Programacion Orientada a Objetos Resumen General

OBJETOS

los objetos puden ser algo real y existencia. LOS OBJETOS TINEN QUE TENER DOS COSAS SIEMOPRE :

1. ATRIBUTOS (caracteristicas o propiedades)
2. METODOS (Aciones o comportamientos terminar en ar, er , ir..)

Ejemplo de un AUTO.

Clases

Atraves de una clase se pude creear (instanciar) objetos. Con una clase se pude creear varios objetos. conjunto de objetos con caracteristica similares(ejepmlo diferentes tipos de coches). SIMPRE VAN EN SINGULAR (COCHE). Ejemplo de Clase (esto van el digrama UML).

Coche. -color. -marca. -km. encender(). acelerar(). frenar().

Codigo de clase

En mayuscula y singular. public class Coche { }

Codigo de Objeto

1 nombre de la clase (coche),2 nombre del objeto auto1 , 3 constructores : new + clase: comando para crear objetos en java. Coche auto1 = new Coche(). Coche auto2 = nex Coche (). **SE CREO UN NUEVO PRYECTO JAVA POO**.

Concepto y creacion de metodos

EJENPLO DEL CODIGO.

Parametros

Son de claraciones en los parentesis dentro de los metodos , puden ser cualquier tipo de dato primitivo **Ejemplo** public class Calculadora { public int sumar(int num1, int num2) { // parametro int num.... return num1 + num2; }

Variables

Dos tipos de variables : Locales : estan declaradas dentro de cada metodo Globales: Se encuentran en la clase en general **Ejemplo**

public class Ejemplo { // Declaración de variable global private int edad; // Declaracion de variable local public void ejemplo() { int numero = 10; // Esto es una variable local System.out.println(numero); }

Constructor

- Metodo especial que lleva el mismo nombre de la clase
- No tienen retorno explicitos , puden incluir parametros
- S utilizan para inicializar variables

 Codigo de contructor con this

Mejor forma de hacer un coantructor public class persona { private String cedula; private String nombre; private String apellido;

```
    setApellido(apellido);  //this.apellido = "";
    setCedula(cedula);     //this.cedula   = "";
    setNombre(nombre);     //this.nombre   = "";
```

Con los this tambien se puede pero es major con los set. En resuemn el constructor inicializa los atributos del objeto

Getters y Setters

Getters.

(get) solo para lectura. (set) para escritura , en este es donde se crean todas las condiciones para robusted del programa

En resumen los getters y setters ayudan a brindar un nivel de seguridad y robusted en el codigo

Diagramas UML

Este tipo de diagrama almcena los atributos y metodos de una clase en este ejemplo de 'Perro'

| Perro |

| - nombre: String |

| - raza: String |

| - edad: int |

| + Perro(nombre: String, |

| raza: String, | | edad: int) | | + ladrar(): void | | + moverse(): void | | + dormir(): void | | + comer(): void | | + getNombre(): String | | + setNombre(nombre: String): | | void | | + getRaza(): String | | + setRaza(raza: String): | | void | | + getEdad(): int |

+ setEdad(edad: int): void

- Primer cuadro : la clase en este caso perro
- Segundo: Los atributos de este es buna practica colorcar todos en privado

- Tercero: Los metodos y los que retorna cada uno Atributos pero con setters y getter con sus parametrso y lo que retornan

Concepto de Herencia

Simpolemente se trata de una clase superior en este caso una super clase (Animal) la que cual Hereda atributos o metodos a una sub clases (perro , gato)

Ejemplo // Superclase public class Animal { protected String nombre; protected int edad;

```
public Animal(String nombre, int edad) {
    this.nombre = nombre;
    this.edad = edad;
}

public void hacerSonido() {
    System.out.println("El animal hace un sonido");
}
```

}

// Subclase public class Perro extends Animal {

```
public Perro(String nombre, int edad) {
    super(nombre, edad);
}

@Override
public void hacerSonido() {
    System.out.println("El perro ladra");
}

public void moverCola() {
    System.out.println("El perro mueve la cola");
}
```

} La herciaa permite la reutilazacion de codigo sin duplicacion

Imagen concepeto de herencia persona Codigo de contructor con this