Sistemi per l'elaborazione dell'informazione 2

Serializzazione logica

A volte il formato standard di serializzazione generato da Java non è il più adatto. Questo è vero, in particolare, se lo stato interno di un oggetto presenta ridondanza.

Per prendere coscienza del problema, provate a estendere `java.util.HashMap` in modo che fornisca un servizio di accesso invertito: di solito, una `Map` permette, tramite il metodo `get()`, di ottenere il valore associato a una chiave. Dovete aggiungere un metodo `lookup()` che permetta di ottenere una chiave, dato un valore. Per fare questo, estendete `HashMap` in modo che contenga una *seconda* mappa in cui inserite le coppie chiave/valore in ordine invertito (per ottenere questo effetto è necessario sovrascrivere `put()` e `clear()`). In pratica, state implementando una biiezione, e dovete fare in modo che le invocazioni di `put()` non violino le proprietà matematiche corrispondenti. Potete lanciare una `IllegalStateException` per segnalare una condizione di errore.

Ovviamente, se rendete la nuova classe serializzabile verranno salvate inutilmente *due* copie delle coppie chiave/valore: una nella `HashMap` che state estendendo, e una nella mappa di inversione. Per ovviare a questo inconveniente, dovete rendere la mappa di inversione transiente, e modificare `readObject()` in modo che la ricostruisca enumerando le coppie chiave/valore della mappa principale (dovete utilizzare il metodo `entrySet()` per enumerare le coppie, e reinserirle in una nuova mappa di inversione con chiave e valore scambiati).

Realizzate un metodo `main()` o una classe di test in cui istanziate la classe che avete creato, la popolate con alcune associazioni, ne stampate il contenuto (utilizzando un metodo `toString()` opportunamente creato), serializzate l'oggetto ottenuto, lo deserializzate e stampate il contenuto dell'oggetto che ottenete, verificando che gli output coincidono.

Create infine un'applicazione che visualizzi una biiezione all'interno di una finestra contenente i seguenti widget:

- una lista che visualizzi le coppie (chiave, valore) contenute nella biiezione
- un pulsante che permetta di rimuovere dalla biiezione le coppie (chiave, valore) selezionate nella lista;
- un pulsante che permetta di rimuovere dalla biiezione tutte le coppie in essa contenute;
- due caselle di testo deputate a contenere una chiave e un valore, affiancate a un pulstante che permetta di inserire una nuova coppia nella biiezione, utilizzando i valori inseriti nelle caselle di testo.

Scegliete un opportuno dispositore per inserire i widget nella shell e gestite la segnalazione di errore nel caso in cui l'inserimento di una nuova coppia violi l'integrità della biiezione (usando magari una `MessageBox`). Aggiungete poi alla vostra applicazione un menu che permetta di serializzare e deserializzare la biiezione (quando deserializzate eliminate l'istanza precedente di `BijectiveHashMap`). I più arditi possono anche verificare i casi specifici in cui si carica una nuova biiezione senza aver salvato quella corrente, avvertendo l'utente della potenziale perdita di dati e offrendo la possibilità di salvare la biiezione corrente prima di procedere. In tal caso sarà utile utilizzare il disegno *command* e un `MessageBox` opportuno.

`readResolve()`

A volte non è possibile utilizzare in assoluto l'oggetto generato implicitamente dal sistema di serializzazione. (Un tipico caso è quello dei singoletti che vedremo in una delle prossime lezioni).

Per risolvere questi problemi, Java fornisce il metodo `readResolve()` (oltre alla sua controparte `writeReplace()`, che però non vi servirà in questo esercizio). Se viene definito un metodo

```
private Object readResolve()
```

all'interno di una classe, esso verrà invocato al momento della deserializzazione (e `this` sarà l'oggetto deserializzato). Il metodo restituisce un oggetto che sarà considerato il risultato della deserializzazione. Per esempio, il metodo può decidere di buttare via la nuova istanza (cioè `this`) e restituire un altro oggetto.

Riscrivete la soluzione del primo esercizio utilizzando il metodo `readResolve`.