CURSO DE PROGRAMACIÓN FULL STACK

APÉNDICE B

CLASES DE UTILIDAD

EGG

CLASES DE UTILIDAD

Dentro del API de Java existe una gran colección de clases que son muy utilizadas en el desarrollo de aplicaciones. Entre las clases de utilidad de Java más utilizadas y conocidas están las siguientes: Arrays, String, Integer, Math, Date, Calendar y GregorianCalendar.

CLASE ARRAYS

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

MÉTODOS DE COMPARACIÓN DE ARREGLOS

La clase Arrays contiene una serie de métodos para determinar si dos arreglos son iguales. Estos métodos se listan a continuación.

```
static boolean equals(boolean[] a, boolean[] a2).

static boolean equals(byte[] a, byte[] a2)

static boolean equals(char[] a, char[] a2)

static boolean equals(double[] a, double[] a2)

static boolean equals(float[] a, float[] a2)

static boolean equals(int[] a, int[] a2)

static boolean equals(long[] a, long[] a2)

static boolean equals(Object[] a, Object[] a2)

static boolean equals(short[] a, short[] a2)
```

MÉTODOS DE INICIALIZACIÓN DE ARREGLOS

La clase Arrays contiene una serie de métodos para inicializar arreglos con un valor determinado.

```
static void fill(char[] a, char valor)
```

Este método lo que hace es inicializar todo el arreglo a con el carácter que pasamos como argumento (valor). El resto de los métodos para otros tipos de datos se detallan a continuación.

```
static void fill(boolean[] a, boolean valor)

static void fill(boolean[] a, int desde, int hasta, boolean valor)

static void fill(byte[] a, byte valor)

static void fill(byte[] a, int desde, int hasta, byte valor)
```

```
static void fill(char[] a, char valor)

static void fill(char[] a, int desde, int hasta, char valor)

static void fill(double[] a, double valor)

static void fill(double[] a, int desde, int hasta, double valor)

static void fill(float[] a, float valor)

static void fill(float[] a, int desde, int hasta, float valor)

static void fill(int[] a, int valor)

static void fill(int[] a, int desde, int hasta, int valor)

static void fill(long[] a, int desde, int hasta, long valor)

static void fill(long[] a, long valor)

static void fill(Object[] a, int desde, int hasta, Object valor)

static void fill(Object[] a, Object valor)

static void fill(short[] a, int desde, int hasta, short valor)

static void fill(short[] a, short valor)
```

MÉTODOS DE ORDENAMIENTO DE ARREGLOS

La clase Arrays también posee una serie de métodos que muy útil para ordenar arreglos.

```
static void sort(byte[] a)

static void sort(byte[] a, int fromIndex, int toIndex)

static void sort(char[] a)

static void sort(char[] a, int fromIndex, int toIndex)

static void sort(double[] a)

static void sort(double[] a, int fromIndex, int toIndex)

static void sort(float[] a)

static void sort(float[] a, int fromIndex, int toIndex)

static void sort(int[] a)

static void sort(int[] a, int fromIndex, int toIndex)

static void sort(long[] a)
```

```
static void sort(long[] a, int fromIndex, int toIndex)

static void sort(Object[] a)

static void sort(Object[] a, Comparator c)

static void sort(Object[] a, int fromIndex, int toIndex)

static void sort(Object[] a, int fromIndex, int toIndex, Comparator c)

static void sort(short[] a)

static void sort(short[] a, int fromIndex, int toIndex)
```

MÉTODOS DE BÚSQUEDA EN ARREGLOS

La clase Arrays posee un método binarySearch que sirve para buscar un elemento determinado en un arreglo. El método devuelve la posición en la cual se encuentra el elemento. La implementación del algoritmo de búsqueda utilizado es el de búsqueda binaria, por lo tanto, antes de utilizar este método debemos asegurarnos que el arreglo se encuentre ordenado.

Los métodos binarySearch se detallan a continuación.

```
static int binarySearch(byte[] a, byte key)

static int binarySearch(char[] a, char key)

static int binarySearch(double[] a, double key)

static int binarySearch(float[] a, float key)

static int binarySearch(int[] a, int key)

static int binarySearch(long[] a, long key)

static int binarySearch(Object[] a, Object key)

static int binarySearch(Object[] a, Object key, Comparator c)

static int binarySearch(short[] a, short key)
```

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.

MÉTODOS

RETORNO	MÉTODO	DESCRIPCIÓN
void	Integer(int value)	Constructor que inicializa un objeto con un dato primitivo.
void	Integer(String s)	Constructor que inicializa un objeto con una cadena de caracteres. Esta cadena debe contener un número entero.
int	compareTo(Integer anotherInteger)	Compara dos objetos Integer numéricamente.
double	doubleValue()	Retorna el valor del Integer en tipo primitivo double
boolean	equals(Object obj)	Compara el Integer con el objeto del parámetro
float	floatValue()	Retorna el valor del Integer en tipo primitivo float
int	intValue()	Retorna el valor del Integer en tipo primitivo int
long	longValue()	Retorna el valor del Integer en tipo primitivo long
static int	parseInt(String s)	Convierte la cadena de caracteres del parámetro en tipo primitivo int
short	shortValue()	Retorna el valor del Integer en tipo primitivo short
String	toBinaryString(int i)	Retorna el número del parámetro en su correspondiente cantidad binaria en una cadena de caracteres.
String	toHexString(int i)	Retorna el número del parámetro en su correspondiente cantidad hexadecimal en una cadena de caracteres.

String	toOctalString(int i)	Retorna el número del parámetro en su correspondiente cantidad octal en una cadena de caracteres.
String	toString()	Retorna el valor del Integer en una cadena de caracteres.
Integer	valueOf(int i)	Retorna el número del parámetro en un objeto Integer.
Integer	valueOf(String s)	Retorna la cadena del parámetro en un objeto Integer.

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.

MÉTODOS

RETORNO	MÉTODO	DESCRIPCIÓN
void	Date()	Constructor que inicializa la fecha en el milisegundo más cercano a la fecha del sistema
void	Date(long date)	Constructor que inicializa la fecha en milisegundos del parámetro a partir del “epoch”
boolean	after(Date when)	Retorna verdadero si la fecha esta después de la fecha del parámetro
boolean	before(Date when)	Retorna verdadero si la fecha esta antes de la fecha del parámetro

int	compareTo(Date anotherDate)	Compara la fecha con la del parámetro. Si retorna 0 las fechas son iguales
boolean	equals(Object obj)	Retorna verdadero si la fecha es igual a la del objeto del parámetro
long	getTime()	Retorna la fecha en milisegundos a partir del “epoch”
void	setTime(long time)	Asigna la fecha en milisegundos a partir del “epoch”
String	toString()	Retorna la fecha en una cadena de caracteres

CLASES CALENDAR Y GREGORIANCALENDAR

Estas funcionalidades las obtenemos de dos clases en particular; una es abstracta: java.util.Calendar (a partir de ahora nos referiremos a ella sólo como Calendar) y permite obtener campos enteros como día, mes y año de objetos de tipo java.util.Date o que hereden de él. La otra java.util.GregorianCalendar (a partir de ahora simplemente GregorianCalendar) es una implementación del calendario gregoriano que es usado en casi todo el mundo (es el que conocemos). Si queremos crear una fecha determinada se debe usar la clase GregorianCalendar.

CONSTRUCTOR	DESCRIPCIÓN
GregorianCalendar()	El constructor por defecto que utiliza los datos locales del equipo.
GregorianCalendar(int anyo, int mes, int dia)	El constructor que utiliza los datos locales del equipo pero con una fecha informada.
GregorianCalendar(int mes, int dia, int hora, int minuto, int segundo)	El constructor utiliza los datos locales del equipo pero con la fecha y hora informada
GregorianCalendar(int anyo, int mes, int dia, int hora, int minuto, int segundo)	El constructor utiliza los datos locales del equipo pero con la fecha y hora informada
GregorianCalendar(Locale locale)	Construye un calendario en base a la zona horario local y el dato local informado
GregorianCalendar(TimeZone zona)	Construye un calendario en base a la zona horaria informada y el dato local del equipo
GregorianCalendar(TimeZone zona, Locale locale)	Construye un calendario en base a la zona horaria y el dato local informado

MÉTODOS	DESCRIPCIÓN
add(int campo, int cantidad)	Agrega la cantidad (con signo) de tiempo informada en el campo de tiempo dado segun las reglas del calendario
computeFields()	Transforma tiempo UTC en milisegundos para valores de campos de tiempo
computeTime()	Anula al calendario para convertir valores del campo de tiempo en UTC como miliegundos
equals(Objeto obj)	Compara el calendario Gregoriano con el objeto de referencia
get(int campo)	Consigue el valor del campo de tiempo informado
getActualMaximum(int campo)	Retorna el valor maximo que el campo informado podria tener de la fecha actual
getActualMinimum(int campo)	Retorna el valor minimo que el campo informado podria tener de la fecha actual
getGreatestMinimum(int field)	Consigue el valor mas alto de los minimos si varia
getGregorianChange()	Consigue el cambio de fecha en el calendario Gregoriano
getLeastMaximum()	Consigue el valor mas bajo de los maximos si varia
getMaximum(int campo)	Devuelve el valor maximo del campo informado
getTime()	Consigue la hora actual del calendario
getTimeInMillis()	Consigue la hora actual en milisegundos y con formato long
getTimeZone()	Consigue la zona horaria
getMinimum(int campo)	Devuelve el valor minimo del campo informado
hashCode()	Anula el código hash
isLeapYear(int anyo)	Determina si el año informado es bisiesto
roll(int campo, boolean up)	Adiciona o substrae una unidad simple de tiempo sobre el campo de tiempo informado sin reportar cambios en los campos más grandes
set(int campo, int value)	Setea el campo del tiempo con el valor informado
set(int anyo, int mes, int dia)	Setea los valores del campo año, mes y dia
set(int anyo, int mes, int dia, int hora, int miinuto)	Setea los valores del campo año, mes, dia, hora y minuto

set(int anyo, int mes, int dia, int hora, int minuto, int segundos)	Setea los valores del campo año, mes, dia, hora, minutos y segundos
setGregorianChange(Date fecha)	Setea el cambio de fecha del calendario Gregoriano
setTime(Date fecha)	Setea la hora actual del calendario Gregoria con la fecha informada
setTimeInMillis(long millis)	Setea la hora actual del calendario Gregoria con el valor informado de tipo long en milisegundos
setTimeZone(TimeZone value)	Setea la zona horaria con el valor informado
toString()	Devuelve una representación en cadena de este calendario