Contenidos de la API

String:

La clase `String` sirve para representar cadenas de caracteres. Todas las literales como “abc”, son implementadas como como instancias de esta clase.

La clase `String` genera objetos inmutables, lo cual quiere decir que no pueden cambiar de estado (contenido), cuando se invoca algún metodo para hacer modificaciones como eliminar, concatenar, etc. se genera un nuevo objeto.

La clase `String` utiliza un mecanismo para ahorrar memoria conocido como *String Constant Pool.*

La clase `String` es final, lo cual quiere decir que no puedes sobre escribir sus métodos.

StringBuilder:

Es una clase que nos permite crear cadenas de caracteres mediante un objeto mutable, es decir que si puede cambiar el estado del objeto con la mayoría de los métodos invocados de esta clase.

`StringBuilder` no sobre escribe los métodos `equals()` y `hashCode()` debido a que como el estado del objeto puede cambiar por ser mutable, el hash no permanecería constante.

Los métodos de esta clase no son seguros en aplicaciones multi-hilos debido a que NO están sincronizados.

StringBuffer:

Es una clase que nos permite crear cadenas de caracteres mediante un objeto mutable, es decir que si puede cambiar el estado del objeto con la mayoría de los métodos invocados de esta clase.

`StringBuffer` no sobre escribe los métodos `equals()` y `hashCode()` debido a que el estado del objeto puede cambiar por ser mutable, el hash no permanecería constante.

La diferencia que tiene `StringBuffer` con respecto a `StringBuilder` es que la mayoría de sus métodos son Thread-safe por estar sincronizados.

Expresiones Regulares:

- `regex` es la abreviatura de expresiones regulares, las cuales son patrones utilizados para hacer búsquedas dentro de una fuente de datos.

- `regex` te permite crear patrones de búsqueda utilizando caracteres literales o meta caracteres, que te permiten buscar datos abstractos como números, correos electrónicos o fechas. Los meta caracteres que necesitas conocer son:

Metacaracter	Descripción
.	Cualquier carácter
\d	Un digito de [0-9]
\s	Un carácter de espacio en blanco:[\t\n\x0B\f\r]
\w	Un carácter [a-zA-Z_0-9]

Uso Date

- Date es una clase que ha quedado como parte del API y porque es útil para generar una fecha de lo mas simple con la fecha actual del sistema.

- El método mas importante es `getTime()` que devuelve el numero de milisegundos que existen desde el 1 de Enero de 1970 00:00:0000 hasta la fecha.

Uso Date

```
import java.util.Date;

public class FechaDate {

    public static void main(String[] args) {
        Date fecha = new Date();
        System.out.println(fecha);

        Date fecha2 = new Date(2343242332L);
        System.out.println(fecha2);

        long milis = fecha.getTime();
        int years = (int)((((milis / 1000)/60)/60)/24)/365);
        System.out.println("han pasado: " + years +
                "años aprox desde 1 Julio de 1970");
    }
}
```

Uso de DateFormat

- `DateFormat` es una clase abstracta que define métodos para dar formato a fechas (objetos `Date`).

- `DateFormat` incluye 4 tipos de formato para las fechas FULL LONG MEDIUM y SHORT como constantes de la misma clase.

- El método `format(int format)` de la clase `DateFormat` sirve para dar formato a la fecha con el estilo y localidad que se tengan asignados.

Uso de DateFormat

```
import java.text.DateFormat;
import java.util.Date;

public class EstilosDateFormat {

    public static void main(String[] args) {
        String [] estilo = {"DateFormat.FULL","DateFormat.LONG",
                            "DateFormat.MEDIUM","DateFormat.SHORT"};

        for(int i=0;i<estilo.length;i++){
            DateFormat d2 = DateFormat.getDateInstance(i);
            String fecha = d2.format(new Date());
            System.out.println("fecha con formato " + estilo[i] +
                               " es: " + fecha);
        }
    }
}
```

Uso de DateFormat

- Otro método importante es `parse(String source)` el cual nos sirve para transformar (parsear) de `String` a una fecha siempre y cuando sea transformable en caso de no serlo lanzara una `ParseException` por lo cual se debe manejar con try-catch o en su defecto lanzar la excepción de ese mismo tipo.

- El patrón aceptado que se puede transformar (parsear) lo define el tipo de formato que tenga el objeto fecha.

Uso de DateFormat

```
public class FormatoDateFormat {

    public static void main(String[] args) {
        Locale loc = new Locale("fr","FR");
        DateFormat df = DateFormat.getDateInstance(2,loc);
        try{
            Date date = df.parse("31 juillet 2013");
            System.out.println(df.format(date));

            date = df.parse("miercoles 31 de julio de 2013");
            System.out.println(date);

        }catch(ParseException pe){
            System.err.println(pe.getMessage());
        }
    }
}
```

La clase Locale

- La clase `Locale` nos proporciona información acerca del país y lenguaje que se utilizan y sirve para internacionalizar las fechas.

- Los métodos estáticos dentro de la clase `Locale` `getISOConuntries()` y `getISOLanguages()` devuelven un arreglo de `String` con todos los elementos de los paises y los lenguajes que tiene integrado el ISO recordando que la norma maneja los estándares para lenguajes en dos letras minúsculas y en países dos letras mayúsculas.

- Los métodos `getCountry()` y `getLanguage()` son útiles para devolver el país y el lenguaje utilizados en el `Locale` que se invoquen.

La clase Locale

```
public class Localidad {
    public static void main(String[] args) throws ParseException{
        String paisDef = Locale.getDefault().getCountry();
        System.out.println("Pais default: " + paisDef);
        String lengDef = Locale.getDefault().getLanguage();
        System.out.println("Idioma default: " + lengDef);

        Locale location = new Locale("fr","FR");
        DateFormat df = DateFormat.getDateInstance(2,location);
        Date date = df.parse("31 juillet 2013");
        System.out.println(df.format(date));

        for(String pais:Locale.getISOCountries())
        System.out.println(pais);
        for(String idioma:Locale.getISOLanguages())
        System.out.println(idioma);
    }
}
```

La clase NumberFormat

- La clase `NumberFormat` nos proporciona mecanismos para dar formato a números (enteros, decimales) de acuerdo a una localidad, así como para transformar de `String` a números (parseo). Igualmente que la clase Date esta clase es abstracta y utiliza métodos como `getInstance()` o `getNumberInstance()`.

- El método `format(double)` de la clase `NumberFormat` sirve para dar formato a números.

- Contiene algunos métodos para determinar el número máximo y mínimo de dígitos decimales.