Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo: Refactoring Parte a

versione 1.0

Questo lavoro è concesso in uso secondo i termini di una licenza Creative Commons
(vedi ultima pagina)

SOME RIGHTS RESERVED creative commons

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Introduzione
  - ⇨Il Catalogo dei Refactoring
  - ⇨Il Refactoring Browser
  - ⇨Test e Integrazione Continua
- Un Esempio di Refactoring

Introduzione

- Il termine “refactoring”
  - ⇨traducibile in italiano con “riorganizzazione in fattori” del codice sorgente
- La definizione di Martin Fowler
  - ⇨“*Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code, yet it improves its external structure*”

Introduzione

- In sintesi
  - ⇨è un processo di modifica del codice
  - ⇨non finalizzato all’aggiunta di nuove funzionalità o alla correzione di difetti
  - ⇨ma puramente al miglioramento della qualità
- Esempi di refactoring
  - ⇨cambiare un identificatore poco chiaro
  - ⇨spezzare un metodo lungo in più metodi

Introduzione

- Problema collegato al refactoring
  - ⇨ogni refactoring ha impatto su diversi punti del codice (es: cambiare un identificatore)
  - ⇨se le modifiche non vengono fatte correttamente il refactoring può introdurre regressioni
- Regressione
  - ⇨errore logico o sintattico che prima non c'era

Introduzione

- Il programmatore tipico
  - ⇨non effettua modifiche di questo tipo al codice perchè teme le regressioni
  - ⇨di conseguenza, il codice tende a crescere senza essere mai ristrutturato
  - ⇨diventa intricato e caotico: "imputridisce"
  - ⇨dopo un pò diventa impossibile da manutenere

Introduzione

- Il programmatore evoluto
  - ⇨considera indispensabile il refactoring come tecnica per sviluppare codice di qualità
  - ⇨non ha paura di effettuare modifiche
  - ⇨si dota degli strumenti necessari a gestire le regressioni
- Gli strumenti a servizio del refactoring
  - ⇨essenzialmente di tre categorie

Introduzione

- Strumenti per il refactoring
  - ⇨il catalogo dei refactoring
  - ⇨il refactoring browser
  - ⇨test e integrazione continua
- Nel seguito
  - ⇨discutiamo ciascuno di questi strumenti

Il Catalogo dei Refactoring

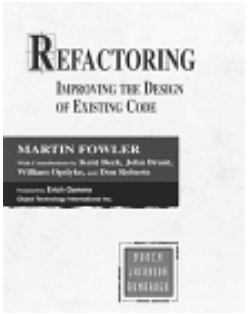
- Idea
  - ⇨il refactoring non deve essere eseguito in modo disordinato ma sistematico
- Catalogo dei Refactoring
  - ⇨le principali tipologie di interventi sul codice sono state classificate e catalogate
  - ⇨ognuna ha un nome e una sequenza di operazioni da fare per evitare di introdurre errori nel codice

Il Catalogo dei Refactoring

- Un esempio di refactoring: “rename”
- Obiettivo
  - ⇨ridenominare un dato (es: classe Circonferenza, proprietà private double lr -> private double lunghezzaRaggio)
- La tecnica
  - ⇨cambiare la dichiarazione del dato
  - ⇨cercare e cambiare tutte le istruzioni in cui compare il vecchio identificatore

Il Catalogo dei Refactoring

- Il libro di riferimento
  - ⇨"Refactoring – Improving the Design of Existing Code" di Martin Fowler
  - ⇨edito da Addison-Wesley
  - ⇨prima edizione del 2000
  - ⇨ormai un classico della programmazione
  - ⇨introduce la tecnica e un catalogo di 72 refactoring



Il Catalogo dei Refactoring

ATTENZIONE al sito refactoring.com

- Successivamente
  - ⇨il catalogo si è allargato
- refactoring.com
  - ⇨il sito di riferimento per il refactoring
  - ⇨creato e gestito da Martin Fowler
  - ⇨contiene attualmente un elenco di quasi 100 refactoring noti

Il Catalogo dei Refactoring

- Idea
  - ⇨il processo di refactoring dovrebbe procedere in modo sistematico
  - ⇨evitando grandi cambiamenti al codice
  - ⇨e invece applicando piccole trasformazioni l'una dopo l'altra che portino dal punto di partenza al punto di arrivo
  - ⇨per ciascuna trasformazione è necessario conoscere ed applicare la tecnica corretta

Il Catalogo dei Refactoring

- Esempio
  - ⇨spostare una parte di un metodo M della classe C1 nella classe C2
  - ⇨operazione n. 1: spezzo M in due parti, M1 ed M2, all'interno di C1
  - ⇨operazione n. 2: sposto il metodo M2 all'interno di C2

Il Refactoring Browser

- In aggiunta
  - ⇨molte trasformazioni possono essere applicate automaticamente applicando uno strumento opportuno
- Refactoring Browser
  - ⇨strumento in grado di automatizzare una certa operazione di refactoging
  - ⇨es: “rename”: rinomina la proprietà raggio dappertutto nel codice del progetto

Il Refactoring Browser

- Tutti i principali IDE
  - ⇨forniscono una qualche forma di supporto al refactoring; es: NetBeans ed Eclipse
  - ⇨inoltre consentono di installare plugin aggiuntivi che ne potenziano le funzionalità di refactoring
- RefactoriIt
  - ⇨plugin che supporta un numero molto maggiore di trasformazioni rispetto agli IDE

Il Refactoring Browser

- Esecuzione del refactoring
  - si suppone di partire da uno stato corretto del codice
  - il programmatore non effettua direttamente la modifica sul codice sorgente
  - chiede allo strumento di effettuare la modifica
  - selezionando l'elemento da trasformare (proprietà, metodo, blocco di codice) e poi scegliendo il refactoring da applicare
  - lo strumento effettua la modifica propagandola in tutto il progetto in modo da portare il codice in uno stato ancora consistente

Il Refactoring Browser

- Nota
  - non tutti i possibili refactoring sono automatizzabili
- Esempio
  - il catalogo originale di Fowler ne contiene 72
  - il catalogo di refactoring.com ne contiene 95
  - NetBeans ne automatizza 4
  - RefactorIt ne automatizza 18

Il Refactoring Browser

- Di conseguenza
  - ⇨in alcuni casi è indispensabile l'intervento manuale del programmatore
  - ⇨e questo rischia di introdurre regressioni
  - ⇨ma l'adozione di tecniche opportune nel processo di sviluppo aiuta a gestirle
- Le tecniche
  - ⇨test di regressione
  - ⇨integrazione continua

Test e Integrazione Continua

- Test di regressione
  - ⇨sono indispensabili per intercettare immediatamente gli errori introdotti dal ref.
  - ⇨dopo ogni (piccolo) passo di refactoring vanno rieseguiti integralmente per accertarsi che non ci siano state regressioni
  - ⇨se ci sono regressioni vengono intercettate immediatamente e possono essere corrette

Test e Integrazione Continua

- Integrazione continua
  - ⇨per non intralciare il processo di sviluppo, l'esecuzione dei test deve essere rapida e indolore
  - ⇨adottando un sistema di costruzione del codice (es: Ant) è possibile eseguire i test ogni volta che il codice viene ricostruito
- In questo modo
  - ⇨il programmatore può "rischiare" di più

Un Esempio di Refactoring

- Per capire meglio il processo
  - ⇨un esempio concreto di refactoring
- Ipotesi
  - ⇨è in corso di sviluppo il sistema informativo di una compagnia aerea
  - ⇨con informazioni su voli, biglietti e passeggeri
  - ⇨la compagnia offre un programma di punti premio basati sulle miglia

Un Esempio di Refactoring

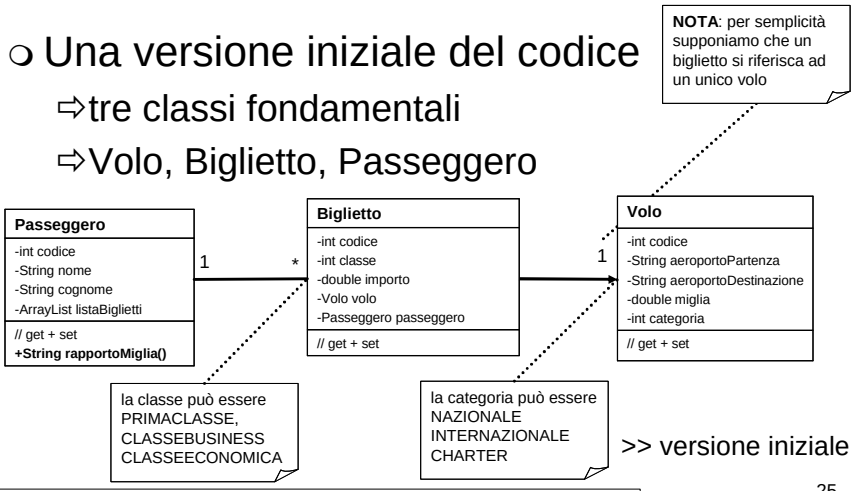
- Le regole del programma punti
  - ⇨il passeggero acquisisce punti per ogni volo
  - ⇨i punti sono proporzionali alle miglia percorse
  - ⇨dipendono dal tipo di volo (nazionale, internazionale, charter...)
  - ⇨e dalla classe in cui il passeggero vola
- Un caso d'uso interessante
  - ⇨"Operatore produce estratto punti cliente"

Un Esempio di Refactoring

- "Operatore produce estratto punti cliente"
  - ⇨l'operatore cerca il cliente specificando il suo codice oppure il suo nome
  - ⇨il sistema stampa l'estratto punti del passeggero su una console con interfaccia a carattere
  - ⇨in particolare, riepiloga i dati del passeggero
  - ⇨riepiloga i dati di tutti i biglietti acquistati e dei voli effettuati, con i relativi punti acquisiti
  - ⇨e infine stampa il totale punti accumulati dall'utente

Un Esempio di Refactoring

- Una versione iniziale del codice
  - ⇨tre classi fondamentali
  - ⇨Volo, Biglietto, Passeggero



>> versione iniziale

Un Esempio di Refactoring

- Il caso d'uso
  - ⇨è implementato quasi completamente nel metodo rapportoMiglia() di Passeggero
  - ⇨scandisce la lista dei biglietti con i relativi voli per produrre i dettagli dell'estratto punti
  - ⇨primo if: analizza la categoria del volo
  - ⇨secondo if: analizza la classe del biglietto

Un Esempio di Refactoring

- Caratteristiche del metodo
  - ⇨lungo (circa 50 linee di codice)
  - ⇨flusso di controllo intricato a causa dei due if
  - ⇨due funzioni diverse: da una parte produrre lo schermo, dall'altra calcolare il totale punti
- Ma...
  - ⇨il metodo implementa correttamente il caso d'uso e non c'è ragione di cambiarlo

Un Esempio di Refactoring

- Quando nasce il problema
  - ⇨viene deciso di sviluppare un'interfaccia alternativa per il sistema (es: HTML) oltre a quella a carattere
  - ⇨viene deciso di introdurre nuove tipologie di volo (offerte speciali, voli fuori stagione ecc.) e nuove regole nei punti premio
  - ⇨quindi il codice deve essere modificato per ospitare le nuove funzionalità

Un Esempio di Refactoring

- Primo approccio (molto discutibile)
  - ⇨tenere il metodo com'è e duplicarne la logica applicativa in estrattoPuntiHTML()
  - ⇨ramificando ulteriormente gli if per ospitare le nuove categorie di voli
- Svantaggio
  - ⇨la logica applicativa è duplicata
  - ⇨ogni cambiamento nelle regole costringe a cambiare entrambi i metodi

Un Esempio di Refactoring

- Secondo approccio (condivisibile)
  - ⇨accorgersi che è opportuno procedere ad un refactoring del codice prima di aggiungere le nuove funzionalità
  - ⇨effettuare il refactoring
  - ⇨e poi aggiungere le nuove funzionalità al codice trasformato
- Seguiamo questo approccio

Un Esempio di Refactoring

- Il presupposto
  - ⇨avere a disposizione una batteria di test di regressione per il codice da riorganizzare
  - ⇨se non sono stati sviluppati in passato, è necessario svilupparli appositamente
  - ⇨nel nostro caso, l'unico metodo che abbia logica applicativa è estrattoPunti() e quindi sviluppiamo test per quello
  - ⇨utilizziamo il sistema di logging per riscontro

>> test versione iniziale

Un Esempio di Refactoring

>> "rename": ognuno in biglietto
>> "rename": ogni in volo

- Per cominciare
  - ⇨analizziamo i problemi del metodo
- I problema
  - ⇨scarsa leggibilità
  - ⇨alcune variabili hanno nomi discutibili (ognuno, ogni)
  - ⇨è necessario procedere alla ridenominazione
  - ⇨applichiamo il refactoring "rename"
  - ⇨e poi rieseguiamo i test

Un Esempio di Refactoring

>> “extract method”: stringaBiglietto()
>> “move method”: in Biglietto
>> “rename”: in toString()

- Il problema
  - ⇨il metodo si assume responsabilità che starebbero meglio in altre classi
  - ⇨es: trasformare in stringa il biglietto
  - ⇨è opportuno spostare il codice in Biglietto
  - ⇨per cominciare applichiamo “extract method” per separare il codice di stringaBiglietto()
  - ⇨poi applichiamo “move method” per spostare il metodo ottenuto in Biglietto.toString()

Un Esempio di Refactoring

- Attenzione
  - ⇨in questo caso, il refactoring browser non consente di effettuare il refactoring voluto in modo completamente automatico
  - ⇨è necessario intervenire manualmente sul codice cambiando il prototipo del metodo toString() ottenuto e la relativa chiamata
  - ⇨questo caso si verifica frequentemente: non sempre lo strumento effettua tutto il lavoro

Un Esempio di Refactoring

>> “extract method”: stringaVolo()
>> “change method signature”
>> “move method”: in Volo
>> “rename”: in toString()

- III problema
  - ⇨ripetiamo la stessa procedura per il volo
  - ⇨extractMethod per stringaVolo
  - ⇨ma stavolta elimino il parametro risultato prima di effettuare il move
  - ⇨eseguendo il refactoring “change method signature” per eliminare il parametro
  - ⇨e poi modifichiamo manualmente il codice
  - ⇨poi sposto il metodo in Volo e rinomino

Un Esempio di Refactoring

- IV problema
  - ⇨il metodo calcola il punteggio relativo al biglietto
  - ⇨il metodo svolge compiti che dovrebbero essere in parte di Volo (punti delle miglia) e in parte di Biglietto (punti di bonus)
  - ⇨è necessario spostare il calcolo dei punti attribuendolo a Biglietto che a sua volta chiamerà Volo per i punti delle miglia

Un Esempio di Refactoring

- A differenza del caso precedente
  - ⇨devo distribuire il codice del metodo di una classe in altre due classi
  - ⇨processo a quattro passi: extract method getPunti() + move in Biglietto
  - ⇨poi da getPunti() extract method getPuntiMiglia() e move in Volo

Un Esempio di Refactoring

>> “extract method”: getPunti()
>> “move method”: in Biglietto
>> “change method signature”
>> “extract method”: getPuntiMiglia()
>> “move method”: in Volo

- Attenzione
  - ⇨anche in questo caso è necessario un intervento manuale
  - ⇨principalmente per via della variabile Volo
  - ⇨il refactoring browser la trasforma in un parametro del nuovo metodo
  - ⇨è opportuno cambiare il prototipo del metodo getPunti() eliminando il parametro e aggiungere l’istruzione Volo volo = getVolo()

Un Esempio di Refactoring

>> “inline” volo

- I Annotazione
  - ⇨a questo punto la variabile volo in estrattoPunti() è praticamente inutile
  - ⇨si tratta di una variabile temporanea utilizzata per chiamare il metodo toString() subito dopo
  - ⇨possiamo eliminarla rimpiazzandola direttamente con la chiamata al metodo getVolo()
  - ⇨refactor. “inline” o “replace temp with query”

Un Esempio di Refactoring

>> refactoring dei test

- II Annotazione
  - ⇨con i recenti refactoring abbiamo introdotto di fatto due nuovi metodi di logica applicativa
  - ⇨getPunti() di Biglietto
  - ⇨e getPuntiMiglia() di Volo
  - ⇨per completare il refactoring è necessario introdurre due nuove classi di test
  - ⇨TestBiglietto e TestVolo

Un Esempio di Refactoring

- V problema di estrattoPunti()
  - ⇨è inopportuno mischiare la produzione dello schermo e il calcolo del totale dei punti
  - ⇨è opportuno separare il calcolo dei punti in un metodo getTotalePunti()
  - ⇨devo applicare di nuovo “extract method” per separare i due metodi

Un Esempio di Refactoring

>> “extract method” getTotalePunti()

- Ma...
  - ⇨in questo caso lo strumento non mi aiuta perchè devo replicare il ciclo
  - ⇨e quindi effettuo il refactoring manualmente
  - ⇨verifico immediatamente con i test
- Anche in questo caso
  - ⇨ho introdotto un nuovo metodo di logica applicativa e quindi sono necessari i test relativi in TestPasseggero

Un Esempio di Refactoring

- Commento
  - ⇨attenzione alle implicazioni di questa scelta
  - ⇨stiamo di fatto “duplicando” la scansione della lista dei biglietti
  - ⇨e quindi raddoppiando la complessità computazionale del metodo
  - ⇨in altri termini stiamo favorendo la buona organizzazione del codice rispetto alla complessità (>>)

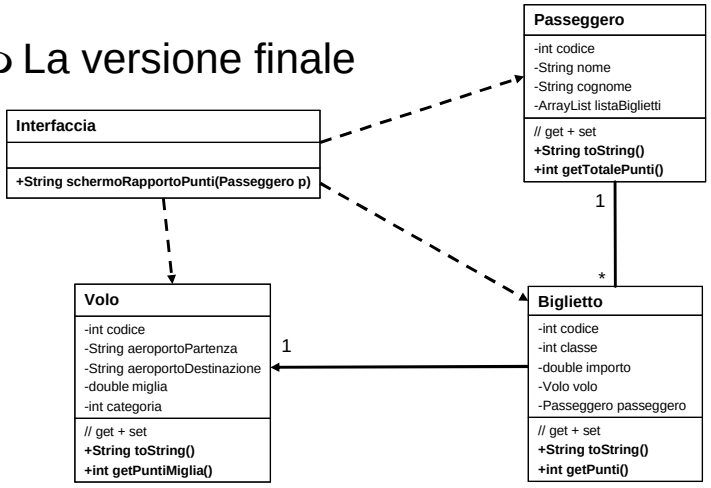
Un Esempio di Refactoring

>> “extract method” toString()
>> “move” schermoEstrattoPunti()

- VI problema
  - ⇨la produzione dello schermo è un compito di interfaccia – è opportuno che Passeggero si limiti ad eseguire un metodo toString()
  - ⇨e che la produzione dello schermo sia affidata ad una classe di interfaccia
  - ⇨introduciamo la classe Interfaccia
  - ⇨riorganizziamo i metodi
  - ⇨e introduciamo la classe TestInterfaccia

Un Esempio di Refactoring

- La versione finale



Un Esempio di Refactoring

- Vantaggi di questa versione
  - ⇨è massimizzata la coesione nelle classi ed è ridotto l'accoppiamento
  - ⇨sono rispettati gli strati applicativi
  - ⇨per modificare lo schermo bisogna intervenire solo su Interfaccia
  - ⇨per modificare le regole sui voli bisogna intervenire solo su Volo
  - ⇨idem per le regole sulla classe del biglietto

Un Esempio di Refactoring

- Alcune caratteristiche del processo
  - ⇨molti piccoli passi intervallati da test continui
  - ⇨l'esempio illustra i refactoring più frequenti (rename, extract method, change method signature, move, inline)
  - ⇨quasi tutti i refactoring sono stati effettuati in maniera automatica dal refactoring browser
  - ⇨in varie circostanze abbiamo dato priorità al refactoring rispetto ai tempi di esecuzione

Riassumendo

- Introduzione
  - ⇨Il Catalogo dei Refactoring
  - ⇨Il Refactoring Browser
  - ⇨Test e Integrazione Continua
- Un Esempio di Refactoring

Ringraziamenti

- L'esempio di refactoring illustrato in questa unità è ispirato all'esempio dei noleggi riportato nel capitolo 1 del libro "Refactoring – Improving the Design of Existing Code" di Martin Fowler, edito da Addison Wesley

Termini della Licenza

- This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/1.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

- Questo lavoro viene concesso in uso secondo i termini della licenza "Attribution-ShareAlike" di Creative Commons. Per ottenere una copia della licenza, è possibile visitare http://creativecommons.org/licenses/by-sa/1.0/ oppure inviare una lettera all'indirizzo Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.