Refactoring

Un esempio pratico nel nostro codice

Requisiti

- Un paziente può avere diritto a più esenzioni dalla spesa o a nessuna
- Esenzioni diverse possono esentare totalmente o parzialmente una prestazione medica
- Le esenzioni possono avere regole di applicabilità diverse a seconda della regione
- Non tutte le esenzioni possono essere applicate su una prestazione
- Ordinare le esenzioni in modo da selezionare quella applicabile e la più conveniente

Situazione attuale

Algoritmo di ordinamento con smell code:

- Large class
- Long method
- Switch statement
- Divergent change: se vogliamo aggiungere un nuovo tipo di esenzione?
- Comments
- Duplicate code
- Feature envy

Principi SOLID

●**Open/closed**: poter aggiungere nuovi filtri o nuovi tipi di esenzione senza dover ricontrollare tutto il codice
●**Liskov substitution:** i filtri devono poter essere usati in qualsiasi regione, semplicemente aggiungendoli a un array e devono poter essere sostituiti senza bisogno di ulteriori controlli esterni
●**Interface segregation**: i filtri che richiedono logica più complessa, non devono rendere più difficile l'implementazione dei filtri più semplici costringendoli a implementare metodi di cui non hanno bisogno

Conclusioni

- Maggiore leggibilità del codice
- Nessuna duplicazione di funzionalità
- Possibilità di espandere le funzionalità popolando un array
- Ridotte le dimensioni della classe principale
- Minor riduzione delle righe di codice totali, ma il vantaggio si vedrà quando saranno implementate tutte le regole

Sviluppi futuri

Ulteriori sviluppi verranno analizzati proseguendo gli sviluppi regionali.

Lo scopo di questo refactoring era di velocizzare i futuri sviluppi permettendo di popolare gli array regionali e di rendere più semplice la creazione di nuovi filtri

Grazie per l’attenzione