# Appendix B

# Survey Questions

## Page 1: Introduction to Survey

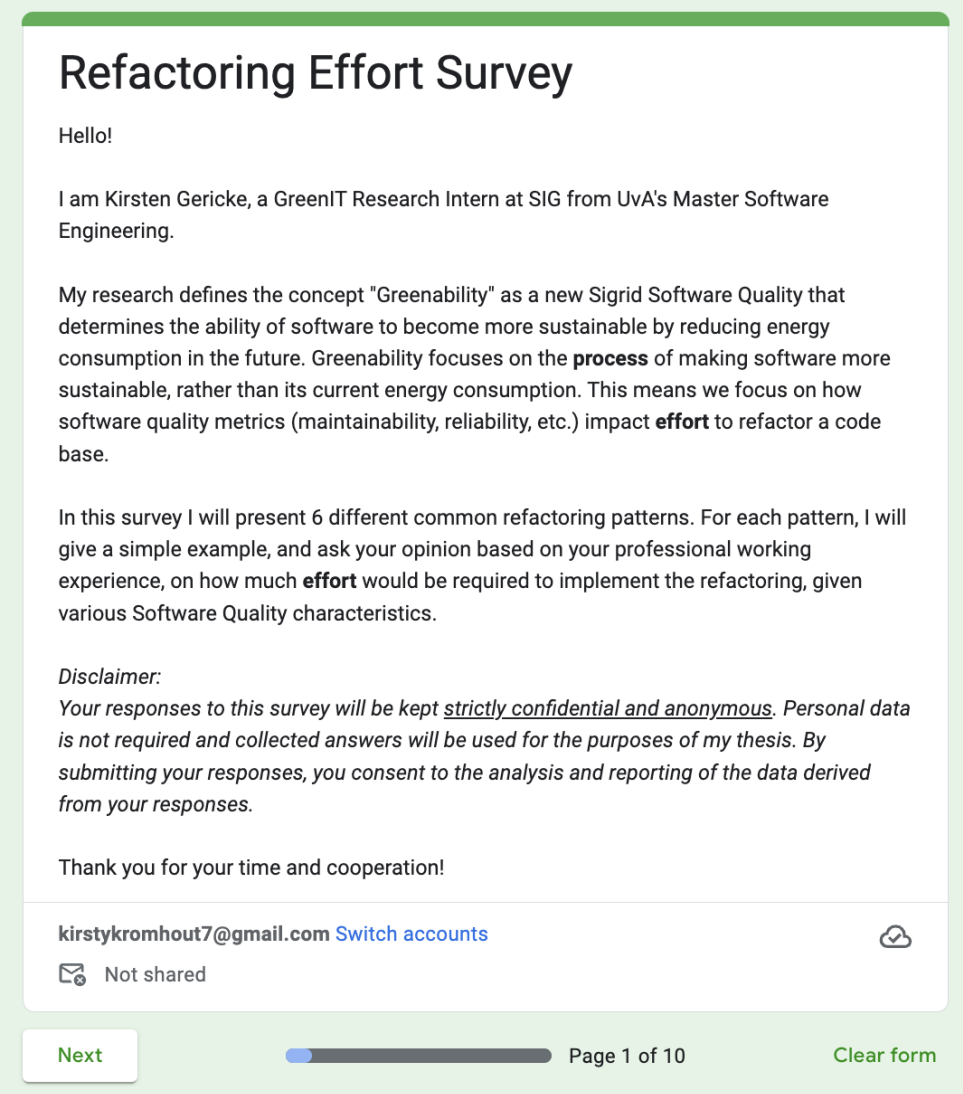


**Figure B.1: Introduction to Survey**

## Page 2: Basic Information about Participant

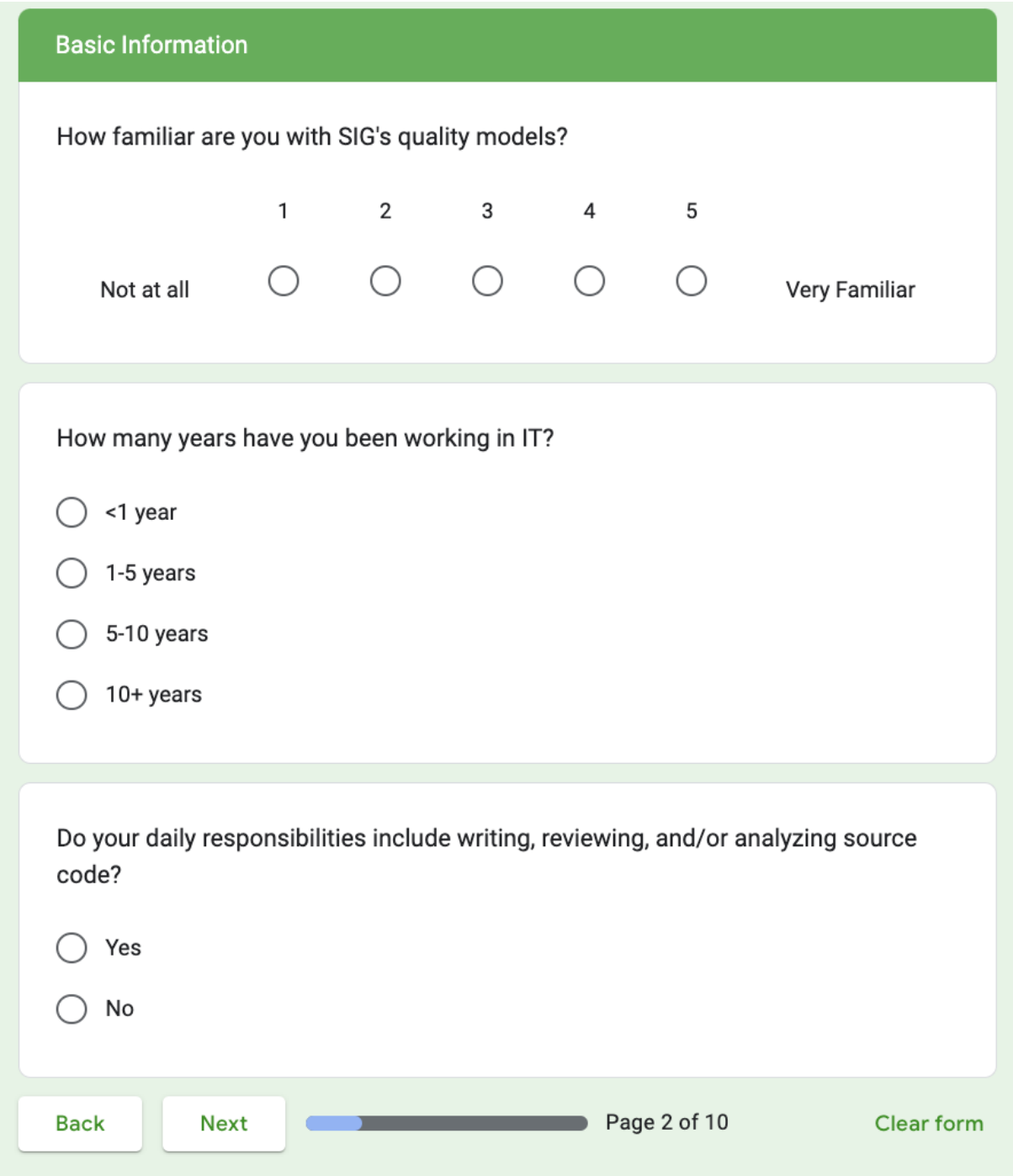


**Figure B.2: Questions about participant**

## Page 3: Task Description

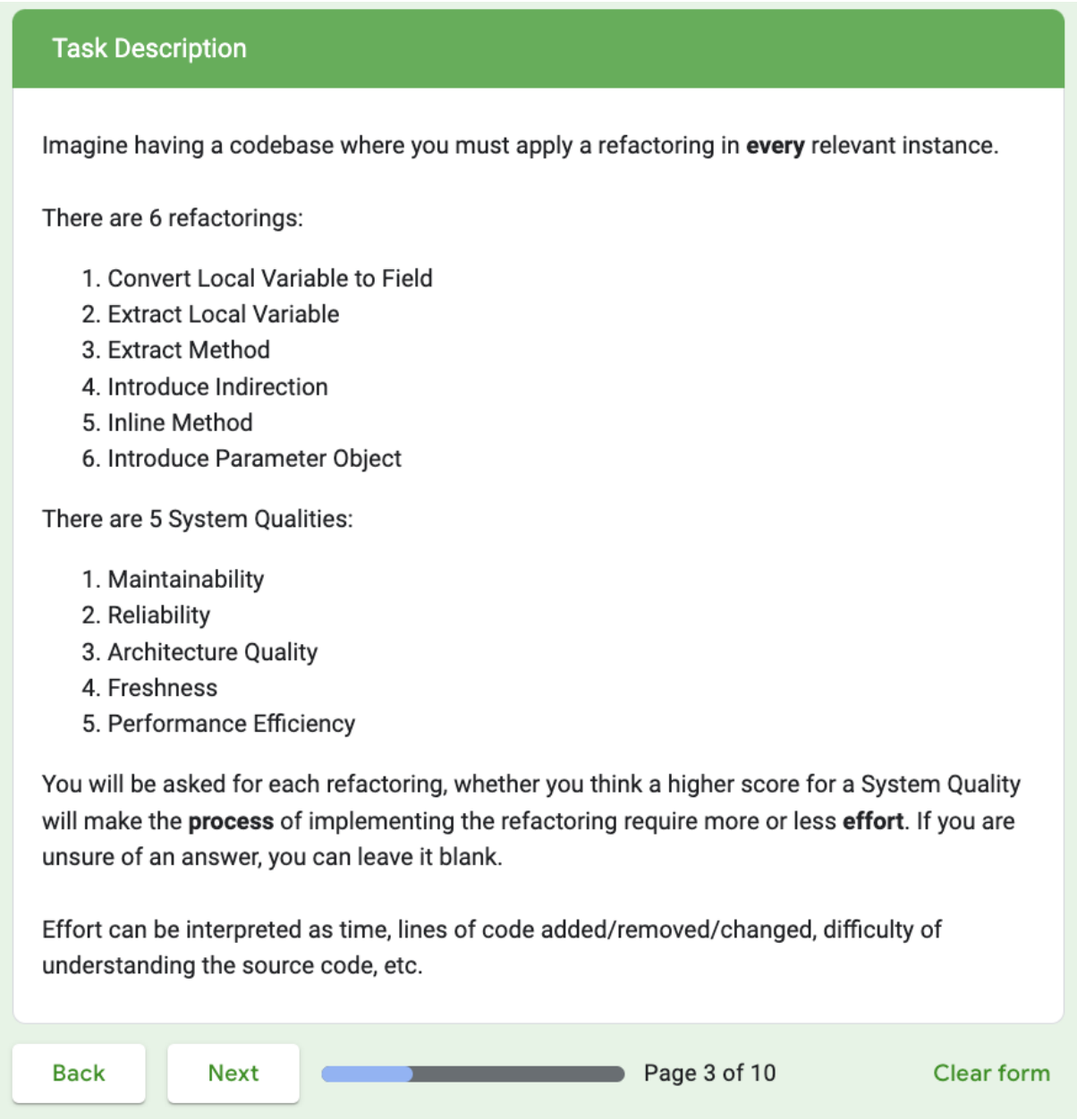


**Figure B.3: Task Description**

## Page 4: Questions for Refactoring 1

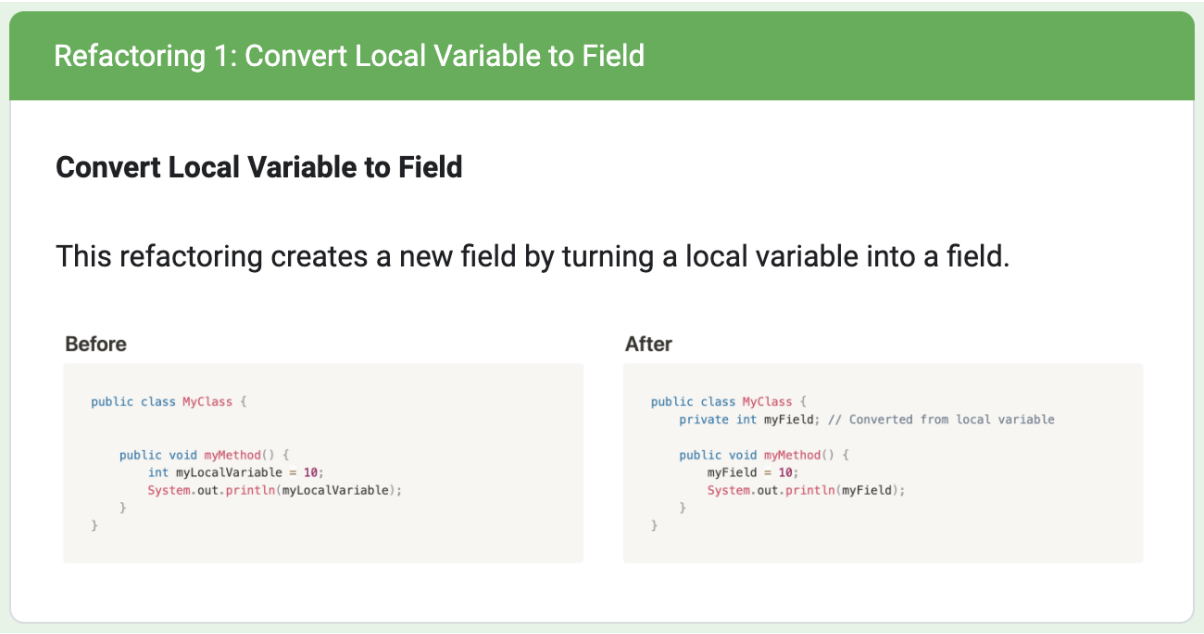


**Figure B.4: Refactoring 1: Name, Description and Example**

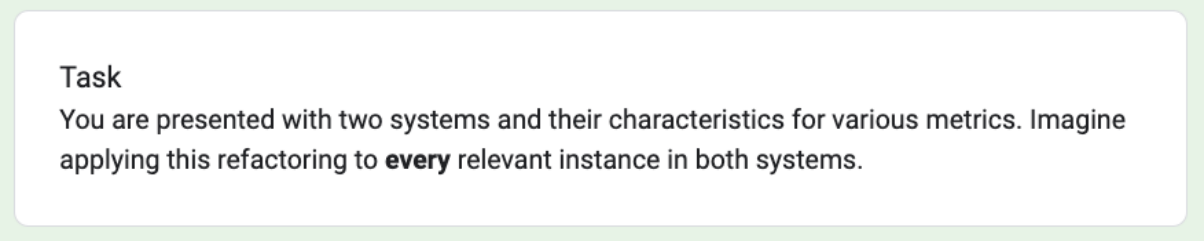


**Figure B.5: Task clarification**

**Maintainability**

Would implementing this refactoring in **System A** require more or less effort than in **System B**?

| Metrics | System A | System B |
|---|---|---|
| Volume | 10K LOC | 50K LOC |
| Duplication | No duplicated lines | 40% of the codebase is duplicated. |
| Unit Size | Average method size: 15 lines | Average method size: 50 lines |
| Unit Complexity | Average McCabe Complexity: 3 | Average McCabe Complexity: 10 |
| Module Coupling | Few parameters per method | Many parameters per method |
| Component Independence | Small interface | Many incoming calls to interface |
| Component Entanglement | Minimal component communication | High component communication |

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Less Effort | ○ | ○ | ○ | ○ | ○ | More Effort |

**Figure B.6: Maintainability Question and Example**

**Reliability**

Would implementing this refactoring in **System A** require more or less effort than in **System B**?

| Metrics | System A | System B |
|---|---|---|
| Inter-process Communication | Effective design patterns | Inefficient communication |
| Logic & Data Flow | Correct boolean logic | Contradicting if statements |
| Memory Allocation & Release | Properly allocated and freed memory | Memory leaks |
| Concurrency & Synchronization | Well-designed concurrency | Deadlocks, poor synchronization |
| Pointers | Proper pointer handling | Frequent null pointer exceptions |
| Resource Management | Scalable under load | Inefficient, poor scalability |
| Hard-coded Configuration | Flexible configurations used | Hard-coded configurations |
| Error Handling | Proper use of try-catch | Frequent unhandled exceptions |
| Arithmetic Operations | Correct precedence | Integer overflows, division by zero |
| Dead & Irrelevant Code | Clean and relevant codebase | Many dead code sections |

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Less Effort | ○ | ○ | ○ | ○ | ○ | More Effort |

**Figure B.7: Reliability Question and Example**

**Architecture Quality**

Would implementing this refactoring in **System A** require more or less effort than in **System B**?

| Metrics | System A | System B |
|---|---|---|
| Code Breakdown | Classes < 200 lines<br>Methods < 20 lines | Classes > 1000 lines<br>Methods > 100 lines |
| Component Coupling | Few component dependencies | Many component dependencies |
| Technology Prevalence | Uses Java 11, Spring Boot 3.0 | Uses Java 6, outdated frameworks |
| Component Cohesion | Follows single responsibility principle | Many responsibilities per component |
| Code Reuse | No duplication across modules | Duplicate code across modules |
| Communication Centralization | 80% internal centralized code | 30% internal code, scattered |
| Data Coupling | Components use their own databases | Components share a single database |
| Bounded Evolution | 10% co-evolution, changes localized | One change impacts entire code base |
| Knowledge Distribution | Knowledge spread across 5 people | Concentrated in 1-2 people |
| Component Freshness | Updated monthly with latest patches | Updated irregularly, every 2-3 years |

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Less Effort | ○ | ○ | ○ | ○ | ○ | More Effort |

**Figure B.8: Architecture Quality Question and Example**

**Freshness**

Would implementing this refactoring in **System A** require more or less effort than in **System B**?

| Metrics | System A | System B |
|---|---|---|
| Freshness Risk | Dependencies updated within the last 6 months | Dependencies not updated for over 2 years |
| Component Freshness | Components updated within the last month | Components not updated for over a year |
| Technology Prevalence | Uses the latest stable versions of technologies | Uses outdated versions of technologies |

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Less Effort | ○ | ○ | ○ | ○ | ○ | More Effort |

**Figure B.9: Freshness Question and Example**

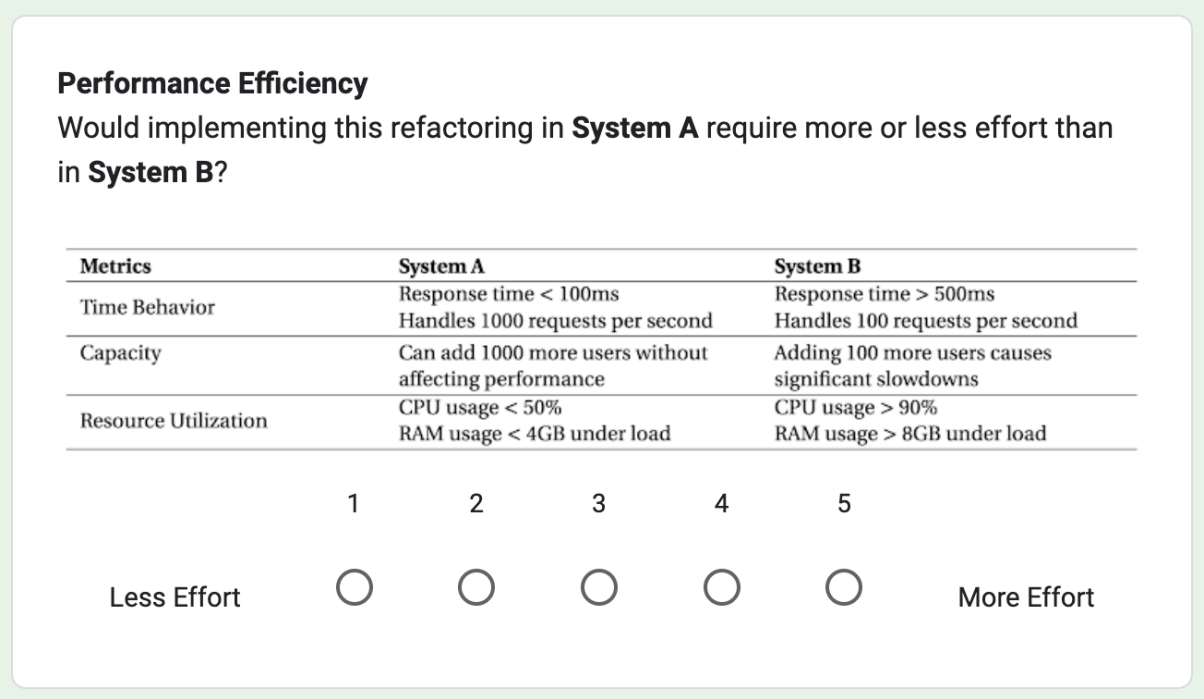


**Figure B.10: Performance Efficiency Question and Example**

## Page 5: Questions for Refactoring 2

From this point on, the property questions and examples questions were identical for all refactorings, and are not repeated to reduce unnecessary space in this report being used.

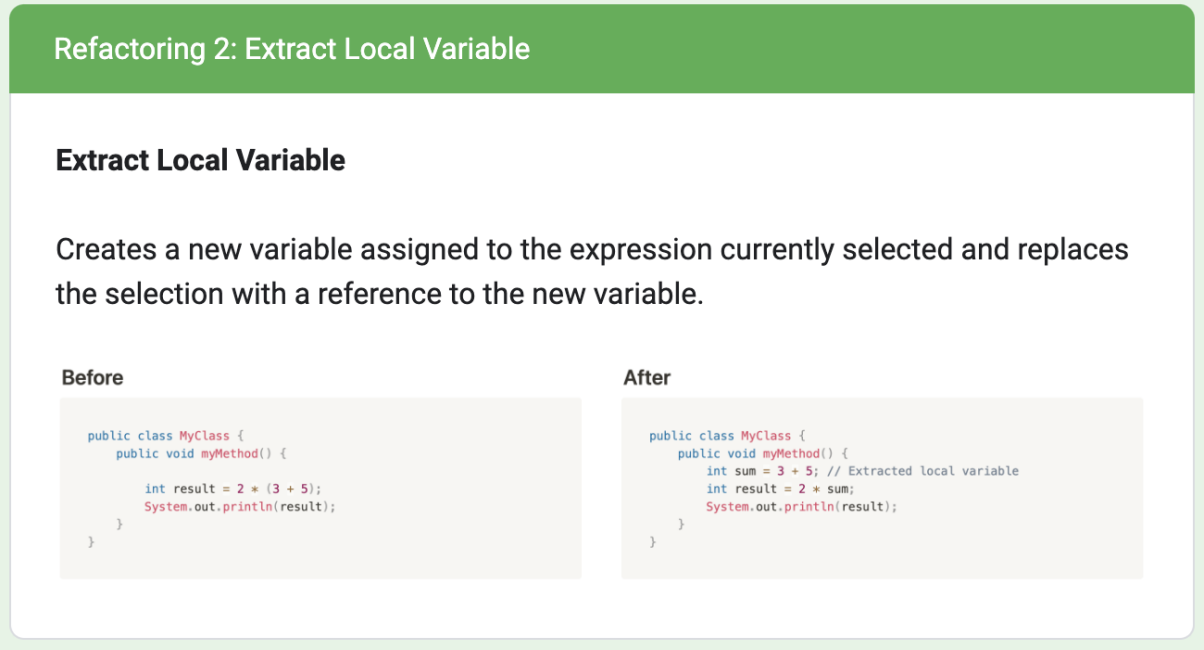


**Figure B.11: Refactoring 2: Name, Description and Example**

## Page 6: Questions for Refactoring 3

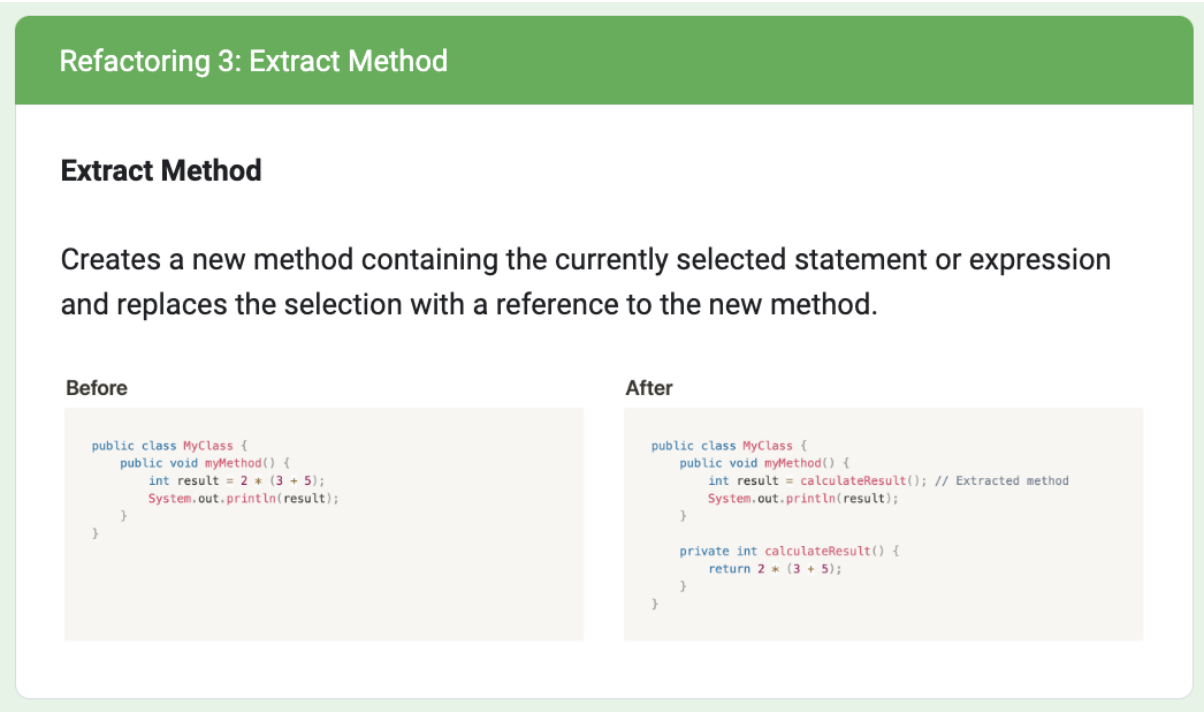


**Figure B.12: Refactoring 3: Name, Description and Example**

## Page 7: Questions for Refactoring 4

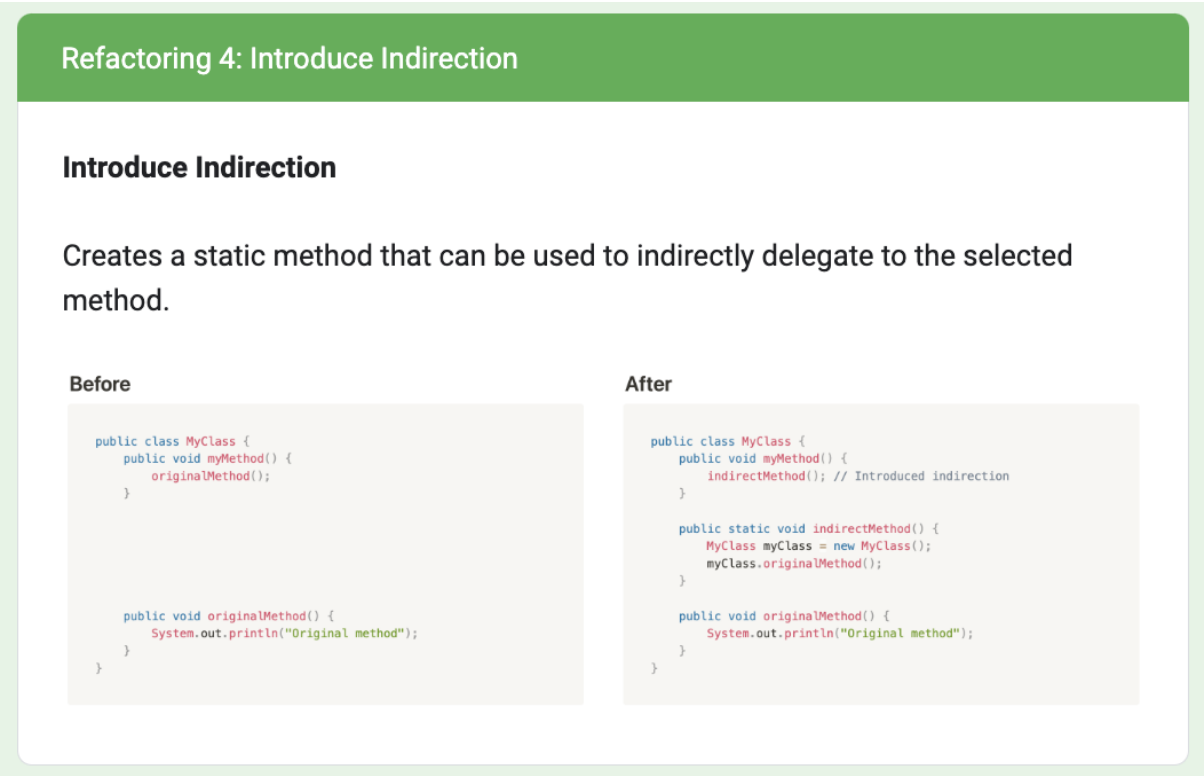


**Figure B.13: Refactoring 4: Name, Description and Example**

## Page 8: Questions for Refactoring 5


**Figure B.14: Refactoring 5: Name, Description and Example**

## Page 9: Questions for Refactoring 6

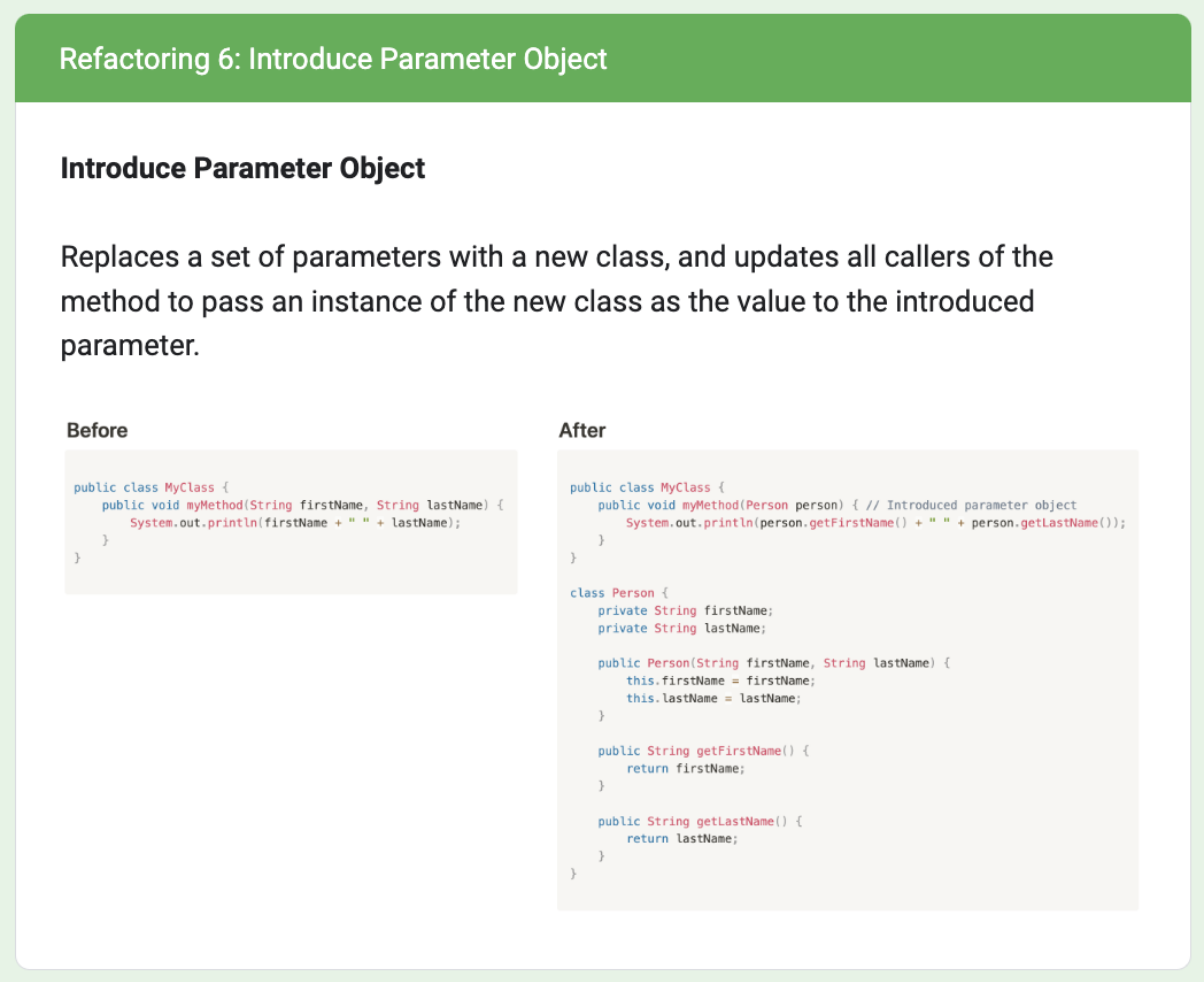


**Figure B.15: Refactoring 6: Name, Description and Example**

## Page 10: Thank you and Additional Questions

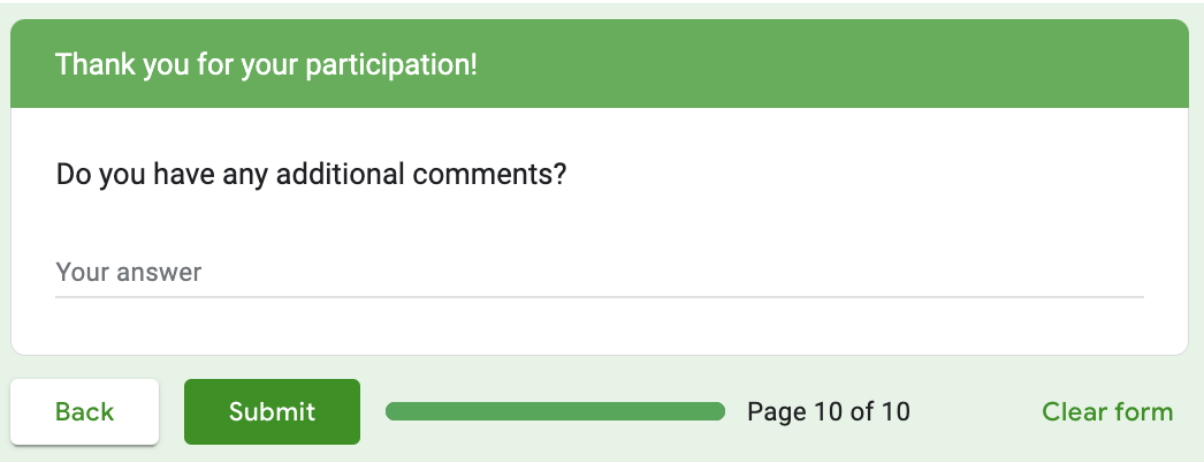


**Figure B.16: Final Thanks and Additional Questions**