Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo: Linee Guida

versione 1.0

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

Sommario

La Filosofia di Sviluppo

- In questo corso
  - ⇨una filosofia di sviluppo definita
- Aspetti centrali del processo
  - ⇨sviluppo centrato sui test
  - ⇨refactoring continuo del codice
  - ⇨build automatizzate

La Filosofia di Sviluppo

- Altre caratteristiche importanti
  - ⇨obiettivo di semplicità
  - ⇨adesione a convenzioni di stile
  - ⇨approccio iterativo allo sviluppo
  - ⇨sviluppo centrato sulla scrittura del codice
- Queste pratiche
  - ⇨sono comuni alla cosiddetta “Programmazione Estrema”

eXtreme Programming (XP)

- Programmazione estrema
  - ⇨movimento nato attorno alla metà degli anni 90 per iniziativa di Kent Beck, Ward Cunningham e Ron Jeffries
- In sintesi
  - ⇨un approccio rivoluzionario allo sviluppo del software
  - ⇨che mette in discussione le metodologie tradizionali

eXtreme Programming (XP)

- La vicenda centrale: il sistema C3
  - ⇨“Chrysler Comprehensive Compensation Program”, il sistema di paghe della Daimler-Chrysler
- Al giorno d’oggi
  - ⇨molto diffuso e “chiacchierato”
- Il sito di riferimento
  - ⇨www.extremeprogramming.org

eXtreme Programming (XP)

- XP in pillole
  - ⇨una serie di pratiche e di valori di riferimento
- Nel seguito
  - ⇨una introduzione alla programmazione estrema
  - ⇨basata su una visione “critica” che tende a metterne in luce aspetti positivi e limiti

XP: Le Pratiche Condivisibili

- Le pratiche di XP pienamente condivisibili
  - ⇨sviluppo guidato dai test (“test driven”)
  - ⇨semplicità (“simple design”)
  - ⇨refactoring continuo
  - ⇨build automatizzate e “integrazione continua”
  - ⇨adesione a convenzioni di stile
  - ⇨approccio iterativo e centrato sulla scrittura del codice

XP: Le Pratiche Condivisibili

- Perchè condivisibili ?
  - ⇨perchè l'esperienza dice che hanno un valore aggiunto molto alto nello sviluppo
- Attenzione
  - ⇨per stessa ammissione dei propositori, nessuna delle pratiche dell'XP è originale
  - ⇨si tratta della riproposizione di tecniche esistenti all'interno di una sistematizzazione nuova

XP: Le Pratiche Difficili

- In altri casi
  - ⇨le pratiche sostenute da XP possono avere un certo valore aggiunto
  - ⇨ma si rivelano tipicamente molto difficili da applicare nei contesti aziendali
  - ⇨incontrano opposizione sia nei programmatori
  - ⇨sia, soprattutto, nei manager

XP: Le Pratiche Difficili

- Nel seguito
  - ⇨alcuni esempi
- “Pair programming”
  - ⇨programmazione a coppie; uno sviluppatore XP non lavora mai da solo
  - ⇨in realtà studi dimostrano che è un metodo efficace e favorisce lo sviluppo del codice
  - ⇨ma praticamente irrealizzabile in azienda

XP: Le Pratiche Difficili

- Sviluppo “test first”
  - ⇨prima si scrivono i test e poi il codice: il codice è un tentativo di far girare i test
  - ⇨in linea di principio è un approccio interessante, ma richiede molta disciplina
- Settimana di 40 ore
  - ⇨i programmatori estremi non lavorano mai fino a sfinirsi – 8 ore al giorno è lo standard
  - ⇨difficile in corrispondenza delle scadenze

XP: Le Pratiche Difficili

- Proprietà collettiva del codice
  - ⇨tutti i programmatori del gruppo hanno diritto a modificare in qualsiasi momento tutto il codice
  - ⇨difficile da gestire e da mettere in pratica
- Cliente sempre disponibile sul posto
  - ⇨molto utile per condurre test di accettazione continui, ma in concreto molto raro

XP: Le Pratiche Discutibili

- Le pratiche di XP negative
  - ⇨sono dovute ad un certo “integralismo” nell’intendere il processo di sviluppo
  - ⇨che le rende difficilmente condivisibili
- In particolare
  - ⇨il rifiuto assoluto di tutte le pratiche consolidate di analisi e progettazione del sofware

XP: Le Pratiche Discutibili

- Rifiuto dei modelli
  - ⇨i programmatori estremi non usano UML
  - ⇨considerano poco produttiva la modellazione concettuale e l'analisi del dominio
  - ⇨i programmatori estremi rifiutano anche i casi d'uso tradizionali
- I sostituti
  - ⇨le carte CRC ("class-responsibility-collabor.")
  - ⇨le "user stories"

XP: Le Pratiche Discutibili

- In generale
  - ⇨le uniche attività che abbiano realmente valore per un programmatore estremo sono la scrittura del codice, la scrittura dei test e il refactoring
- L'atteggiamento rispetto ai test
  - ⇨"una funzionalità di un programma per cui non c'è un test di regressione è una funzionalità che non esiste realmente"

Il Giudizio sulla XP

- Per queste ragioni
  - ⇨la XP viene considerata molto controversa
- Una indagine della IBM del 2000
  - ⇨“What are your thoughts on XP ?”
  - ⇨l’ho provata e la detesto: 8%
  - ⇨è una pessima idea, non funzionerà: 16%
  - ⇨è una buona idea, ma non funzionerà: 25%
  - ⇨l’ho provata e mi affascina: 51%

Il Giudizio sulla XP

- In altri termini
  - ⇨gli sviluppatori sono molto divisi in merito
  - ⇨anche chi crede che l’idea sia buona poi si scontra contro difficoltà oggettive
- Una critica diffusa
  - ⇨la programmazione estrema funziona solo per programmatori molto abili
  - ⇨non è adatta a programmatori “normali”

Il Giudizio sulla XP

ATTENZIONE
al rapporto
con la XP

- Il nostro approccio
  - ⇨conserviamo le pratiche migliori della programmazione estrema
  - ⇨evitando di adottarne quelle più discutibili
  - ⇨in modo da cercare di inserire la filosofia “agile” in un processo di sviluppo compatibile con quello standard
  - ⇨che funga da supporto a tutte le categorie di sviluppatori

Un Episodio di Programmazione

- In concreto
  - ⇨per dimostrare concretamente analogie e differenze tra i due metodi proposti illustriamo un caso di studio
- I punteggi del bowling
  - ⇨è necessario scrivere il sistema informativo di una sala bowling
  - ⇨che calcoli i punteggi ottenuti dai giocatori nelle varie piste

Un Episodio di Programmazione

- I punteggi del bowling
  - ⇨logica applicativa complicata
- Le regole
  - ⇨la partita si gioca tra due giocatori o due squadre
  - ⇨ciascun giocatore gioca 10 “frame”
  - ⇨in ogni frame l’obiettivo è buttare giù i 10 birilli sulla pista

Un Episodio di Programmazione

- Le regole (continua)
  - ⇨il giocatore ha a disposizione due palle
  - ⇨a seconda dei birilli abbattuti con le due palle si ottengono punteggi diversi
- I caso: birilli rimanenti
  - ⇨il giocatore non abbatte tutti i birilli
  - ⇨in questo caso il suo punteggio è pari ai birilli abbattuti

Un Episodio di Programmazione

- II caso: “spare”
  - ⇨il giocatore abbatte tutti i birilli con la seconda palla (es: 3 + 7 birilli abbattuti)
  - ⇨si tratta di uno “spare”
  - ⇨in questo caso il suo punteggio è pari ai birilli abbattuti (10), più i birilli abbattuti con la palla successiva
  - ⇨ovvero la prima palla del prossimo frame
  - ⇨o un tiro aggiuntivo se siamo nel 10 frame

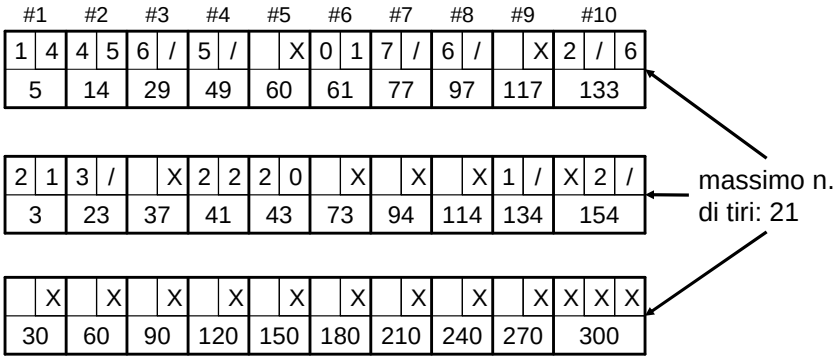
Un Episodio di Programmazione

- III caso: “strike”
  - ⇨il giocatore abbatte tutti i birilli con la prima palla: si tratta di uno “strike”
  - ⇨il frame si conclude con la prima palla
  - ⇨il suo punteggio è pari a 10 più i birilli abbattuti con le due palle successive
  - ⇨le due palle del prossimo frame
  - ⇨o il prossimo strike più la palla successiva
  - ⇨o tiri aggiuntivi se siamo nel 10 frame

Un Episodio di Programmazione

- Alcune “score card”

/ : spare
X : strike



“perfect game”: 12 strike

Un Episodio di Programmazione

- Una soluzione XP
  - ⇨“An eXtreme Programming Episode”
  - ⇨un articolo famoso di Robert Martin pubblicato su objectmentor.com e poi nel libro “Agile Software Development”
  - ⇨racconta come due programmatori estremi risolvono il problema dei punteggi del bowling

>> materiale\xepisode.htm

Un Episodio di Programmazione

- Caratteristiche della soluzione
  - ⇨le tre attività di sviluppo del codice, scrittura dei test e refactoring sono state altrernate sistematicamente
  - ⇨i test vengono scritti prima dei metodi relativi
  - ⇨l'attività di modellazione è stata ridotta al minimo e non ha avuto praticamente nessun impatto reale sulla soluzione finale (per stessa ammissione di Robert Martin)

Un Episodio di Programmazione

- Caratteristiche della soluzione (continua)
  - ⇨inoltre, la soluzione nasce più o meno improvvisamente per iniziativa di uno dei due programmatori
  - ⇨e poi viene raffinata
  - ⇨non viene realmente progettata
- Il commento di Robert Martin
  - ⇨"per questo programma i diagrammi sono inappropriati"

Una Soluzione Alternativa

- Una soluzione alternativa
  - ⇨il package it.unibas.bowling
  - ⇨sviluppato indipendentemente applicando il processo di sviluppo illustrato finora
- Caratteristiche simili
  - ⇨test e refactoring
- Differenza
  - ⇨il processo è guidato dall'analisi

Una Soluzione Alternativa

- Riassumiamo il processo di sviluppo
  - ⇨analizzare le specifiche e scrivere i casi d'uso
  - ⇨costruire il modello concettuale
  - ⇨scegliere i componenti
  - ⇨attribuire le responsabilità ai componenti
  - ⇨scrivere il codice dei metodi

Una Soluzione Alternativa

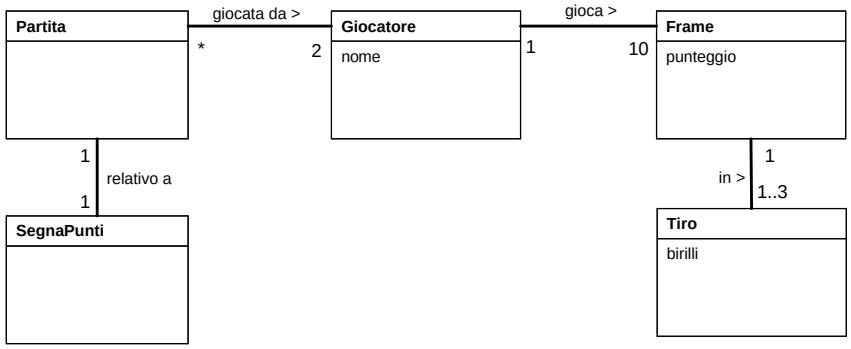
- I casi d'uso
  - ⇨"Giocatori giocano partita"
- "Giocatori giocano partita"
  - ⇨i due giocatori selezionano una pista
  - ⇨a turno effettuano i tiri
  - ⇨il sistema aggiorna progressivamente i punteggi di ciascun giocatore
  - ⇨al termine della partita decreta il vincitore

Una Soluzione Alternativa

- Una annotazione importante
  - ⇨è possibile calcolare il punteggio di un frame solo dopo che il frame si è effettivamente concluso
  - ⇨se il giocatore ha ottenuto uno spare, dopo il tiro successivo; se il giocatore ha ottenuto uno strike, dopo i due tiri successivi
  - ⇨quindi il sistema aggiorna i punteggi solo alla conclusione dei vari frame

Una Soluzione Alternativa

- Un possibile modello concettuale



Una Soluzione Alternativa

- Le pratiche di programmazione
  - ⇨cominciamo dai componenti del modello (controllo e interfaccia alla fine)
  - ⇨selezionando per cominciare quelli di maggior rischio
  - ⇨stabiliamo un piano di test e guidiamo lo sviluppo del modello con i test
  - ⇨utilizziamo il sistema di logging
  - ⇨e uno stile di programmazione difensiva

Una Soluzione Alternativa

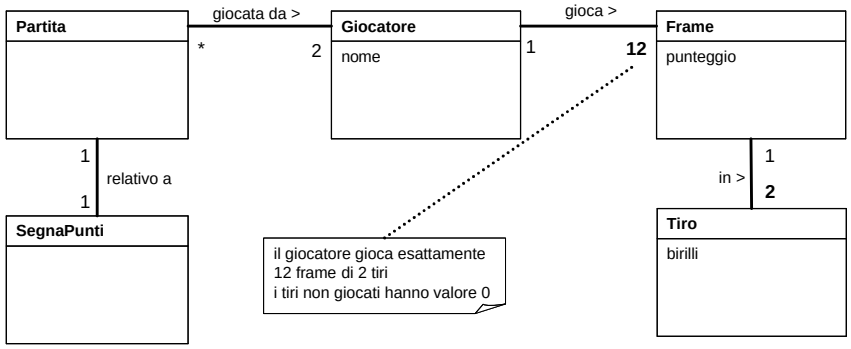
- In questo caso
  - ⇨scartiamo la classe Tiro
  - ⇨dubbi su SegnaPunti
  - ⇨sviluppiamo per cominciare Frame e Giocatore
  - ⇨per risolvere il problema del calcolo dei punteggi della partita
  - ⇨i casi di test corrispondono alle score card

Una Soluzione Alternativa

- Una intuizione importante durante lo sviluppo
  - ⇨per lavorare con i frame è possibile vedere la partita come fatta di un massimo di 12 frame
  - ⇨gli ultimi due frame corrispondono agli eventuali tiri aggiuntivi
  - ⇨il punteggio del giocatore è quello ottenuto al 10 frame
  - ⇨in quest'ottica ogni frame si può considerare composto di esattamente due tiri (il secondo eventualmente pari a 0 birilli)

Una Soluzione Alternativa

- Il modello concettuale corretto



Una Soluzione Alternativa

- Dopo vari refactoring
  - ⇨it.unibas.bowling.modello.base
- Le classi
  - ⇨Frame.java
  - ⇨Giocatore.java
  - ⇨TestGiocatore.java
  - ⇨TestFrame.java

>> it.unibas.bowling.modello.base

Una Soluzione Alternativa

- Confronto con la soluzione di Martin
  - ⇨le classi sono più aderenti al modello concettuale
  - ⇨in totale 126 linee di codice rispetto alle 113 di Martin (11% in più)
- Nota
  - ⇨per rendere possibile il confronto sono state rimosse tutte le istruzioni di logging e di programmazione difensiva

Una Soluzione Alternativa

- La versione finale
  - ⇨it.unibas.bowling.modello
  - ⇨introduce la classe SegnaPunti che ha la responsabilità di produrre la score card
  - ⇨introduce la classe Partita che consente di giocare una intera partita tra i due giocatori

>> it.unibas.bowling.modello

Riassumendo

- La Filosofia di Sviluppo
- eXtreme Programming (XP)
  - ⇨XP: Le Pratiche Condivisibili
  - ⇨XP: Le Pratiche Difficili
  - ⇨XP: Le Pratiche Discutibili
  - ⇨Il Giudizio sulla Programmazione Estrema
- Un Episodio di Programmazione
- Una Soluzione Alternativa

Termini della Licenza