Example13

Table of contents

- Aspect-Oriented Programming
  1. Persistent Aspect
  2. `it.unipr.informatica.aspects`
     1. `PersistentAspect.java`
  3. Example13
     1. `Example13.java`

Aspect-Oriented Programming

Persistent Aspect

Si dice persistente se *persiste*, un oggetto il cui stato sopravvive alla chiusura dell'applicazione. L'oggetto rimarra' vivo, che rimarra' in un file, tabella DBMS, e rimarra' finche' l'applicazione non viene riaperta.

Il modo che useremo e' esplicito: costruiremo degli oggetti e assoceremo a questi, dei file: nel momento in cui l'oggetto andra' salvato, chiameremo `commit()` per terminare la transazione; faremo `rollback()` per tornare indietro.

L'interfaccia `java.io.Serializable` sara' quella che useremo siccome usa la trasformazione in un array di byte, e viceversa.
Usiamo questa tecnica per caricare/scaricare l'oggetto.

`it.unipr.informatica.aspects`

`PersistentAspect.java`

La solita struttura di un metodo statico `attach()`, passando `file` e l'oggetto che vogliamo serializzare `object` a cui vogliamo attaccare l'aspetto persistente.
Se il file esiste, viene caricato e reimpostato ai valori esistenti, altrimenti l'oggetto passato come argomento verra' usato come argomento.

In uscita ha un `PersistentHandler`, lo usiamo per gestire tutte le operazioni extra che vengono ad assumere gli oggetti nel momento in cui attacchiamo un aspetto.

```
        // ...
        public static <T extends Serializable> PersistentHandler<T>
                attach(File file, T object) throws IOException {
                if (object == null)
                        throw new IllegalArgumentException("object == null");
                if (file == null)
                        throw new IllegalArgumentException("file == null");
                if (file.exists() && !file.isFile())
                        throw new IllegalArgumentException
                                ("file.exists() && !file.isFile()");
```

```
                InnerPersistentHandler<T> handler =
                        new InnerPersistentHandler<T>(file);
                if (file.exists())
                        handler.load();
                else
                        handler.target = object;
                return handler;
        }
        // ...
```

L'implementazione dell'interfaccia `PersistentHandler` ha 3 metodi:

- `commit()`
  salva l'oggetto sul file;
- `rollback()`
  carica l'oggetto dal file, con tutte le modifiche perse;
- `get()`
  accedere all'oggetto a cui e' stato attaccato l'aspetto persistente.

Per l'operazione abbiamo file di `InputStream` per la lettura (di blocchi di byte), `OutputStream` per la scrittura sul file.

Usiamo l'esempio dei libri con l'unica differenza essere che l'interfaccia `Bean` estende l'interfaccia `Serializable`: qualsiasi implementazione potra' essere messa in un file.

Example13

`Example13.java`

`handler` ci permette di lavorare sull'oggetto.
Viene stampato il contenuto attuale dell'elenco dei libri, aggiungendo un numero casuale, costruendo come identificativo il numero generato, l'autore e il titolo.

```
public class Example13 {
        private void printAndAddBooks(List<Book> books) {
                if (!books.isEmpty()) {
                        System.out.println("Current books:");
                        for(Book book : books)
                                System.out.println(book);
                }
                for(int i = 0; i < 3; ++i) {
                        int n = 10 + (int)(90 * Math.random());
                        books.add
                                (new SimpleBook(n, "Author #" + n, "Title #" + n));
                }
        }

        private void go() {
                try {
                        PersistentHandler<ArrayList<Book>> bookHandler =
                                PersistentAspect.attach
                                ("Books.dat", new ArrayList<Book>());
                        List<Book> books = bookHandler.get();
```

```
                        printAndAddBooks(books);
                        bookHandler.commit();
                        System.out.println("Books saved");
                } catch(Throwable throwable) {
                        throwable.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new Example13().go();
        }
}
```

I libri vengono salvati in modo persistente.

Estrapoliamo i libri.