1) Número de classes do projeto original e do refatorado.

Original: 7 classes

Refatorado: 12 classes

2) Número de linhas de códigos do projeto original e refatorado.

Original: 328

Refatorado: 491

3) Analise as seguintes métricas dos dois projetos, e discuta o que significa a medida e o impacto nos projetos o resultado da medida (Utilize o plugin Project Usus para capturar as métricas).

- a. Average componente dependency

A dependência média geral dos componentes na base de código refatorada diminuiu de 2,4 para 2,3.

A classe Principal agora depende apenas de outros dois componentes: ServicoCriarEstacionamento e Menu. Esta é uma redução significativa na sua contagem de dependências, uma vez que anteriormente dependia de quatro outros componentes. As classes ServicoCriarEstacionamento e ServicoConsultarVagas agora estão isoladas do restante da base de código. Cada um deles depende apenas de um outro componente, Vaga. Isso os torna mais fáceis de testar e manter isoladamente. A classe ServicoEstacionamento passa a ter uma responsabilidade mais focada, pois é responsável apenas pelas operações de estacionamento. Anteriormente tinha dependências das classes Registro e ServicoRegistro, mas essas dependências foram removidas na base de código refatorada.

A base de código refatorada tem uma dependência média de componentes mais baixa, o que a torna mais modular e mais fácil de manter.

- b. Class Size

Os tamanhos gerais das classes na base de código refatorada permanecem semelhantes aos da base de código original.

A classe Veiculo foi reduzida em tamanho ao passar parte de sua lógica para a classe ServicoEstacionamento. Isso melhora a organização do código e a separação de interesses. As classes ServicoCriarEstacionamento, ServicoConsultarVagas, ServicoThread e ServicoHora permanecem pequenas e focadas em suas responsabilidades específicas.

A base de código refatorada mantém os tamanhos de classe razoáveis da base de código original, ao mesmo tempo que melhora a organização do código e a separação de interesses.

- c. Cyclomatic complexity

A base de código refatorada tem uma complexidade ciclomática significativamente menor do que a base de código original.

O principal motivo da redução da complexidade ciclomática é a remoção da instrução switch no método escolherOpcaoMenu. Essa instrução switch foi substituída por uma série de instruções if, o que reduziu o número de pontos de decisão no código.

Outras mudanças que podem ter contribuído para a redução da complexidade ciclomática incluem:

Extraindo funcionalidades comuns em métodos reutilizáveis

Dividir código complexo em funções menores e mais gerenciáveis

Usar instruções condicionais com moderação e evitar instruções condicionais aninhadas

A base de código refatorada tem uma complexidade ciclomática menor que a base de código original. Isso torna o código mais fácil de entender, manter, testar e executar.

- d. Lack of cohesion of classes

A base de código original apresenta uma grande falta de coesão de classes. Isto é evidente das seguintes maneiras:

A classe ServicoVagas combina gerenciamento de estacionamento e funcionalidades relacionadas ao tempo.

A classe ServicoRegistro combina cálculo de taxas e funcionalidade de gerenciamento de registros.

A interface do Menu combina apresentação do menu, tratamento de entradas do usuário, operações de estacionamento e validação de entradas.

A base de código refatorada melhora a coesão das classes, dividindo algumas das responsabilidades em classes separadas. Por exemplo:

A interface ServicoRegistro é dividida em uma classe ParkingFeeCalculator e uma classe ParkingRecordManager.

A classe ServicoEstacionamento pode ser dividida em uma classe ParkingManager e uma classe VehicleSearch.

A classe ServicoConsultarVagas pode ser dividida em uma classe ParkingSpaceAvailability e uma classe ParkingSpaceOccupancy.

A classe ServicoHora poderia fazer parte de uma classe mais ampla de validação de entrada ou interação do usuário.

A base de código refatorada tem uma falta de coesão de classes média a baixa em comparação com a base de código original. Isto se deve à separação de responsabilidades em classes menores e mais focadas.

- e. Method length

Os comprimentos dos métodos nas bases de código originais e refatoradas são geralmente razoáveis. Não existem métodos excessivamente longos que possam prejudicar a legibilidade ou a manutenção do código.

As interfaces ServicoRegistro e ServicoVagas na base de código original contém alguns métodos que são relativamente longos, mas isso se deve à complexidade das tarefas que executam. A interface Menu na base de código refatorada contém alguns métodos que também são relativamente longos, mas isso se deve novamente à complexidade das tarefas que eles executam, como exibir o menu principal e lidar com a entrada do usuário.

As bases de código original e refatorada têm comprimentos de método razoáveis. É importante observar que não existe um comprimento de método perfeito para todas as bases de código. A duração ideal de um método dependerá dos requisitos específicos e da complexidade da tarefa que ele executa. No entanto, em geral, é benéfico ter métodos curtos e fáceis de entender.

- f. Number of non-static, non-final public fields

A base de código refatorada tem menos campos públicos não estáticos e não finais do que a base de código original. Os campos removidos eram das interfaces Registro e Menu, que não precisam possuir campos. Os demais campos públicos não estáticos e não finais estão nas classes Vaga e Veiculo, que representam vagas de estacionamento e veículos, respectivamente. Esses campos são essenciais para representar o estado desses objetos.

A base de código refatorada tem menos campos públicos não estáticos e não finais do que a base de código original, o que torna o código mais conciso e fácil de entender.

- g. Package size

Os tamanhos dos pacotes nas bases de código originais e refatoradas são geralmente razoáveis. O pacote br.com.fourcamp.fourpark.service em ambas as bases de código é o maior, contendo a maior parte da lógica e funcionalidade do programa. Os outros pacotes em ambas as bases de código contêm principalmente classes de modelo de dados e a classe principal, que é relativamente menor em tamanho.

Tanto as bases de código originais quanto as refatoradas têm tamanhos de pacote razoáveis. O tamanho de um pacote é apenas um aspecto da organização do código, e é importante considerar as responsabilidades das classes dentro de cada pacote e seus relacionamentos para garantir uma base de código bem estruturada e de fácil manutenção.

- h. Package with cyclic dependencies

O único pacote com dependências cíclicas tanto na base de código original quanto na refatorada é br.com.fourcamp.fourpark.service. A base de código refatorada mantém as dependências cíclicas do pacote br.com.fourcamp.fourpark.service, mas agora estão limitadas a um número menor de classes e interfaces. Tanto a base de código original quanto a refatorada possuem dependências cíclicas no pacote br.com.fourcamp.fourpark.service. No entanto, a base de código refatorada possui menos dependências cíclicas neste pacote.

- i. Unreferenced classes

A base de código original possui mais classes não referenciadas do que a base de código refatorada. A base de código refatorada possui apenas uma classe não referenciada, ServicoThread. A base de código refatorada possui menos classes não referenciadas do que a base de código original. Isso torna a base de código mais concisa e fácil de entender.