Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo: Refactoring Parte b

versione 1.0

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

Sommario

- Il Catalogo dei Refactoring
- Linee Guida per il Refactoring
- "Code Smells"

Il Catalogo dei Refactoring

- Nell'esempio
  - ⇨vari refactoring del catalogo originale
- "Rename"
  - ⇨rinomina una variabile, proprietà, metodo, package o classe
- Meccanica di "rename"
  - ⇨richiede di modificare tutte le istruzioni che usano l'identificatore modificato

Il Catalogo dei Refactoring

- "Extract Method"
  - ⇨preleva un blocco di istruzioni da un metodo
  - ⇨crea un nuovo metodo nella stessa classe
  - ⇨e sostituisce il blocco con la chiamata al nuovo metodo
- Nota
  - ⇨si tratta di un refactoring molto importante
  - ⇨ma anche abbastanza complesso

Il Catalogo dei Refactoring

- Meccanica
  - ⇨richiede un'analisi dettagliata delle variabili locali e dei parametri del metodo originale
  - ⇨per capire quali diventano variabili locali al nuovo metodo
  - ⇨e quali diventano parametri del nuovo metodo
  - ⇨è opportuno eseguirlo automaticamente

Il Catalogo dei Refactoring

- "Inline" o "Replace Temp with Query"
  - ⇨una variabile temporanea viene sostituita con una chiamata diretta ad un metodo
  - ⇨prima: `Volo volo = biglietto.getVolo();`
    `risultato += volo.toString();`
  - ⇨dopo: `risultato += biglietto.getVolo().toString();`
- Vantaggio
  - ⇨abbrevia il metodo e lo rende più leggibile
  - ⇨semplifica l'applicazione di "extract method"

Il Catalogo dei Refactoring

- "Introduce explaining variable"
  - ⇨per certi versi il contrario di "inline"
  - ⇨introduce una variabile per dare un nome ad un'espressione complessa
  - ⇨prima: `if (biglietto.getClasse() == 2) {...)`
  - ⇨dopo:
```
boolean isPrimaClasse =
                    biglietto.getClasse() == 2;
if (isPrimaClasse) {...}
```


Il Catalogo dei Refactoring

- "Move"
  - ⇨tipicamente applicato ad un metodo
  - ⇨lo sposta in una classe diversa
  - ⇨come extract method ha una meccanica abbastanza delicata
- Infatti
  - ⇨ha lo svantaggio di cambiare l'interfaccia della classe
  - ⇨è opportuno eseguirlo automaticamente

Il Catalogo dei Refactoring

- "Change method signature"
  - ⇨un refactoring frequente
  - ⇨modifica il prototipo di un metodo
  - ⇨per introdurre un nuovo parametro o per rimuoverne uno che esiste
  - ⇨o per cambiare l'ordine dei parametri
  - ⇨è necessario modificare anche tutte le chiamate al metodo
  - ⇨normalmente richiede interventi sul codice

Il Catalogo dei Refactoring

- Altri refactoring automatizzati
  - ⇨sono principalmente mirati ad aumentare la produttività del programmatore
- "Fix imports" (di NetBeans)
  - ⇨rimuove tutte le clausole import inutili e aggiunge quelle mancanti
- "Encapsulate field"
  - ⇨data una proprietà produce i metodi get/set per una proprietà

Il Catalogo dei Refactoring

- "Create delegate method"
  - ⇨un refactoring molto utile
  - ⇨crea in una classe client i metodi per accedere ad un oggetto in associazione
  - ⇨es: Passeggero contiene una lista di biglietti
  - ⇨crea in passeggero i metodi add(), remove(), size() ecc. delegando l'esecuzione alla lista
  - ⇨può richiedere ridenominazioni e piccole modifiche

Il Catalogo dei Refactoring

- "Create constructor"
  - ⇨a partire dalle proprietà selezionate, costruisce il costruttore relativo
- "Minimize access rights"
  - ⇨analizza le chiamate di un metodo e restringe la visibilità al minimo indispensabile (public, friendly, private)

Il Catalogo dei Refactoring

- Ulteriori refactoring
  - ⇨refactoring collegati all'ereditarietà ed al polimorfismo (>>)
  - ⇨refactoring collegati ai design pattern (>>)
- Il catalogo completo
  - ⇨disponibile in linea su refactoring.com
  - ⇨una lettura molto istruttiva e consigliata a tutti i programmatori

Linee Guida per il Refactoring

- La filosofia del refactoring
  - ⇨miglioramento continuo della qualità del codice
  - ⇨la prima organizzazione scelta per il codice non è mai quella perfetta
  - ⇨è opportuno migliorarla continuamente man mano che cresce la conoscenza del prob.
  - ⇨questo processo va condotto in modo sistematico e per piccoli passi

Linee Guida per il Refactoring

- Un aspetto centrale per il refactoring
  - ⇨i test di regressione
- Di conseguenza
  - ⇨il programmatore deve alternare continuamente tre attività diverse ma interconnesse
  - ⇨sviluppo e correzione del codice
  - ⇨scrittura dei test
  - ⇨refactoring del codice

Linee Guida per il Refactoring

- Linea guida n. 1: “i tre cappelli”
  - ⇨queste attività vanno condotte in modo separato (“con tre cappelli diversi”)
  - ⇨quando si fa refactoring non si sviluppa il codice
  - ⇨quando si scrivono i test non si fa refactoring
- Nota
  - ⇨il refactoring naturalmente influenza anche i test oltre che il codice

Linee Guida per il Refactoring

- Vantaggi di questo processo
  - ⇨se adottato sistematicamente migliora sensibilmente la leggibilità del codice
  - ⇨migliora l'organizzazione del codice
- Di conseguenza
  - ⇨migliora la manutenibilità del codice e aiuta ad individuare rapidamente i bachi
  - ⇨in altri termini aiuta a programmare più velocemente

Linee Guida per il Refactoring

- Linea guida n. 2: "non ditelo ai manager"
  - ⇨molti manager di gruppi di sviluppo non hanno una formazione tecnica e considerano la programmazione un compito relativamente semplice
  - ⇨pensano che test di regressione e refactoring siano una perdita di tempo
  - ⇨Fowler consiglia esplicitamente di non discutere con questi manager il processo di sviluppo adottato

Linee Guida per il Refactoring

- Nota
  - ⇨il refactoring continuo ha anche degli svantaggi
- Uno svantaggio notevole
  - ⇨spesso cambia le interfacce dei componenti
  - ⇨questo può avere influsso su altri sottogruppi di lavoro e rendere più instabile il processo
  - ⇨in questo caso va utilizzato in modo controllato

Linee Guida per il Refactoring

- Refactoring e prestazioni
  - ⇨come visto nell'esempio, in alcuni casi l'obiettivo di buona organizzazione confligge con quello di avere codice efficiente
- Esempio tipico
  - ⇨un unico metodo che con una scansione effettua due operazioni su una collezione
  - ⇨due metodi che effettuano separatamente le stesse operazioni con due scansioni

Linee Guida per il Refactoring

- Linea guida n. 3: refactoring e prestazioni
  - ⇨nelle applicazioni attuali il problema delle prestazioni deve essere affrontato solo nel caso in cui si ponga realmente
  - ⇨ovvero quando attraverso i test viene riscontrata una eccessiva lentezza nel codice
  - ⇨in questo caso è opportuno far girare l'applicazione con un profiler (>>) ed analizzare le cause reali di problemi

Linee Guida per il Refactoring

- La logica dietro questo fatto
  - ⇨a differenza della programmazione procedurale, le macchine virtuali dei linguaggi a oggetti sono molto complesse
  - ⇨tranne che nei casi di grossolani errori di progetto, la complessità computazionale tradizionale non aiuta molto a capire le cause di inefficienza
  - ⇨è opportuno studiarle a basso livello quando si presentano

Linee Guida per il Refactoring

- Un esempio dal libro di Fowler
  - in un progetto concreto l'applicazione girava molto lentamente
  - vengono fatte molte ipotesi sulla complessità del codice scritto e sulle collezioni
  - eseguito nel profiler, il problema si rivela essere di tutt'altra natura: la macchina virtuale non riusciva a recuperare correttamente stringhe di media lunghezza (circa 12 KByte) costruite nel codice
  - la soluzione consistette nel cambiare la logica di creazione di queste stringhe

Linee Guida per il Refactoring

- Linea guida n. 4: quando fare refactoring
  - deve essere effettuato sistematicamente
  - ma ci sono momenti in cui è particolarmente opportuno dedicarsi al refactoring
  - prima di aggiungere nuove funzionalità
  - durante, prima o dopo il debugging
  - quando ci si accorge di avere introdotto duplicazioni o ridondanze nel codice

Linee Guida per il Refactoring

- La “regola del tre”
  - ⇨la prima volta che ci si accorge di stare duplicando codice si prosegue senza preoccuparsene troppo
  - ⇨la terza volta, viceversa, si effettua il refactoring e si cerca di eliminare tutte le ridondanze introdotte

Linee Guida per il Refactoring

- Linea guida n. 5: dove fare refactoring
  - ⇨una questione interessante
  - ⇨quali sono i punti del codice che richiedono realmente refactoring ?
- La linee guida di Kent Beck
  - ⇨“il codice cattivo puzza” (“bad code smells”)

"Code Smells"

- Gli "smell" semplici
  - ⇨codice duplicato
  - ⇨metodi lunghi
  - ⇨metodi con molti parametri
  - ⇨classi molto grandi
  - ⇨classi prive di responsabilità
  - ⇨lunghi if/switch nidificati
  - ⇨gerarchie di ereditarietà parallele

"Code Smells"

- Gli altri smell
  - ⇨sono leggermente meno semplici da individuare
- "Divergent changes"
  - ⇨"cambiamenti divergenti": una classe viene modificata spesso e per ragioni diverse
  - ⇨suggerisce che la coesione nella classe non è altissima
  - ⇨potrebbe essere il caso di spezzarla in due

"Code Smells"

- "Shotgun surgery"
  - ⇨"chirurgia con il fucile a pallini": il cambiamento alla classe A richiede quasi sempre un cambiamento a B, C e D
  - ⇨suggerisce un livello di accoppiamento troppo alto tra le classi
  - ⇨è opportuno ridurre il livello di accoppiamento ridistribuendo le responsabilità

"Code Smells"

- "Data clumps" e "Primitive obsession"
  - ⇨gruppi di dati che tendono ad andare sempre assieme
  - ⇨es: coordinate x e y di un punto nel piano > suggerisce di introdurre un oggetto
  - ⇨le classi piccole non sono necessariamente negative se hanno precise responsabilità (in questo caso rappresentare un concetto rilevante del modello)

"Code Smells"

- "Feature envy"
  - ⇨"invidia delle caratteristiche": una classe utilizza prevalentemente i membri di un'altra classe invece che i propri
- "Speculative generality"
  - ⇨"generalità non richiesta": una classe è complicata perchè scritta per essere più generale di quello che serve
  - ⇨è opportuno semplificarne il codice

"Code Smells"

- Idea
  - ⇨lo sviluppatore dovrebbe imparare a riconoscere quando il proprio codice si sta imputridendo
  - ⇨e capire che è il momento di lavorare per il refactoring

Riassumendo

- Il Catalogo dei Refactoring
- Linee Guida per il Refactoring
- “Code Smells”

Termini della Licenza