CLASE DATE

La clase Date modela objetos o variables de tipo fecha. La clase Date representa un instante de tiempo específico con una precisión en milisegundos y permite el uso del formato Universal Coordinated Time (UTC). Por otro lado, muchas computadoras están definidas en términos de Greenwich Mean Time (GMT) que es equivalente a Universal Time (UT). GMT es el nombre estándar y UT es el nombre científico del estándar. La diferencia entre UT y UTC es que UTC está basado en un reloj atómico y UT está basado en un reloj astronómico.

Las fechas en Java, comienzan en el valor standar based time llamado “epoch” que hace referencia al 1 de Enero de 1970, 0 horas 0 minutos 0 segundos GMT.

La clase Date posee métodos que permiten la manipulación de fechas. La clase Date pertenece al paquete java.util del API de Java.

Método	Descripción.
Date()	Constructor que inicializa la fecha en el milisegundo más cercano a la fecha del sistema.
Date(int dia, int mes, int año)	Constructor que inicializa la fecha sumándole 1900 al año.
after(Date fecha2)	Retorna verdadero si la fecha esta después de la fecha del parámetro
before(Date fecha2)	Retorna verdadero si la fecha esta antes de la fecha del parámetro
compareTo(Date fecha)	Compara la fecha con la del parámetro. Retorna 0 si son iguales, entero negativo si el primer número es menor o entero positivo si el primer número es mayor.
equals(Object obj)	Compara el Date con el objeto del parámetro. Devuelve true si son iguales y false si no.
getDay()	Retorna el valor del día de la semana de la fecha. Ejemplo: si es lunes devuelve 0, martes 1, miércoles 2, jueves 3, viernes 4, sábado 5 y domingo 6.
getDate()	Retorna el número del día de la fecha.
getMonth()	Retorna el mes de la fecha.
getYear()	Retorna el año de la fecha.
getTime()	Retorna la fecha en milisegundos a partir del “epoch”.
setDate(int dia)	Asigna un día a la fecha.
setMonth(int mes)	Asigna un mes a la fecha.
setYear(int anio)	Asigna un año a la fecha.
setTime(long time)	Asigna la fecha en milisegundos a partir del “epoch”.
toString()	Retorna la fecha en una cadena de caracteres.

CLASES DE UTILIDAD PARTE 2

Recordemos que las clases de utilidad son clases dentro del API de Java que son muy utilizadas en el desarrollo de aplicaciones. Las clases de utilidad son clases que definen un conjunto de métodos que realizan funciones, normalmente muy reutilizadas. Estas nos van a ayudar junto con las estructuras de control, a lograr resolver problemas de manera más sencilla.

Entre las clases de utilidad de Java más utilizadas y conocidas están las siguientes: Arrays, String, Integer, Math, Date, Calendar y GregorianCalendar. En la guía anterior vimos solo las clases Math y String. Ahora vamos a ver el resto de las clases.

CLASE ARRAYS

La clase Arrays es una clase de utilidad que posee una gran cantidad de métodos para manipular arreglos.

Método	Descripción
Arrays.equals(arreglo1, arreglo2)	Retorna true o false, si dos arreglos del mismo tipo de dato son iguales.
Arrays.fill(arreglo, variable) **Arrays.fill(arreglo, int desde, int hasta, variable)**	Este método lo que hace es inicializar todo el arreglo con la variable o valor que pasamos como argumento. Este método se puede usar con cualquier tipo de dato y le podemos decir desde y hasta que índice queremos que llene con ese valor.
Arrays.sort(arreglo) **Arrays.sort(arreglo, int desde, int hasta)**	Este método sirve para ordenar un arreglo de manera ascendente. A este método también le podemos decir desde y hasta que índice queremos que ordene.
Arrays.toString(arreglo)	Este método imprime el arreglo como una cadena, la cadena consiste en una lista de los elementos del arreglo encerrados entre corchetes ("[]"). Los elementos adyacentes están separados por comas (",").

CLASE INTEGER

La clase Integer permite convertir un tipo primitivo de dato int a objeto Integer. La clase Integer pertenece al paquete java.lang del API de Java y hereda de la clase java.lang.Number.

Método	Descripción
Integer(String s)	Constructor que inicializa un objeto con una cadena de caracteres. Esta cadena debe contener un número entero.
compareTo(entero, otroEntero)	Compara dos objetos Integer numéricamente. Retorna 0 si son iguales, entero negativo si el primer número es menor o entero positivo si el primer número es mayor.
doubleValue()	Retorna el valor del Integer en tipo primitivo double
equals(Object obj)	Compara el Integer con el objeto del parámetro. Devuelve true si son iguales y false si no.
parseInt(String s)	Convierte la cadena de caracteres numérica del parámetro en tipo primitivo int.
toString()	Retorna el valor del Integer en una cadena de caracteres.