**CONCEPTOS GENERALES**

**La Clase Object**

La clase Object está situada en la parte más alta del árbol de la herencia en el entorno de desarrollo de Java. Todas las clases del sistema Java son subclases de la clase Object.

Esta clase define los estados y comportamientos básicos que todos los objetos deben tener, como la posibilidad de compararse unos con otros, de convertirse a cadenas.

Alguno de sus métodos más comunes: .getClass() .toString() .equals() .hasCode().

**Ejemplo de código:**

Object obj=new Object();

System.out.println("Clase :"+obj.getClass().getName());

System.out.println("SuperClase :"+obj.getClass().getSuperclass());

**Salida:**

Clase: java.lang.Object

SuperClase: null

**Ejemplo2:**

String str=new String();

System.out.println("Clase:"+ str.getClass().getName());

System.out.println("SuperClase:"+ str.getClass().getSuperclass());

**Salida:**

Clase: java.lang.String

SuperClase: class java.lang.Object

**La clase String**

**Definición**

La clase String es utilizada para manejar cadenas de caracteres. El operador + se utiliza para concatenar Strings.

Entre los métodos más conocidos estan charAt(), equals(), length(), replaceAll().

**Inicialización de un String**

Las formas de declarar e inicializar un objeto de la clase String son las siguientes:

  String cadena = "Soy una cadena de caracteres";

  String cadena = new String("Soy una cadena de caracteres");

  char[] caracteres = {'h','o','l','a'};

  String cadena = new String(caracteres);

Todos los objetos en java heredan de la clase Object, y la clase Object tiene un método toString() que nos retorna la representación del objeto en un String.

Este método puede ser sobreescrito de modo tal que la información que se retorne en el String resultado sea propia de la instancia de la clase.

Ejemplo:

  public class Auto{

        private String patente;

        private String marca;

        private String modelo;

        // Aqui van los Constructores

    // Aqui van los setters y los getters

    // Este es el metodo que retorna al objeto Auto como un String

    public String toString(){

          return "[Patente:"+this.patente+"] Marca:"+this.marca+" Modelo:"+this.modelo;

        }

  }

**Métodos más importantes**

La clase String también tiene los siguientes métodos:

compareTo(String) que compara un String con otro por igualdad.

concat(String) que concatena un String con otro.

contains() que evalúa si una cadena de caracteres está contenida en el String.

startsWith(String prefijo) que evalúa si el String comienza con el prefijo.

endsWith(String sufijo) que evalua si el String termina con el sufijo.

valueOf() que convierte cualquier tipo básico, boolean, long, double, int, en un String.

split(String regularExpression) que separa en partes según la expresión regular.

subString(int comienzo, int fin) que retorna un bloque desde comienzo y hasta fin.

matches(String regularExpression) que evalua si el String contiene la expresion regular.

**La clase System**

**Definición**

La clase System representa al sistema donde se está ejecutando el programa Java. Por lo tanto, se la puede utilizar para interactuar con el entorno en el que corre, y utilizar las propiedades del entorno, sistema operativo, usuario y demás.

**Dónde utilizarla**

Puede accederse a la salida standard del proceso, a la salida de error del proceso y a la entrada standard del proceso mediante, System.out, System.err, System.in respectivamente, de modo de poder interactuar con el Sistema Operativo. También permite obtener el tiempo en milisegundos llamando al sistema operativo para lograr este objetivo.

Otra cosa muy útil de la Clase System es que permite cargar librerías externas en una línea de ejecución con el método load(), mediante esta operación se logra cargar librerías en tiempo de corrida del programa y sumar así funcionalidades a nuestros programas.

Posee atributos y métodos de uso general, y son todos estáticos, es decir atributos y métodos de clase. No es una clase instanciable.

Entre los métodos más conocidos estan:

      exit(): termina la ejecución de la Java Virtual Machine

      gc(): invoca al Garbage Collector

      getProperties(): trae todas las propiedades del sistema

      getProperty(): trae una propiedad en particular del sistema

**Los Wrappers de los tipos de dato primitivos**

**Definición**

Es la representación de los valores primitivos como objetos, son utilizados para envolver los tipos de datos primitivos. En general tienen dos constructores:

* Con los valores primitivos como parámetro
* Con los valores primitivos como cadena de caracteres, como parámetro

Tienen un método estático valueOf() que retorna un objeto Wrapper.

**La clase Integer**

Utilizada para envolver el tipo de dato primitivo int. Es una sub clase de java.lang.Number.

Ejemplo en java:

Integer miEntero = new Integer(50);

Integer miOtroEntero = Integer.valueOf("50");

**La clase Float**

Utilizada para envolver el tipo de dato primitivo float. Es una subclase de java.lang.Number.

Ejemplo en Java:

Float miPuntoFlotante = new Float(45.67);

Float miOtroPuntoFlotante = Float.valueOf(45.67);

**La clase Number**

Representa el concepto abstracto de un numero en Java. Tiene como objetivo ser la superclase de los Wrappers de los tipos de dato numéricos primitivos. Es una clase abstracta, es decir que no se puede instanciar.

**Comparación entre objetos**

**El operador ==**

Este operador esta sobrecargado para operar con objetos. Se utiliza para saber si dos punteros apuntan al mismo objeto, es decir que son dos referencias de la misma instancia de objeto. Retorna true o false según el resultado de la comparación.

Ejemplo:

String c1 = new String("Juan");

String c2 = c1;

if(c1 == c2){

System.out.println("c1 y c2 apuntan al mismo objeto");

}

**El método equals()**

Se utiliza para saber si dos objetos son iguales, si el contenido es igual. Esta redefinido por subclase que lo utiliza. En la clase String se utiliza para saber si dos objetos String son iguales.

En la clase Integer se utiliza para saber si dos objetos Integer son iguales. Retorna true o false según el resultado de la comparación.

Ejemplo:

String c1 = new String("Juan");

String c2 = new String("Pedro");

if(c1.equals(c2)){

System.out.println("c1 y c2 son iguales");

}