Properties

Clase `Properties` en Java

La clase `Properties` es una subclase de `Hashtable` en Java que permite almacenar pares clave-valor. Se utiliza principalmente para manejar configuraciones y propiedades de un programa, especialmente en archivos de propiedades (con extensión `.properties`).

Características principales de la clase `Properties`:

1. **Herencia**: `Properties` hereda de la clase `Hashtable`, por lo que se comporta como un mapa (clave-valor), pero está específicamente diseñada para trabajar con datos de configuración.
2. **Claves y valores como cadenas**: En un objeto `Properties`, tanto las claves como los valores son **strings**. Esto facilita el almacenamiento de configuraciones que se representan comúnmente como texto.
3. **Métodos adicionales**: La clase `Properties` proporciona métodos adicionales para cargar y guardar propiedades, como `load()` y `store()`, que facilitan la lectura y escritura de archivos `.properties`.

Métodos más utilizados de la clase `Properties`:

1. **`load(InputStream inputStream)`**:
   - Carga las propiedades desde un archivo de entrada. Este archivo suele tener una extensión `.properties` y contiene pares clave-valor.
   - Ejemplo:

```
Properties propiedades = new Properties();
FileInputStream fis = new FileInputStream("config.properties");
propiedades.load(fis);
```

2. **`store(OutputStream outputStream, String comments)`**:
   - Guarda las propiedades en un archivo de salida, generalmente en formato `.properties`.
   - Ejemplo:

```
Properties propiedades = new Properties();
propiedades.setProperty("usuario", "admin");
propiedades.setProperty("contraseña", "12345");
FileOutputStream fos = new FileOutputStream("config.properties");
propiedades.store(fos, "Configuración del sistema");
```

3. **`storeToXML(OutputStream os, String comments)`**:
   - Guarda las propiedades en un archivo de salida en formato **XML**.

- Es útil cuando necesitas un formato más estructurado y legible que `.properties`.
- **Ejemplo**:

```
Properties propiedades = new Properties();
propiedades.setProperty("usuario", "admin");
propiedades.setProperty("contraseña", "12345");
FileOutputStream fos = new FileOutputStream("config.xml");
propiedades.storeToXML(fos, "Configuración del sistema");
fos.close();
```

4. **`getProperty(String key)`**:
   - Obtiene el valor asociado con la clave proporcionada.
   - Ejemplo:

```
String usuario = propiedades.getProperty("usuario");
```

5. **`setProperty(String key, String value)`**:
   - Establece un valor para una clave dada. Si la clave ya existe, su valor será actualizado.
   - Ejemplo:

```
propiedades.setProperty("usuario", "admin");
```

6. **`stringPropertyNames()`**:
   - Devuelve un conjunto con todas las claves (como `String`) que están presentes en el objeto `Properties`.
   - Ejemplo:

```
Set<String> claves = propiedades.stringPropertyNames();
```

7. **`getProperty(String key, String defaultValue)`**:
   - Devuelve el valor asociado con una clave, o un valor por defecto si la clave no existe.
   - Ejemplo:

```
String usuario = propiedades.getProperty("usuario", "defaultUser");
```