ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

NOMBRE: GABRIEL GUDUPA **FECHA:** 13-NOVIEMBRE-024

NRC: 1323

PROFESOR: MGTR. LUIS ENRIQUE JARAMILLO MONTAÑO

PROGRAMACIÓN ORIENTADA A OBJETOS

1.- Qué es paradigma de la programación orientada a objetos

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. En programación se conoce como paradigmas de programación a los métodos usados para realizar determinadas tareas o proyectos.
En este sentido podemos concluir que son métodos de programación de software que sirven para resolver un problema o sistemas o para llegar a los resultados esperados.

2. Qué es CLASE, OBJETO, MÉTODO Y ATRIBUTO.

Los componentes fundamentales de un programa codificado con programación orientada a objetos son:

Clase.- Son tipos de datos definidos por el usuario. Es donde creamos un modelo para la estructura de métodos y atributos. Los objetos individuales se crean como instancia a partir de la clase. La clase contiene campos para atributos y métodos para comportamientos.

Objeto.- Son instancias de una clase creada con datos específicos.

Método.- Representan comportamientos. Los métodos realizan acciones y pueden devolver información sobre un objeto o actualizar los datos de un objeto. El código del método se define en la definición de clase.

Atributos.- Son la información que se almacena. Los atributos se definen en la plantilla Clase. Cuando se crean instancias de objetos; los objetos individuales contienen datos almacenados en el campo Atributos.

El estado de un objeto está definido por los datos en los campos de atributos del objeto.

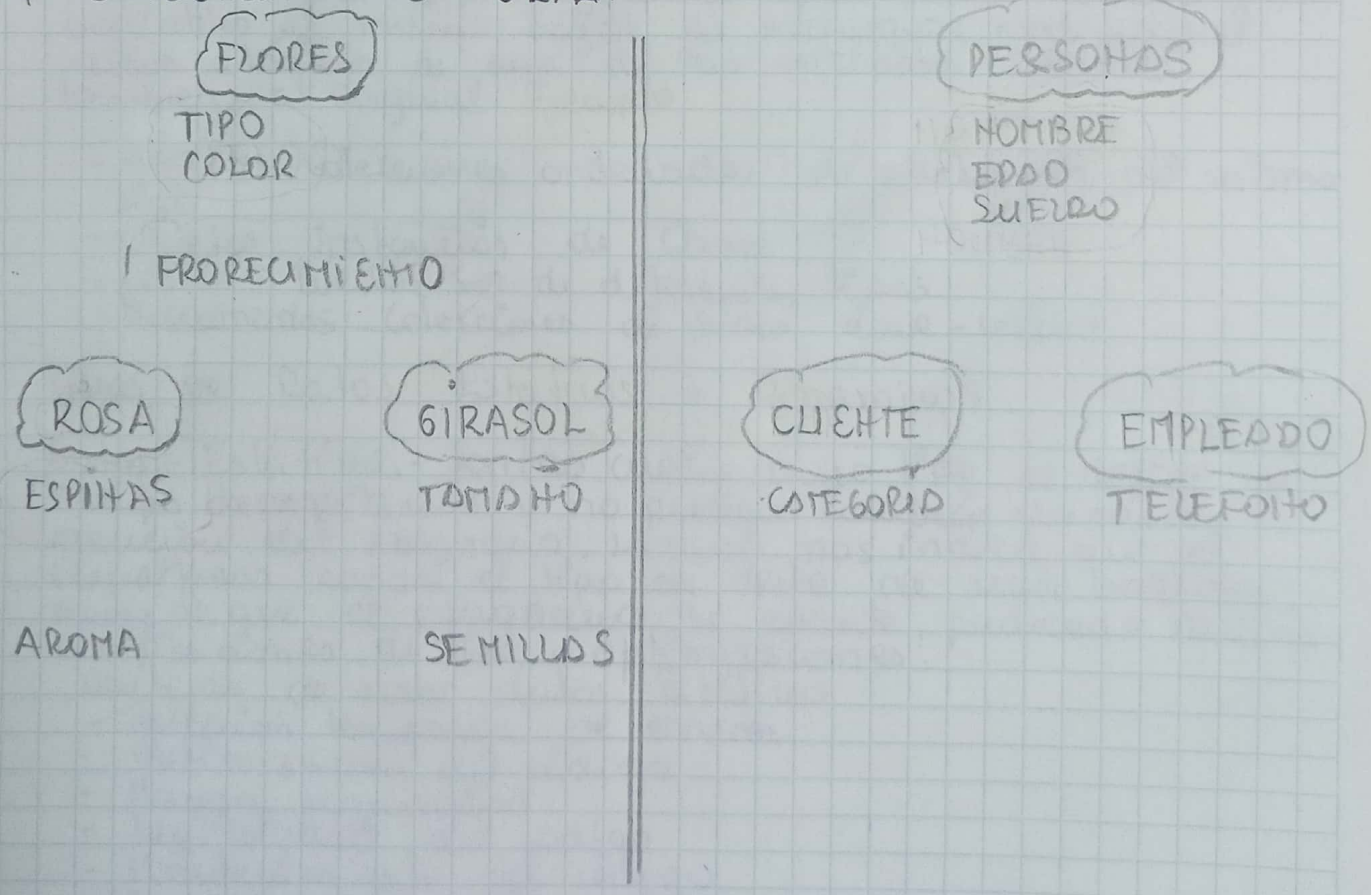
3 Qué es un sistema de controlamiento y para que sirve.

Un sistema de controlamiento o sistema de control se refiere a los mecanismos y estructuras que permiten supervisar, gestionar y regular el comportamiento de un sistema o aplicación. Puede asociarse con componentes como controladores, manejadores de eventos y sistemas de flujo de datos que gestionan cómo se procesa y dirige la lógica dentro del programa.

Estos sistemas de control sirven para:

- Organizar el flujo de datos y acciones
- Facilitan el mantenimiento del código
- Implementan patrones de diseño (Modelo - Vista - Controlador)
- Aumentan la estabilidad.
- Monitoreo - regulación - optimización - automatización - prevención de errores y coordinación.

4 Elaboran dos UML.



Tipos de Datos Primitivos y Referenciados

Datos Primitivos.- Este tipo de datos presentan valores simples y se almacenan directamente en la memoria. Su tamaño es fijo y se copian por valor. Esto significa que al copiar una variable primitiva, se crea una copia independiente del valor original, modificar la copia no afecta al original. Ejemplo:

- Enteros (int): Números enteros (-2, 0, 10, 1000). La cantidad de bits que ocupa depende del lenguaje de programación
- Números de punto flotante (float, double): Números con decimales (3.14, 3.25, -2.5)
- Booleanos (bool): Representan valores de verdad, true o false
- Caracteres (char): Representan un solo caracter ('a', 'A', '5', '!')
- Cadenas de caracteres (String, en algunos lenguajes) Secuencia de caracteres ("Hola mundo")

Datos Referenciales.- Este tipo de datos no almacenan directamente el valor en la memoria, sino una referencia a la ubicación de memoria donde se almacena el valor. Su tamaño es variable y se copian por referencia. Esto significa que al copiar una variable referenciada, se crea una copia de la referencia apuntando ambas variables al mismo lugar en memoria; modificar el valor, a través de una de las variables afectará también al original Ejemplo.

- Arrays: Colecciones ordenadas de elementos del mismo tipo
- Objetos: Instancias de Clases
- Listas: Pueden ser de diferentes tipos.
- Diccionarios: Colecciones de pares clave-valor.

Tipos de Datos Estáticos y Dinámicos

Datos Estáticos.- Son los cuales cuyo tipo se define en tiempo de compilación y no pueden cambiar durante la ejecución del programa. Lo cual nos indica que el compilador conoce el tipo de dato de una variable antes de que el programa se ejecute, pudiendo realizar verificaciones de tipo y optimizaciones.

Ventajas de usar datos Estáticos.

- Detección temprana de errores
- Optimización del código
- Mayor seguridad
- Legibilidad del código
- Mantenimiento del código
- Seguridad y eficiencia.

Datos Dinámicos :- Son los que se determinan en tiempo de ejecución, en lugar de en tiempo de compilación como ocurre con los tipos de datos estáticos. Esto significa que una variable puede contener valores de diferentes tipos durante la ejecución del programa sin necesidad de declararlos previamente.

Ventajas de los datos dinámicos:

- Flexibilidad
- Rapidez de desarrollo
- Prototipado rapido.

Tipos de datos Primitivos y Referenciales

DATOS PRIMITIVOS

```
public class TiposPrimitivos {
    public static void main (String[] args) {
    // Declaración de tipos de datos primitivos

    byte b = 10;
    short s = 150;
    int i = 10000;
    long l = 10 0000L; // La 'L' indica que es un long
    float f = 5,76f; // La 'f' inica que es un float
    double d = 16.30
    char c = 'A'
    boolean bool = true;

    // Impresión de los tipos de datos
    System.out.println("Byte:" + b);
    System.out.println("Short:" + s);
    System.out.println("int:" + i);
    System.out.println("Long:" + l);
    System.out.println("float:" + f);
    System.out.println("Double:" + d);
    System.out.println("Char:" + c);
    System.out.println("Boolean:" + bool);

    }
}
```

DATOS REFERENCIALES

1 Clases.

```
class Persona
  String nombre ;

  Persona (String nombre) {
    this. nombre = nombre ;
  }
}
  public class Main {
    Public static void main (String[ ] args ) {
        // Creando un objeto de la clase Persona

      Persona persona = new Persona ("Juan");
        // Imprimiendo el nombre de la persona

      System.out.println ("Hombre: "+ persona. nombre);
    }
}
```

2 Arrays

```
Public class Main {
  public static void main (String[ ] args ) {
      // Creando un array de enteros
      int [ ] numeros = { 1, 2, 3, 4, 5 };

      // Imprimiendo los valores del array
      for (int numero : numeros ) {

      System.out.println (numero);
    }
  }
}
```

3 Interface

```
Interface Animal {
  void hacerSonido ( );
}
Class Perro implements Animal {
  Public void hacerSonido ( ) {
  System. out. println ("Guau"),
  }
}
  Public class Main{
  Public static void main (String[ ] args) {
      // Referencia a un objeto de clase Perro
      Animal miPerro = new Perro ( );
      // Invocando el método miPerro.hacerSonido ( );
      }
    }
  }
```