Design Pattern e Code Smell: Classificazione e Definizioni ↗

Apri questa trascrizione su Scriba

2 sezioni · 14 termini · 20 flashcard

> *Il corso definisce design pattern come soluzioni comuni a problemi ricorrenti di progettazione, classifica i GOF in creazionali, strutturali e comportamentali. Vengono introdotti bloaters, object orientation abusers e change preventers.*

INDICE

PUNTI CHIAVE

01 Pattern di Progettazione e Code Smell

- Definizione di design pattern: soluzione comune a un problema ricorrente di progettazione.
- Classificazione GOF (Gang of Four): 23 pattern totali, sette trattati nel corso.
- Pattern creazionali, strutturali e comportamentali.
- Esempi di combinazioni: factory method (creazionale su classi), adapter (strutturale), observer e strategy (comportamentali su oggetti).
- Strategy pattern: scelta tra algoritmi correlati a runtime.
- Observer pattern: notifica di cambiamenti a più interessati senza dipendenza diretta.
- Adapter vs. Facade: adapter converte interfacce esistenti, facade semplifica accesso a sottosistemi complessi.
- Singleton pattern: una sola istanza globale di una classe con metodi statici e costruttore privato.
- Limiti del singleton: sincronizzazione su getInstance(), non funziona bene nelle applicazioni distribuite.
- Code smell: caratteristiche del codice che indicano cattive pratiche, ma non necessariamente bug.
- Refactoring: riscrittura del codice senza cambiare il comportamento esterno.

02 Code Smell: Famiglie e Definizioni

- I bloaters sono codici sproporzionati come long method, large class, long parameter list.
- Gli object orientation abusers sono l'uso sbagliato dell'OOP come switch statement e refused bequest.
- I change preventers impediscono le modifiche come shotgun surgery.
- I dispensables sono parti superflue come duplicated code, data class, comments.
- I couplers hanno accoppiamento eccessivo come feature envy.

GLOSSARIO

design pattern — Soluzione comune a un problema ricorrente di progettazione.

GOF — Acronimo per Gang of Four, riferimento ai quattro ideatori dei 23 pattern standard.

strategy pattern — Pattern comportamentale che permette di scambiare algoritmi a runtime attraverso un'interfaccia comune.

observer pattern — Pattern comportamentale che notifica cambiamenti a più interessati senza dipendenza diretta tra loro.

adapter pattern — Pattern strutturale che converte interfacce incompatibili per permettere la comunicazione tra sistemi diversi.

facade pattern — Pattern strutturale che semplifica l'accesso a un sottosistema complesso attraverso una singola interfaccia.

singleton pattern — Pattern creazionale che garantisce la presenza di una sola istanza globale di una classe con metodi statici e costruttore privato.

code smell — Caratteristiche del codice che indicano cattive pratiche, ma non necessariamente bug.

refactoring — Riscrittura del codice senza cambiare il comportamento esterno a piccoli passi.

long method — Un metodo con troppe righe che dovrebbe essere diviso in metodi separati.

refused bequest — La sottoclasse non usa quello che eredita, indicando una gerarchia errata.

shotgun surgery — Una modifica piccola richiede modifiche in molte classi diverse.

data class — Classi che sono solo contenitori di dati senza logica.

feature envy — Un metodo usa più i dati di un altro oggetto che i propri.

FLASHCARD

Che cos'è un design pattern?

Una soluzione comune a un problema di progettazione ricorrente.

Cosa rappresentano i bloaters?

Codice sproporzionato come long method, large class, long parameter list.

Per cosa sta GOF?

Gang of Four

Che cosa sono gli object orientation abusers?

L'uso sbagliato dell'object oriented come lo switch statement e refused bequest.

Quanti pattern GOF ci sono in totale?

23

Cosa significano i change preventers?

Parti troppo accoppiate che impediscono le modifiche, ad esempio shotgun surgery.

Quali sono i due assi di classificazione dei design pattern?

Scopo e raggio d'azione.

Quali sono gli dispensables?

Cose che è meglio togliere come duplicated code e data class.

Cosa significa un pattern creazionale su classi?

Un pattern che si occupa della creazione di oggetti in modo da isolare la logica di creazione dal codice client.

Cosa è un long method?

Un metodo con troppe righe che dovrebbe essere diviso in metodi separati.

Qual è il problema risolto dallo strategy pattern?

Un insieme di algoritmi correlati e variabili che vogliamo poter cambiare a runtime.

Che cosa significa refused bequest?

La sottoclasse non usa quello che eredita, quindi la gerarchia è sbagliata.

Come si risolve il problema dello strategy pattern?

Ogni algoritmo viene messo in una classe separata, tutte con un'interfaccia comune. Il context tiene un riferimento alla strategia e la usa senza sapere quale sia.

Cosa rappresenta shotgun surgery?

Per una modifica piccola devi toccare tante classi diverse creando problemi di modifiche e rischi di errori non immediatamente evidenti.

Qual è la distinzione tra strategy e observer pattern?

Lo strategy serve a scambiare un comportamento, mentre observer serve a notificare un cambiamento a più interessati.

Che cosa sono le data class?

Classi che sono solo contenitori di dati ma senza logica.

Cosa risolve l'adapter pattern?

Un'incompatibilità tra due interfacce incompatibili che però hanno bisogno di parlarsi.

Cosa è il feature envy?

Un metodo usa più i dati di un altro oggetto che i propri.

Qual è la differenza tra adapter e facade pattern?

L'adapter converte un'interfaccia che c'è già, mentre facade semplifica l'accesso a un insieme di cose.

Come si risolve il switch statement?

Sfruttare il polimorfismo con replace conditional with polymorphism.

SIMULAZIONE D'ESAME

> *Domande generate automaticamente dalla registrazione: usale per ripassare, ma verifica le risposte sugli appunti prima di darle per certe.*

1. Quali sono le tre classificazioni principali dei design pattern?

- **A** diagrammi UML
- **B** metodologie agile
- **C** pattern GRASP
- **D** creazionali, strutturali e comportamentali

2. Qual è il sintomo del long method?

- **A** Metodo che usa dati di un altro oggetto
- **B** Una classe troppo grande
- **C** Un switch statement eccessivo
- **D** Un metodo con troppe righe che dovrebbe essere diviso in metodi separati.

3. Qual è il problema che risolve il pattern Strategy?

- **A** un insieme di algoritmi correlati e variabili che vogliono poter essere cambiati
- **B** complessità in un sottosistema
- **C** incompatibilità tra interfacce esistenti
- **D** più oggetti che vogliono sapere quando un altro cambia stato

4. Qual è il sintomo del refused bequest?

- **A** La sottoclasse non utilizza quello che eredita, indicando una gerarchia errata.
- **B** Una classe contenitore di dati senza logica
- **C** Metodo che usa più i dati di un altro oggetto
- **D** Un metodo con troppe righe

5. Qual è il problema che risolve il pattern Observer?

- **A** un insieme di algoritmi correlati e variabili
- **B** complessità in un sottosistema
- **C** più oggetti che vogliono sapere quando un altro cambia stato
- **D** incompatibilità tra interfacce esistenti

6. Che cosa rappresenta il shotgun surgery?

A Un metodo con troppe righe
B Una sottoclasse che non usa l'ereditarietà correttamente
C Per una piccola modifica si devono modificare molte classi diverse.
D Metodo che utilizza più dati di un altro oggetto

7. Qual è la differenza principale tra Adapter e Facade?

A singleton è una sola istanza di una classe con un punto di accesso globale
B l'adapter converte un'interfaccia che c'è già, mentre facade semplifica l'accesso a un insieme di cose
C observer serve a notificare un cambiamento a più interessati
D strategy serve a scambiare un comportamento

8. Che cosa sono le data class?

A Sottoclasse che non usa l'ereditarietà correttamente
B Classi contenitori di dati senza logica.
C Metodo che utilizza più dati di un altro oggetto
D Metodo con troppe righe

9. Quali sono i due limiti principali del pattern Singleton?

A observer serve a notificare un cambiamento a più interessati
B serve synchronized su getInstance oppure con più thread rischiamo di avere due o più istanze, non funziona bene nelle applicazioni distribuite
C adapter converte un'interfaccia che c'è già
D strategy serve a scambiare un comportamento

10. Che cosa è il feature envy?

A Un metodo che utilizza più i dati di un altro oggetto che quelli del proprio.
B Una classe troppo grande
C Per una piccola modifica si devono modificare molte classi diverse
D Sottoclasse che non usa l'ereditarietà correttamente

SOLUZIONI

1-D · 2-D · 3-A · 4-A · 5-C · 6-C · 7-B · 8-B · 9-B · 10-A