Relatório de Refatoração de Código

1. Code Smells Identificados:

Classe Car:

- Convenção de Nomes: Métodos com nome iniciando com letra maiúscula.
- Comentários: Uso de comentários desnecessários.
- Falta de Encapsulamento: Atributos públicos.

Classe Parking Lot:

- Long Method: Função main com muitas responsabilidades.
- Code Duplication: Duplicação de código para exibição dos carros.
- Excessive Comments: Comentários redundantes.
- Nested Conditionals: Muitos condicionais aninhados.
- Magic Numbers: Números mágicos para comprimento de entrada.
- Unused Code: Trecho de código comentado não utilizado.
- Nome Convention: Nomes de atributos/métodos com letra maiúscula.
- Falta de Pacote: O código não está dentro de um pacote.

Classe TotalTime:

- Long Method: Função main com muitas responsabilidades.
- Code Duplication: Duplicação de código para exibição dos carros.
- Excessive Comments: Comentários redundantes.
- Nested Conditionals: Muitos condicionais aninhados.
- Magic Numbers: Números mágicos para comprimento de entrada.
- Unused Code: Trecho de código comentado não utilizado.
- Name Convention: Nomes de atributos/métodos com letra maiúscula.
- Falta de Pacote: O código não está dentro de um pacote.

Outras Classes:

- Nome Convention: Nomes de atributos/métodos com letra maiúscula.
- Magic Numbers: Números mágicos sem explicação.
- Falta de Encapsulamento: Atributos públicos.
- Unused Imports: Importações não utilizadas.
- Lack of Meaningful Variable Names: Nomes de variáveis sem significado claro.

- Feature Envy: Uso excessivo de atributos de outra classe.

2. Dificuldades na Refatoração:

A refatoração envolveu algumas dificuldades, especialmente na quebra de métodos longos e na eliminação de condicionais aninhados. Além disso, a alteração de convenções de nomenclatura exigiu atenção para garantir que a lógica não fosse afetada. A identificação e remoção de código não utilizado também foi um desafio, mas as ferramentas auxiliaram nesse processo.

3. Melhoria no Código:

- A refatoração trouxe várias melhorias ao código:
- Legibilidade: A alteração das convenções de nomenclatura tornou o código mais legível e aderente aos padrões.
- Encapsulamento: A correção da falta de encapsulamento aumentou a segurança do código e evitou o acesso direto aos atributos.
- Organização: A divisão de métodos longos e a eliminação de duplicação de código melhoraram a organização e facilitaram a manutenção.
- Redução de Code Smells: A remoção de condicionais aninhados, números mágicos e código não utilizado reduziu os code smells, tornando o código mais robusto.
- Clareza de Comentários: Comentários desnecessários foram removidos, resultando em um código mais autoexplicativo.

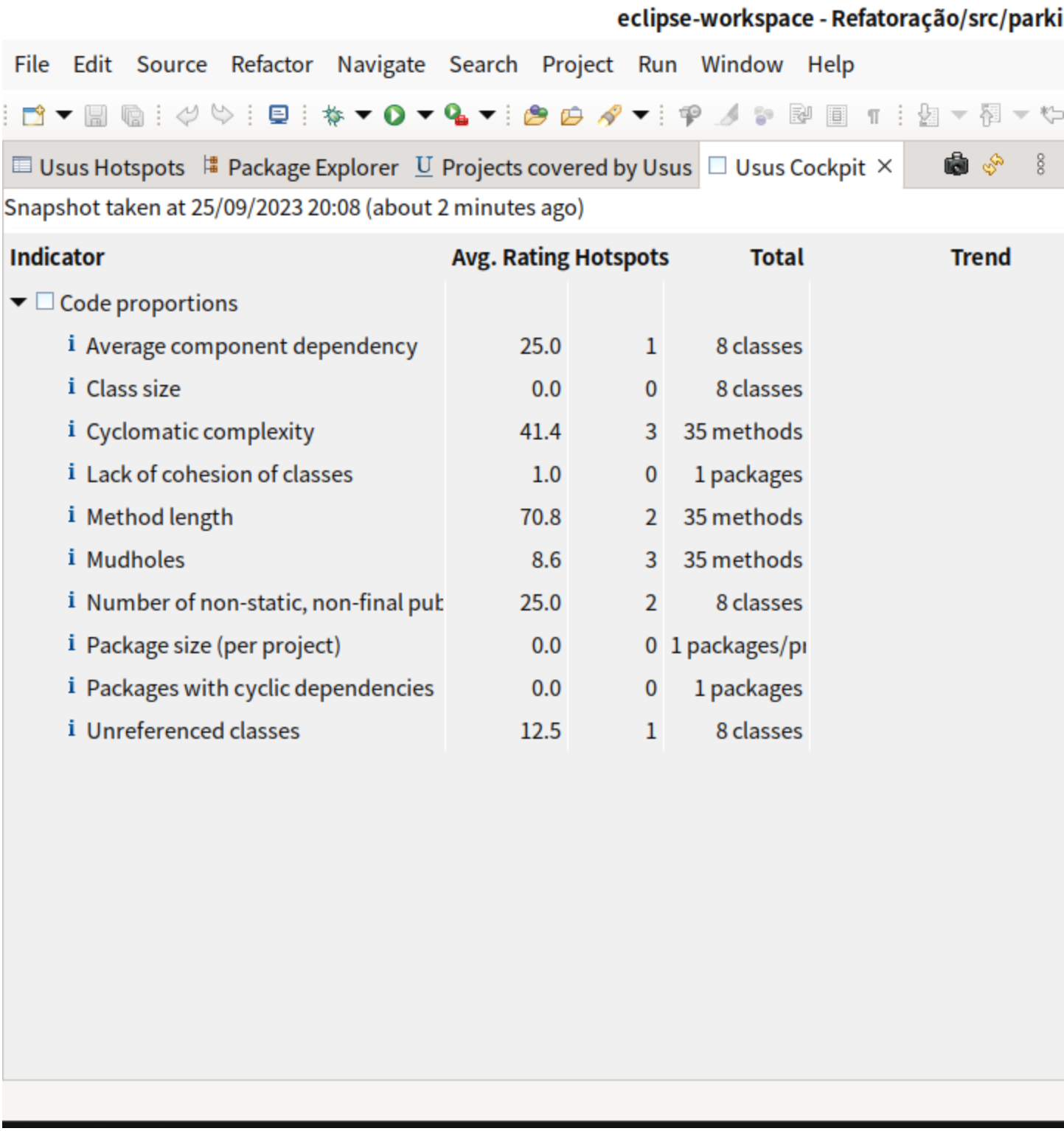
4. Uso da ferramenta

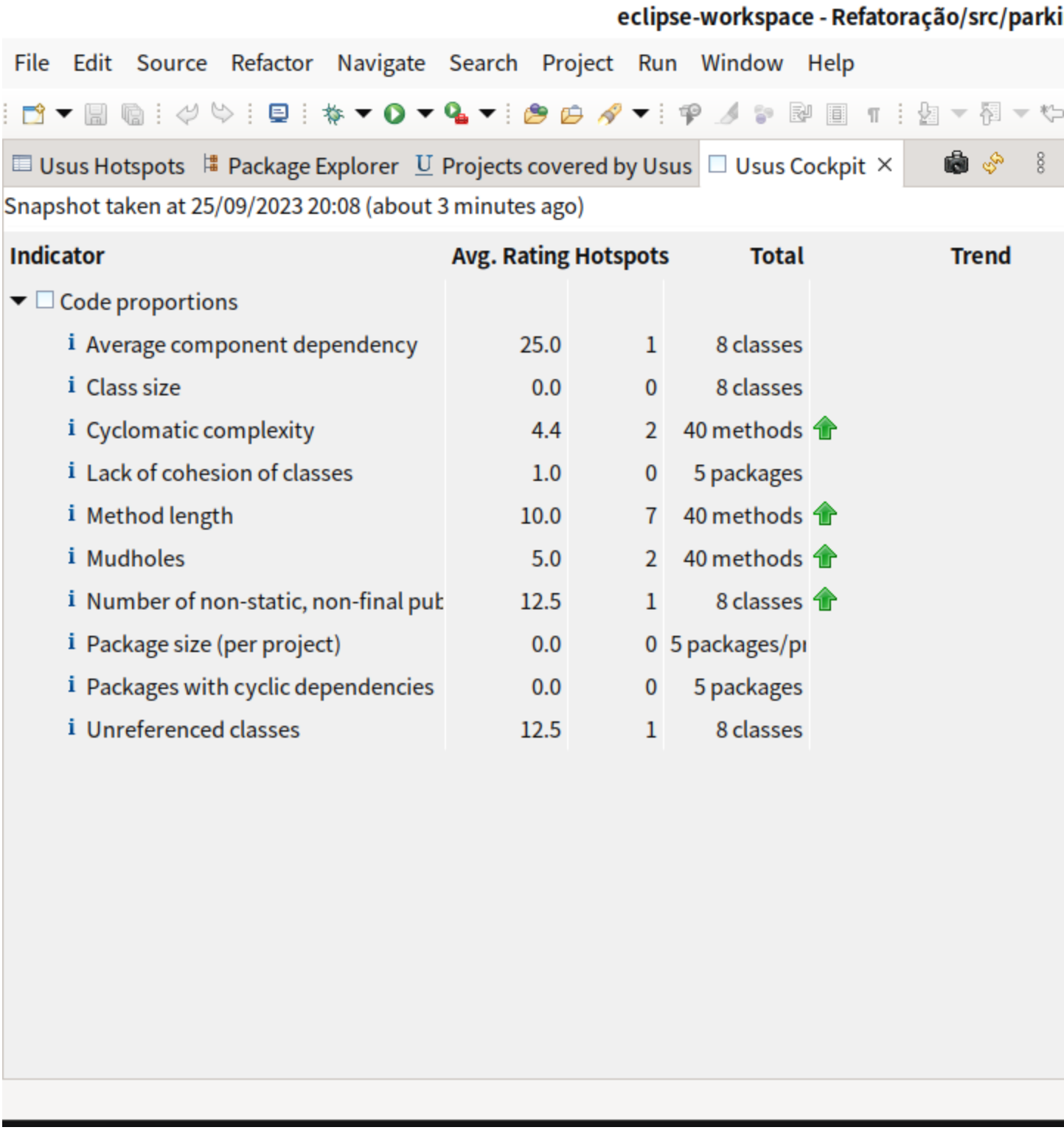
Para efeitos de comparação foi utilizado o project usus onde foi avaliado algumas métricas:

a. Average componente dependency
b. Class Size
c. Cyclomatic complexity
d. Lack of cohesion of classes
e. Method length
f. Number of non-static, non-final public fields
g. Package size
h. Package with cyclic dependencies
i. Unreferenced classes

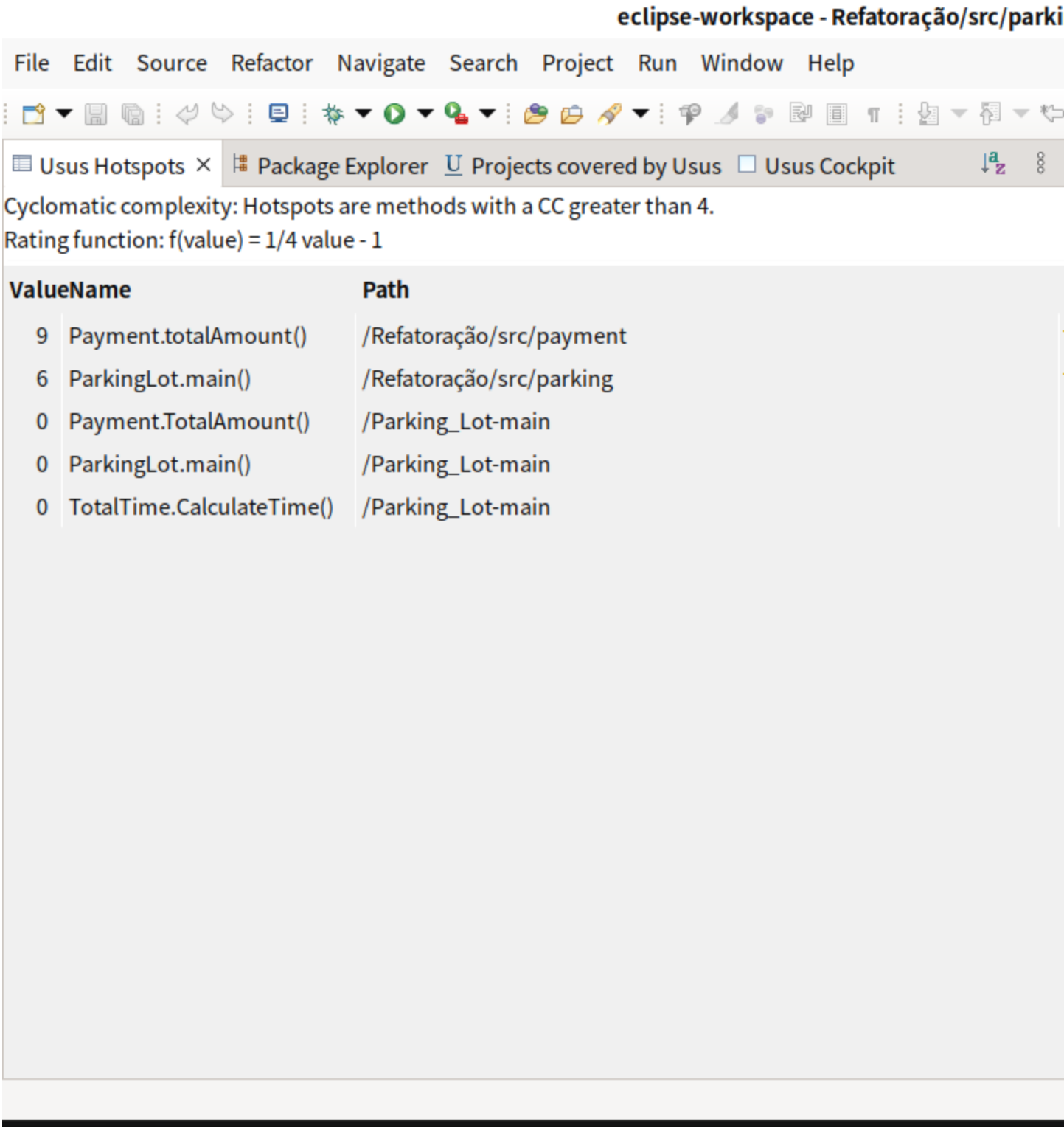
Com o uso da ferramenta podemos concluir que houve uma melhora significativa na refatoração feita, abaixo algumas imagens que ilustram o uso da ferramenta para a

comparação, onde a primeira imagem é a snapshot do código original e a imagem abaixo a comparação com o refatorado

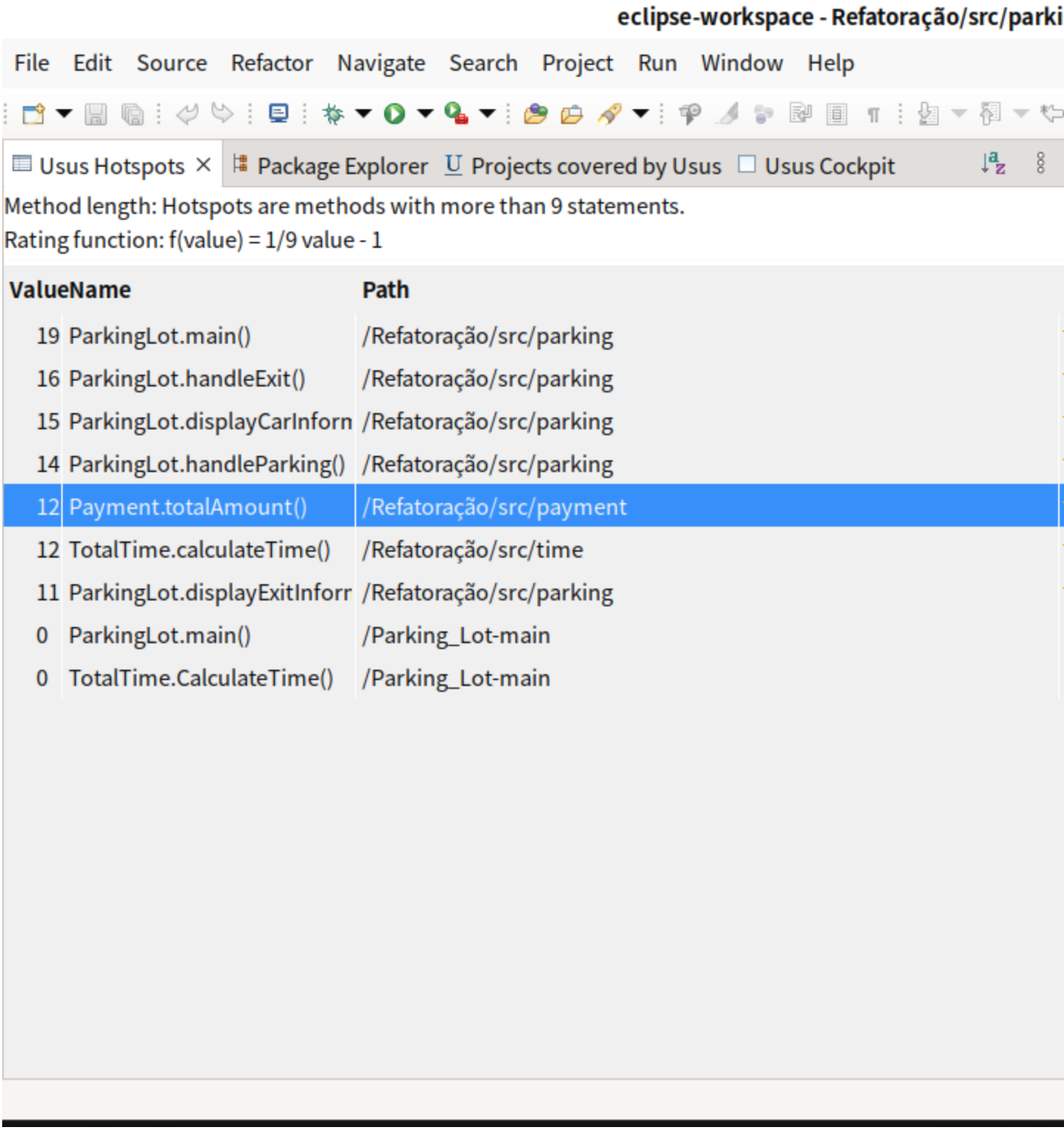

eclipse-workspace - Refatoração/src/parki
File Edit Source Refactor Navigate Search Project Run Window Help
Usus Hotspots
Package Explorer
Projects covered by Usus
Usus Cockpit
Snapshot taken at 25/09/2023 20:08 (about 2 minutes ago)
Indicator
Avg. Rating
Hotspots
Total
Trend
Code proportions
Average component dependency 25.0 1 8 classes
Class size 0.0 0 8 classes
Cyclomatic complexity 41.4 3 35 methods
Lack of cohesion of classes 1.0 0 1 packages
Method length 70.8 2 35 methods
Mudholes 8.6 3 35 methods
Number of non-static, non-final pub 25.0 2 8 classes
Package size (per project) 0.0 0 1 packages/pı
Packages with cyclic dependencies 0.0 0 1 packages
Unreferenced classes 12.5 1 8 classes



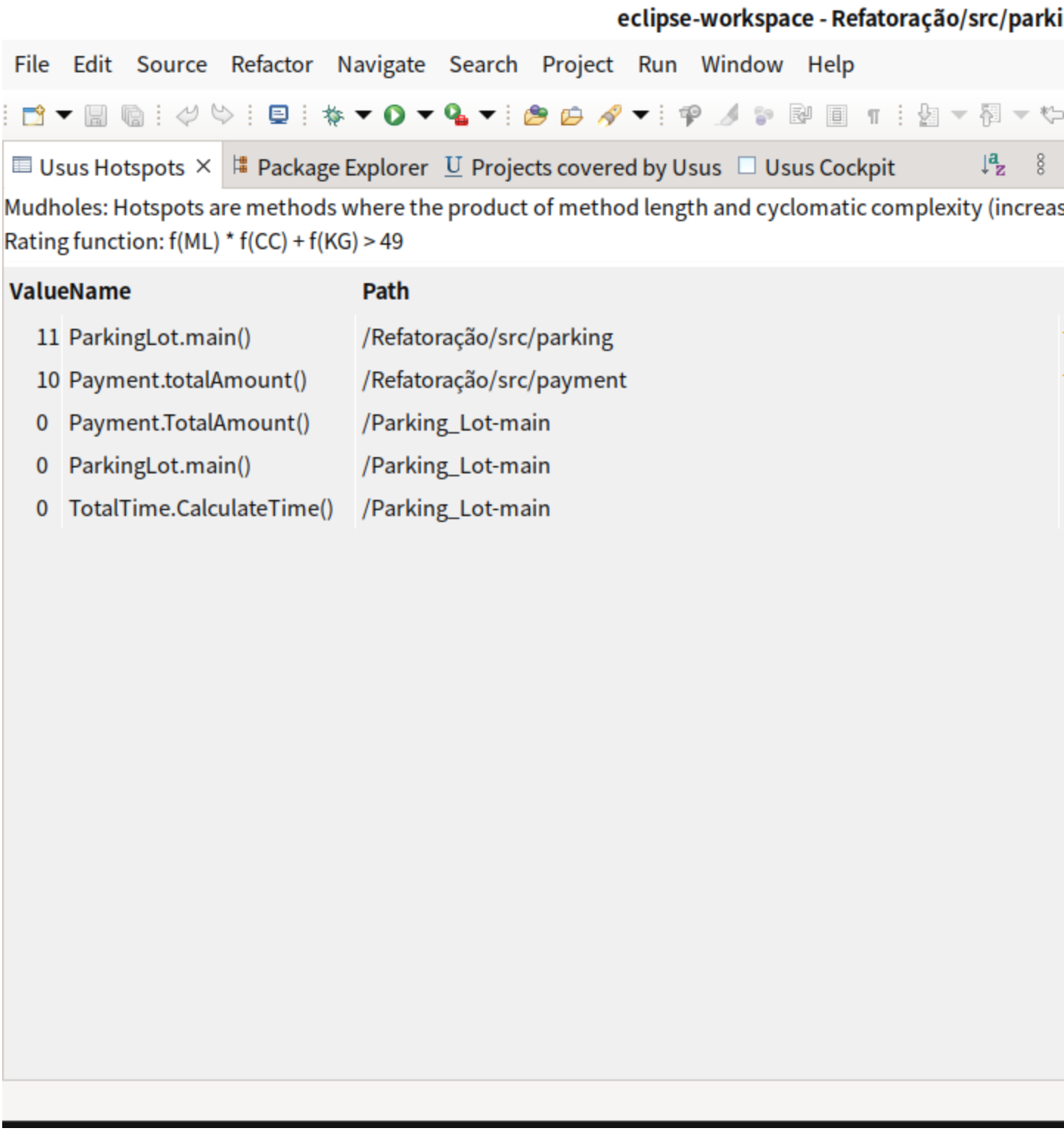
Cyclomatic Complexity:



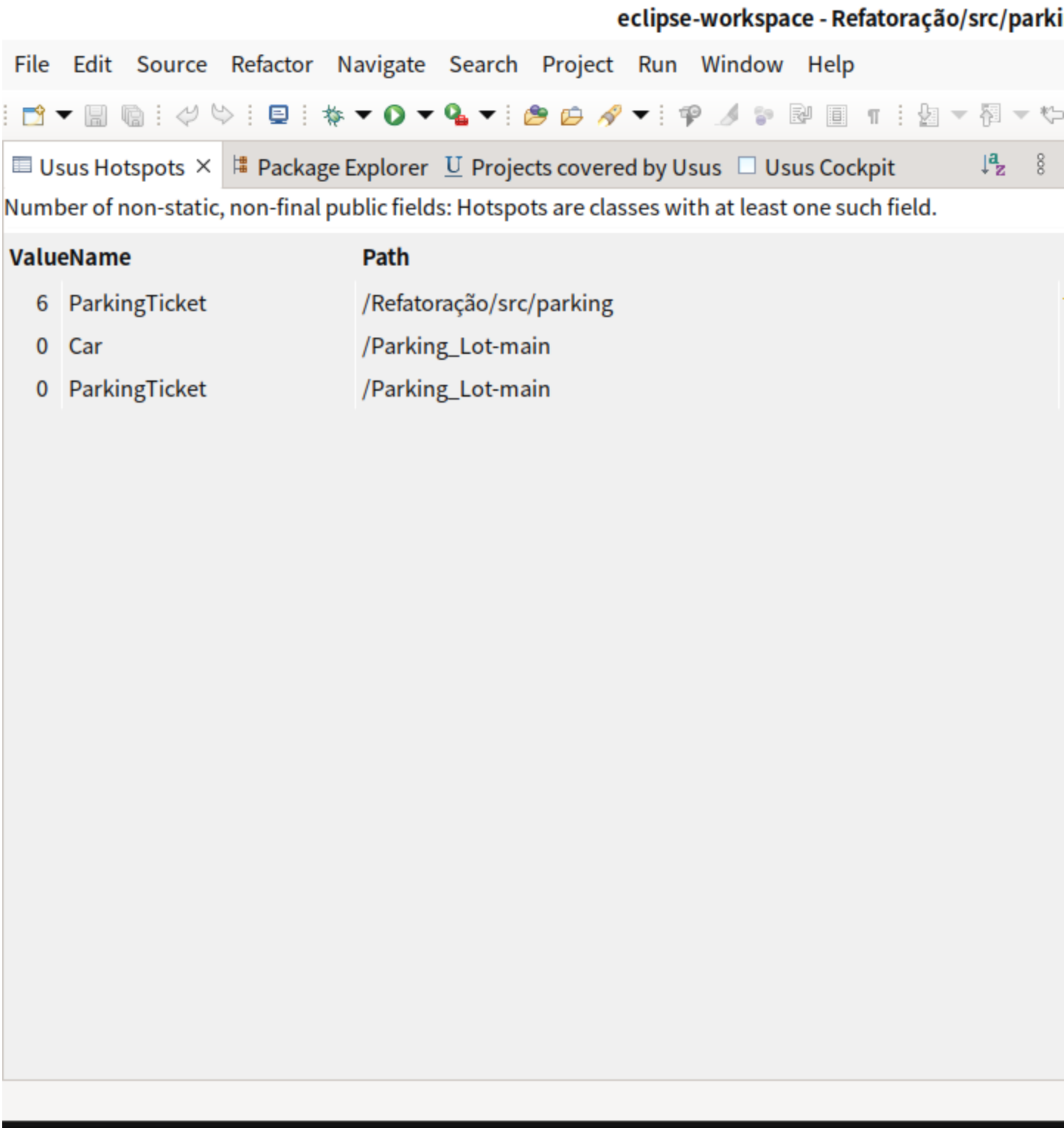
Method Length:


eclipse-workspace - Refatoração/src/parki
File Edit Source Refactor Navigate Search Project Run Window Help
Usus Hotspots
Package Explorer
Projects covered by Usus
Usus Cockpit
Method length: Hotspots are methods with more than 9 statements.
Rating function: f(value) = 1/9 value - 1
ValueName
Path
19 ParkingLot.main() /Refatoração/src/parking
16 ParkingLot.handleExit() /Refatoração/src/parking
15 ParkingLot.displayCarInforn /Refatoração/src/parking
14 ParkingLot.handleParking() /Refatoração/src/parking
12 Payment.totalAmount() /Refatoração/src/payment
12 TotalTime.calculateTime() /Refatoração/src/time
11 ParkingLot.displayExitInforr /Refatoração/src/parking
0 ParkingLot.main() /Parking_Lot-main
0 TotalTime.CalculateTime() /Parking_Lot-main

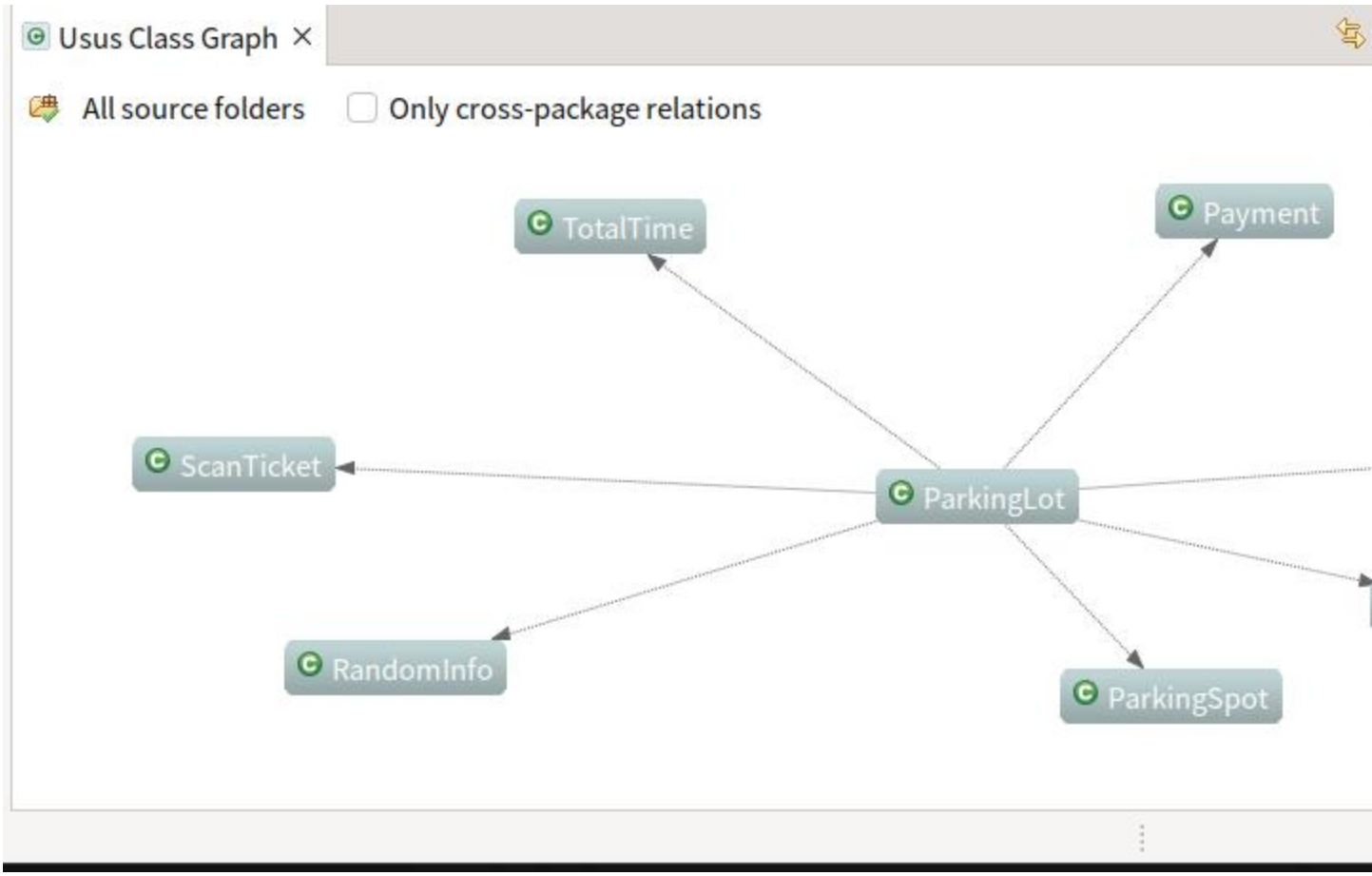
Mudholes:

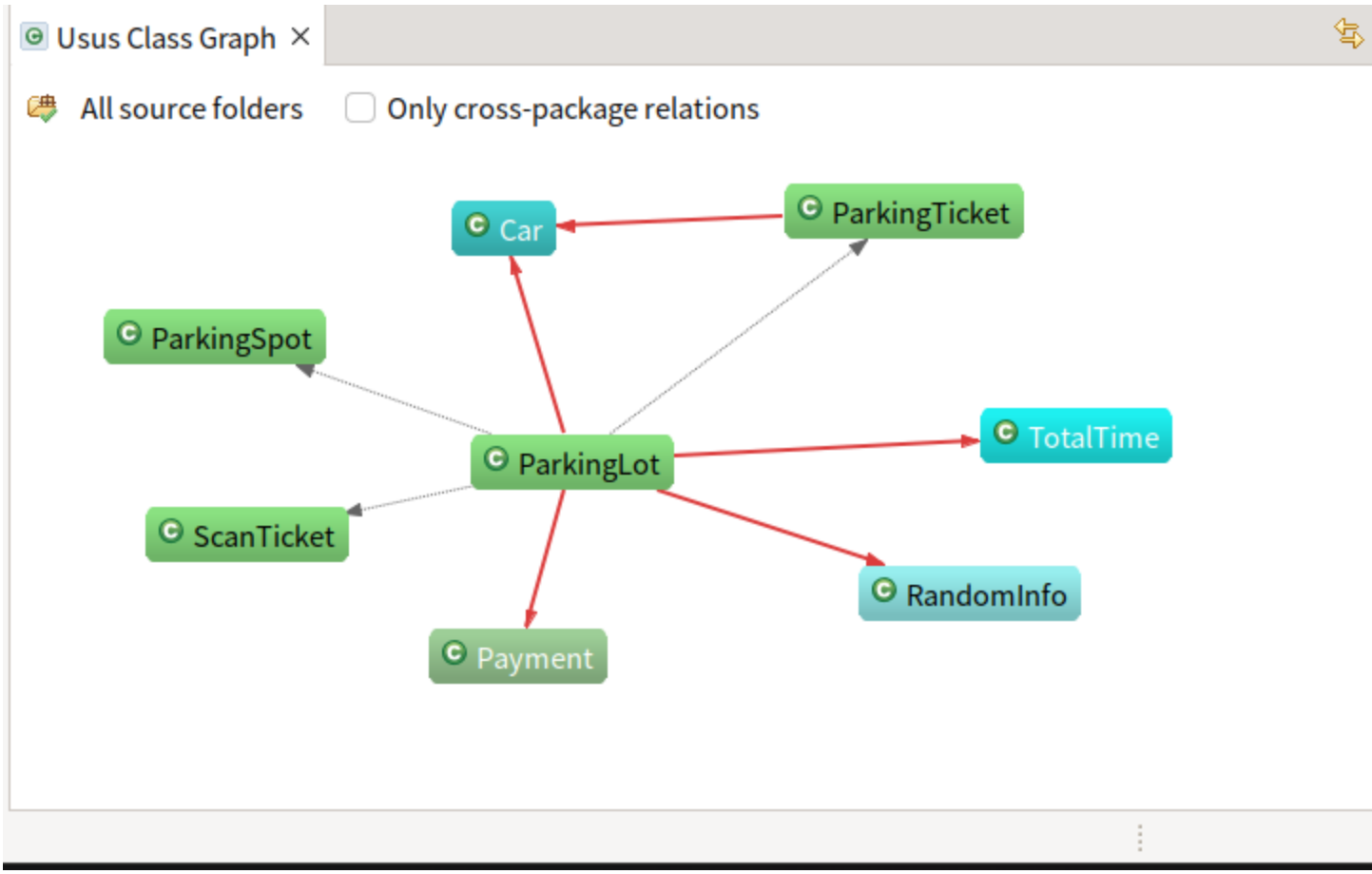


Number of non-static, non-final public fields:



Usus class graph:


Usus Class Graph
All source folders
Only cross-package relations
TotalTime
Payment
ScanTicket
ParkingLot
RandomInfo
ParkingSpot



5. Conclusão:

A refatoração teve um impacto positivo significativo no código. Embora tenha envolvido desafios, a melhoria na legibilidade, organização e qualidade geral do código justificam os esforços. O código agora adere às melhores práticas de programação, o que não apenas torna mais fácil a manutenção futura, mas também reduz a probabilidade de erros e facilita a colaboração entre os desenvolvedores.