (booleano):** Son valores lógicos como por ejemplo: Si/No, S/N, 0/1, Encendido/apagado, Falso/Verdadero, True/False).
- **Flotantes (Float):** Son números con punto decimal como por ejemplo: 1.5, 5.8 10.50, etc.
- **Carácter (Char):** Es la unidad mínima de información (dato)

3.1 Concepto de variable: Es un espacio de memoria que sirve para almacenar un cierto tipo de dato que puede variar con el tiempo, el cual tendrá un nombre representativo de lo que contiene como:

VARIABLES

Nombre de la Variable (identificador)	Valor Almacenado	Tipo de Dato
area	57.92	Real
Resultado	12	Entero
Dirección	'Calzada de los Muertos'	Cadena de Caracteres
RFC	'COSA120519'	Cadena de Caracteres
Verdadero	True	Lógico
Opción	's'	Carácter

3.2 Concepto de constante: Es un espacio de memoria que sirve para almacenar un cierto tipo de dato y este valor no cambia, se mantiene, el cual tendrá un nombre representativo de lo que contiene como: PI, una formula como: area =base * altura /2

CONSTANTES

Nombre de la Constante (Identificador)	Valor Almacenado	Tipo de Dato
Letra	'A'	Carácter
Suma	53	Entero
Promedio	9.83	Real
Sexo	'F'	Carácter
Nombre	'Juan Pérez'	Cadena de Caracteres
Bandera	True	Lógico
PI	3.1416	Real

Operadores Aritméticos

OPERADOR ARITMÉTICO	OPERACIÓN
**	Exponenciación
*	Multiplicación
/	División
+	Suma
-	Resta
Mod	Módulo (residuo)
Div	División entera

Reglas de Prioridad

Operación	Operador	Jerarquía
Exponenciación	**	1
Multiplicación División	* /	2
Modulo División entera	Mod Div	3
Suma Resta	+ -	4

Operadores Relaciónales

OPERADOR RELACIONAL	OPERACIÓN
=	Igual qué
< >	Diferente a
>	Mayor qué
<	Menor qué
>=	Mayor o igual qué
<=	Menor o igual qué

Operadores Lógicos

OPERADOR LÓGICO	OPERACIÓN
NO (NOT)	Negación
Y (AND)	Conjunción
O (OR)	Disyunción

TABLAS DE VERDAD

OPERADOR NOT

A	NOT A
Verdad	Falso
Falso	Verdad

OPERADOR OR

A	B	A OR B
Verdad	Verdad	Verdad
Verdad	Falso	Verdad
Falso	Verdad	Verdad
Falso	Falso	Falso

OPERADOR AND

A	B	A AND B
Verdad	Verdad	Verdad
Verdad	Falso	Falso
Falso	Verdad	Falso
Falso	Falso	Falso

EJERCICIOS PARA LAS EXPRESIONES.

Obtén el resultado de las siguientes expresiones:

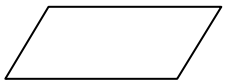
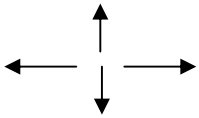
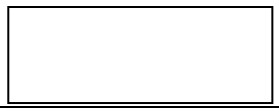
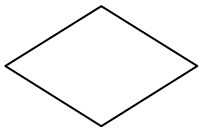
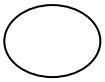
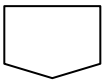
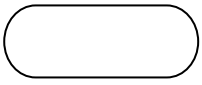
a) 23 – 45 * 6 – 7
b) (34 – 32) – 89
c) 45 ** 2 – 45 + (34 – 9)
d) 8 * 3 ** 3 div 6
e) 36 + 4 – (24 mod 6)
f) Si el valor de a =2, b =3 y c =4, evaluar la expresión: (a ** b) * c
g) 5 * (4 + 15)
h) (4 * (3 + 12 – 8) / 2) * (15 + 10)

De las siguientes expresiones booleanas son legales si x, y, z son de tipo real y bandera de tipo booleano. El valor de cada expresión, es mostrado entre paréntesis y asuma que x=3.0, y=4.0, z=2.0 y bandera = falso.

(x > z) and (y > z)
(x + y / z) <= 3.5
(z > x) or (z > y)
not bandera
(x = 1.0) or (x = 3.0)
(z < x) and (x < y)
(x <= z) or (z >= y)
(not bandera) or ((y + z) >= (x – z))

4. **Diagrama de flujo:** Es la representación gráfica de un algoritmo.

SIMBOLOGÍA BÁSICA

OPERACIÓN	SÍMBOLO	UTILIDAD
Entrada / Salida		Representa la entrada ó salida de los datos.
Líneas de Flujo		Representa el flujo de la información entre los símbolos.
Proceso		Indica una operación.
Decisión		Representa un punto de decisión Falso /Verdadero dentro del diagrama.
Conector		Conector dentro de una misma página.
Conector		Conector entre diferentes páginas.
Subrutina ó Submódulo		Submódulo que realiza operaciones y al terminar regresa al menú principal.
Documento		Representa un documento ó una impresión.
Inicio / Termino		Identifica el inicio y termino del diagrama.

REGLAS PARA LA CONSTRUCCIÓN DE UN DIAGRAMA DE FLUJO

1. El símbolo de inicio y termino deberá aparecer una sola vez.
2. El flujo de las operaciones será de arriba hacia abajo y de izquierda a derecha.
3. Se deben de evitar los cruces de línea con la ayuda de los conectores bien identificados.
4. Las líneas de flujo que se utilicen deben estar conectadas a un símbolo y deben ser rectas verticales y horizontales.
5. No debe de llegar más de una línea a un símbolo.

4.1 Prueba de escritorio: LA PRUEBA DE ESCRITORIO ES UN SEGUIR LOS PASOS DEL PSEUDOCODIGO Ó EL PROGRAMA Y PONERLE VALORES VERDADEROS PARA COMPROBAR SI ES CORRECTO EL PSEUDOCODIGO Y SE EMPIEZA DESDE LA LECTURA DE DATOS!!!!

4.2 EJEMPLOS:

- Realizar el pseudocódigo, programa y prueba de escritorio del siguiente problema:

Realizar la suma de 3 numeros

PSEUDOCODIGO	PROGRAMA	PRUEBA DE ESCRITORIO
• Inicio • Se llama SUMA (ya que hace la suma de tres números enteros) • numero1, numero2, numero3, suma: entero • dame el primer numero • dame el segundo numero • dame el tercer numero • suma = numero1 + numero2 + numero3 • escribe suma • Termina ó Fin	• Inicio • Se llama SUMA (ya que hace la suma de tres números enteros) • numero1, numero2, numero3, suma: integer • Read el primer numero • Read el segundo numero • Read el tercer numero • suma = numero1 + numero2 + numero3 • write suma • End.	numero1 = 5 numero2 = 10 numero3 = 15 suma = numero1 + numero2 + numero3 suma = 5 + 10 + 15 = 30 Escribe (lo que vale suma) suma =30

5. **Pseudo código:** Es la descripción de un algoritmo utilizando palabras en ingles ó español antes de traducirlas a un lenguaje de programación, el pseudo código le permite al programador analizar la lógica del programa y corregir, si existe algún error.

TABLA DE COMPARACIÓN

ALGORITMO	PSEUDOCODIGO
• INICIO ó COMIENZA • ENTRADA DE DATOS • PROCESO • SALIDA DE DATOS • FIN ó TERMINO	• INICIO ó COMIENZA • NOMBRE DEL PSEUDOCODIGO (LO QUE HACE EL PROROGRMA) • DECLARACIÓN DE VARIABLES • LECTURA Ó ENTRADA DE DATOS. • PROCESOS • ESCRIBE ó SALIDA DE DATOS • TERMINA

- Realizar el pseudocódigo, programa y prueba de escritorio del siguiente problema:

1/1 + 1/3 + 1/5 + 1/7 + 1/9 + 1/11 + 1/13

PSEUDOCODIGO	PROGRAMA	PRUEBA DE ESCRITORIO
• Inicio • Se llama SUMA_DE_DIV (ya que hace la suma varias divisiones) • numero1, numero2, numero3, numero4, numero5, numero6, numero7: entero • div1, div2, div3, div4, div5, div6, div7, suma: decimal • dame el primer numero • dame el segundo numero • dame el tercer numero • dame el cuarto	• Inicio • Se llama SUMA_DE_DIV (ya que hace la suma varias divisiones) • numero1, numero2, numero3, numero4, numero5, numero6, numero7: integer • div1, div2, div3, div4, div5, div6, div7, suma: float • Read el primer numero • Read el segundo numero • Read el tercer numero • Read el cuarto numero • Read el quinto numero • Read el sexto numero • Read el séptimo numero • div1 = numero1/ numero1 • div2 = numero1/ numero2 • div3 = numero1/ numero3 • div4 = numero1/ numero4 • div5 = numero1/ numero5 • div6 = numero1/ numero6	numero1 = 1 numero2 = 3 numero3 = 5 numero4 = 7 numero5 = 9 numero6 = 11 numero7 = 13 div1 = numero1/ numero1 = 1/1 div2 = numero1/ numero2 = 1/3 div3 = numero1/ numero3 = 1/5 div4 = numero1/ numero4 = 1/7 div5 = numero1/ numero5 = 1/9 div6 = numero1/ numero6 = 1/11 div7 = numero1/ numero7 = 1/13 suma = div1 + div2 + div3 +div4 + div5 + div6 + div7numero1 + numero2 + numero3 suma = 1 + 0.333 + 0.2 + 0.1428 + 0.1111 + 0.0909 + 0.0769

numero • dame el quinto numero • dame el sexto numero • dame el septimo numero • div1 = numero1/ numero1 • div2 = numero1/ numero2 • div3 = numero1/ numero3 • div4 = numero1/ numero4 • div5 = numero1/ numero5 • div6 = numero1/ numero6 • div7 = numero1/ numero7 • suma = div1 + div2 + div3 +div4 + div5 + div6 + div7 • escribe suma • Termina ó Fin	• div7 = numero1/ numero7 • suma = div1 + div2 + div3 +div4 + div5 + div6 + div7 • write suma • End.	suma = 1.9601 write suma =1.9601 End.

TAREA: Realizar el pseudo código, programa y prueba de escritorio de los siguientes ejercicios.

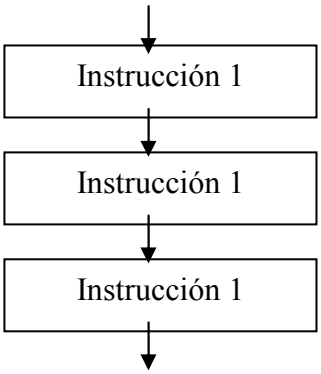
- ❖ Realizar las 4 operaciones básicas (suma, resta, multiplicación y división) con dos números decimales
- ❖ Realizar la suma de **1 * 1 + 2 * 2 + 3 * 3 + 4 *4 + 5 * 5 + 6 * 6 + 7 * 7 + 8* 8**
- ❖ Sacar el área de un triangulo
- ❖ Sacar la conversión de temperaturas de grados fareget a centígrados y de centígrados a farenget.

6. Metodología de programación (lineal, estructurada, eventos y orientada a objetos)

7. **Estructuras de control de flujo programático**

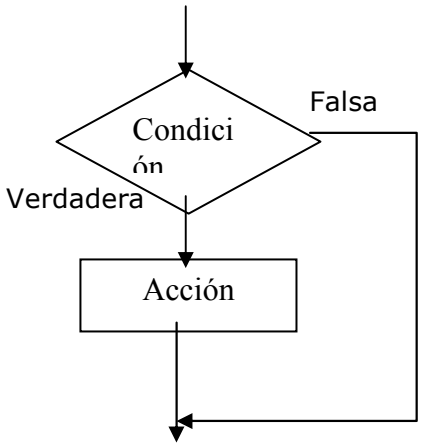
ESTRUCTURAS ALGORITMICAS

ESTRUCTURAS SECUENCIALES. Es aquella en que una instrucción sigue a la otra en secuencia. Su estructura es la siguiente:

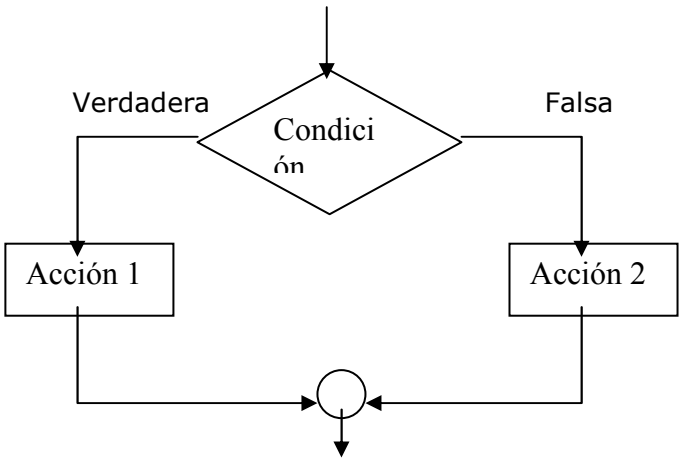


ESTRUCTURAS SELECTIVAS. Se utilizan para la toma de decisiones lógicas en los programas y son llamadas también Estructuras de decisión ó Alternativas. Este tipo de estructura evalúa una "condición" y en función del resultado que puede ser *verdadero o falso* se realiza una acción o secuencia de acciones. Existen tres tipos de estructuras alternativas:

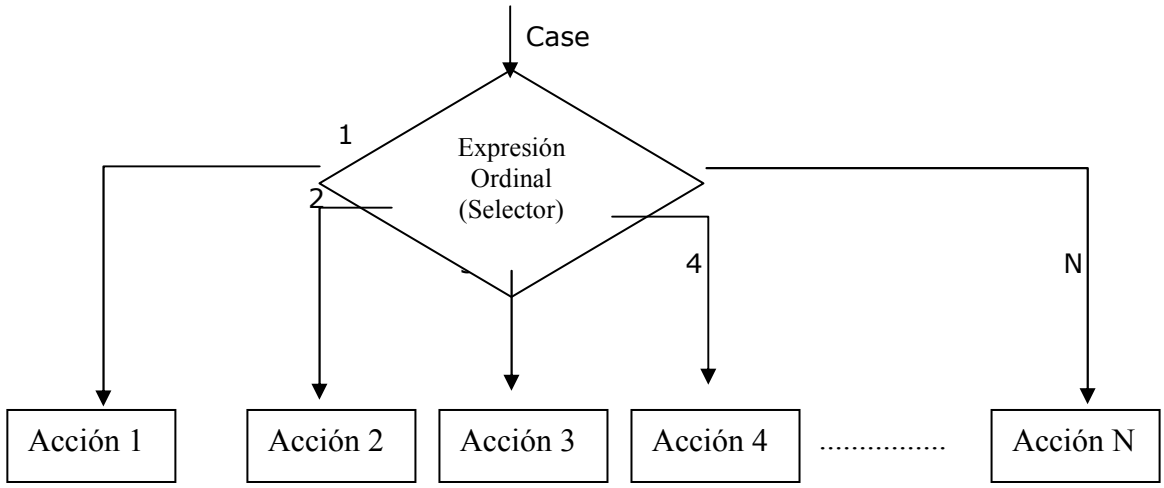
Alternativa Simple. Esta estructura también llamada declaración de bifurcación por que nos permite tomar cualquiera de dos caminos. Esta estructura funciona de la siguiente manera: Se evalúa la condición (expresión booleana) esto produce un valor de verdadero (true) o falso (false). Si la condición es verdadera se ejecutará la acción 1 (instrucciones) o acciones, si la condición es falsa no se ejecutará nada.



Alternativa Doble. Esta estructura también llamada declaración de bifurcación por que nos permite tomar cualquiera de dos caminos. Esta estructura funciona de la siguiente manera: Se evalúa la condición (expresión booleana) esto produce un valor de verdadero (true) o falso (false). Si la condición es verdadera se ejecutará la acción 1 (instrucciones) o acciones, si la condición es falsa no se ejecutará la acción2

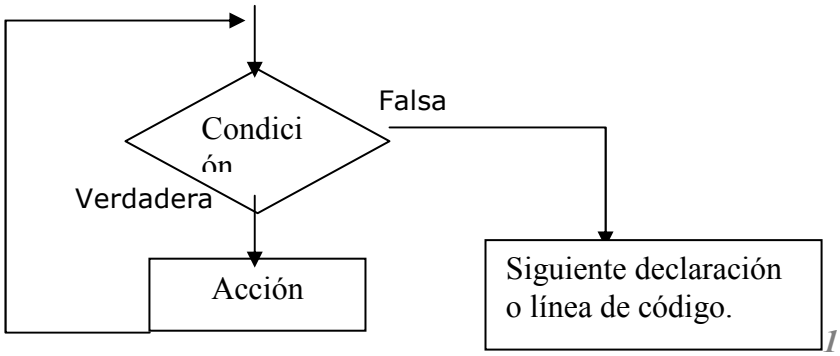


Alternativa Múltiple Esta estructura utiliza una expresión ordinal o selector. El selector podrá tomar solamente un valor de tipo ordinal(es decir no puede ser real). El valor que tome el selector determina la acción que se va a ejecutar.

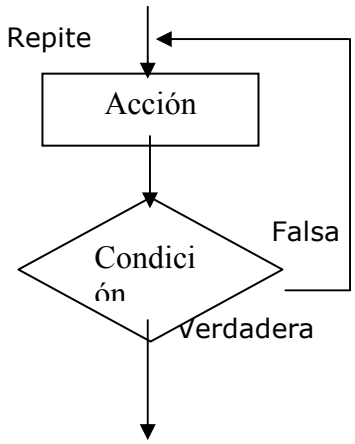


ESTRUCTURAS REPETITIVAS. Es aquella estructura que repite un determinado número de veces una secuencia de instrucciones hasta que cierta condición se cumpla. Existen tres tipos de estructuras repetitivas.

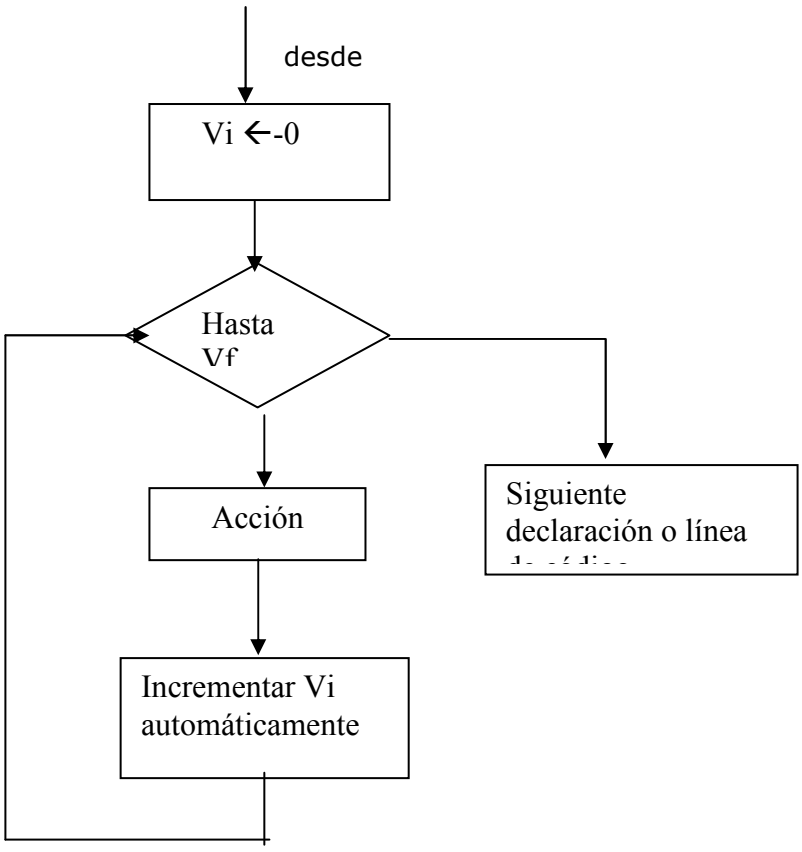
MIENTRAS - HACER. Esta estructura permite la ejecución de un bloque de código del programa en forma repetitiva mientras que la condición se cumpla(sea verdadera). El ciclo termina cuando la condición cambia a falso.



REPITE – HASTA <Condición>. Está estructura funciona de la siguiente manera: la acción o acciones se ejecutan hasta que se cumpla una condición determinada que se comprueba hasta el final del ciclo, como se nota en la estructura se ejecuta al menos una vez ya que la condición se prueba hasta el final.



DESDE / PARA. Esta estructura ejecuta la acción o acciones del cuerpo del ciclo un número determinado de veces, se puede decir que es un contador automática ya que controla el número de iteraciones. Esta estructura se utiliza cuando se conoce el número de veces en que una acción o acciones se repetirán. La estructura inicia desde un valor inicial (Vi) y la acción o acciones se ejecutan si el valor inicial es menor que el valor final (Vf), y termina cuando el valor inicial es mayor que el valor final.



Vi : valor inicial del ciclo (se le asigna un número entero positivo) y en cada iteración se va incrementando su valor.
Vf : valor final determina el número de veces que se ejecutará el ciclo.

PROGRAMACION ORIENTADA A OBJETOS (POO)

Es la programación que hace que un objeto responda a cierta acción ya sea de usuario o de interacción. Mediante la incorporación de procedimientos (bloque de código) y manipulación de algunas de sus propiedades y su vinculación con las bd con la finalidad de hacer interfaces en ambiente gráfico

CLASE

Es un molde o prototipo que define un tipo de objeto determinado. Una clase define los atributos y métodos que va a poseer un objeto

HERENCIA

Es un mecanismo mediante el cual podemos reutilizar clases ya definidas. Dentro de la POO es un mecanismo fundamental que se puede definir también como una transmisión de las características de padres a hijos. Podemos heredar un método de la clase padre pero en la hija la podemos dar una implementación diferente para que se adecue a la nueva clase.

OBJETO

Son los bloques de construcción basados en aplicaciones de elementos programados que puedes usar para crear una aplicación.

SUS CARACTERISTICAS SON:

ESTADO O ATRIBUTO
Viene definido por una serie de parámetros que lo definen y diferencian de objetos del mismo tipo.

COMPORTAMIENTO O MÉTODO
Viene definido por las acciones que pueden realizar los objetos permitiendo distinguir a objetos de distinto tipo.

ENCAPSULAMIENTO
Es el hecho de ocultar la implementación interna de un objeto, es decir, como está construido y de que se compone. Presenta varios beneficios, entre los que destacan la modularidad y la ocultación de la información.

ABSTRACCIÓN

Capacidad de ignorar determinados aspectos de la realidad con el fin de facilitar la realización de una tarea. También nos permite ignorar los aspectos de implementación de los objetos en los pasos iniciales, con lo cual sólo necesitamos conocer qué es lo que hace un objeto, y no cómo lo hace, para definir un objeto y establecer las relaciones de éste con otros objetos.

POLIMORFISMO

Podemos definir dentro de dos clases distintas dos operaciones con el mismo nombre y aspecto externo, pero con distintas implementaciones para cada clase.

SOBRECARGA

Se produce cuando una clase tiene métodos con el mismo nombre pero que difieren o bien en el número o en el tipo de los parámetros que reciben dichos métodos.