Utilizando clases de la librería estándar de Java en MATLAB.

MATLAB tiene una facilidad inherente para la integración simple de código Java en un script ordinario. Es posible instanciar objetos de una clase Java de la librería estándar o de terceros, y utilizarlas dentro del espacio de trabajo.

Lo anterior puede resultar muy útil en casos donde ciertas características del lenguaje Java no implementadas nativamente en MATLAB sean requieridas. En este caso vamos a ver cómo importar, instanciar y utilizar algunas clases y sus métodos de la librería estándar de Java.

Lo primero que debemos hacer para utilizar una clase Java es importar esta en nuestro espacio de trabajo, por ejemplo, si queremos utilizar las clases de la librería `lang` de Java:

```
>> import java.lang.*;
```

Ahora instanciaremos un objeto de la clase String de Java:

```
>> cadena=String('Hola Java')
cadena =
Hola Java
```

Podemos verificar la clase del objeto `cadena` utilizando la función class de MATLAB.

```
>> class(cadena)
ans =
java.lang.String
```

O directamente utilizando el método `getClass`:

```
>> cadena.getClass()
ans =
class java.lang.String
```

De la misma forma podemos utilizar algunos de los métodos disponibles para este objeto:

```
>> cadena.length() % Longitud de la cadena
ans =
     9
>> cadena.replace('a','o') % Remplazando caracteres
ans =
Holo Jovo
>> cadena.toUpperCase()  % Todas mayúsculas
ans =
HOLA JAVA
>> cadena.concat(' y MATLAB')  % Concatenando caracteres al final
```

```
ans =
Hola Java y MATLAB
```

Es posible instanciar un objeto Java sin necesidad de importar las librerías, en cambio es necesario indicar el nombre/ruta completo de la clase:

```
>> cad=java.lang.String('Hola!!!')
cad =
Hola!!!
```

Bueno, pero, ¿no tiene MATLAB un tipo de dato String nativo que hace lo mismo? o ¿para qué nos servirá entonces utilizar objetos Java?.

Si, MATLAB tiene un tipo `char` equivalente al String de MATLAB, pero existen algunos estructuras de datos como las *Hashtables* que no tienen un equivalente en MATLAB, y es ahí donde podríamos aprovechar algunas versatilidades proporcionadas por Java.

¿Y qué es una Hashtable?, en resumen, es un estructura que permite almacenar variables mediante las características clave-valor (como un diccionario en Python, por si vienen de Python). Por ejemplo:

```
>> import java.util.*; % Importamos las clases de java.util
>> ht=Hashtable(); % Instanciamos un objeto de la clase Hashtable
>> ht.put('A',10); % Agregamos un clave-valor
>> ht.put('B',15); % Agregamos un clave-valor
>> ht.put('C',5); % Agregamos un clave-valor
>> ht
ht =
{A=10.0, C=5.0, B=15.0}
>> class(ht)
ans =
java.util.Hashtable
```

Ahora podemos acceder al valor guardado en un determinado campo de la Hashtable:

```
>> ht.get('B')  % Tomando el valor de la clave B
ans =
    15
```

Y bueno, hasta aquí un poco de este tema sobre cómo utilizar clases de la librería estándar de Java en MATLAB. En entradas posteriores veremos cómo añadir controles gráficos de la librería Swing en una GUI MATLAB.