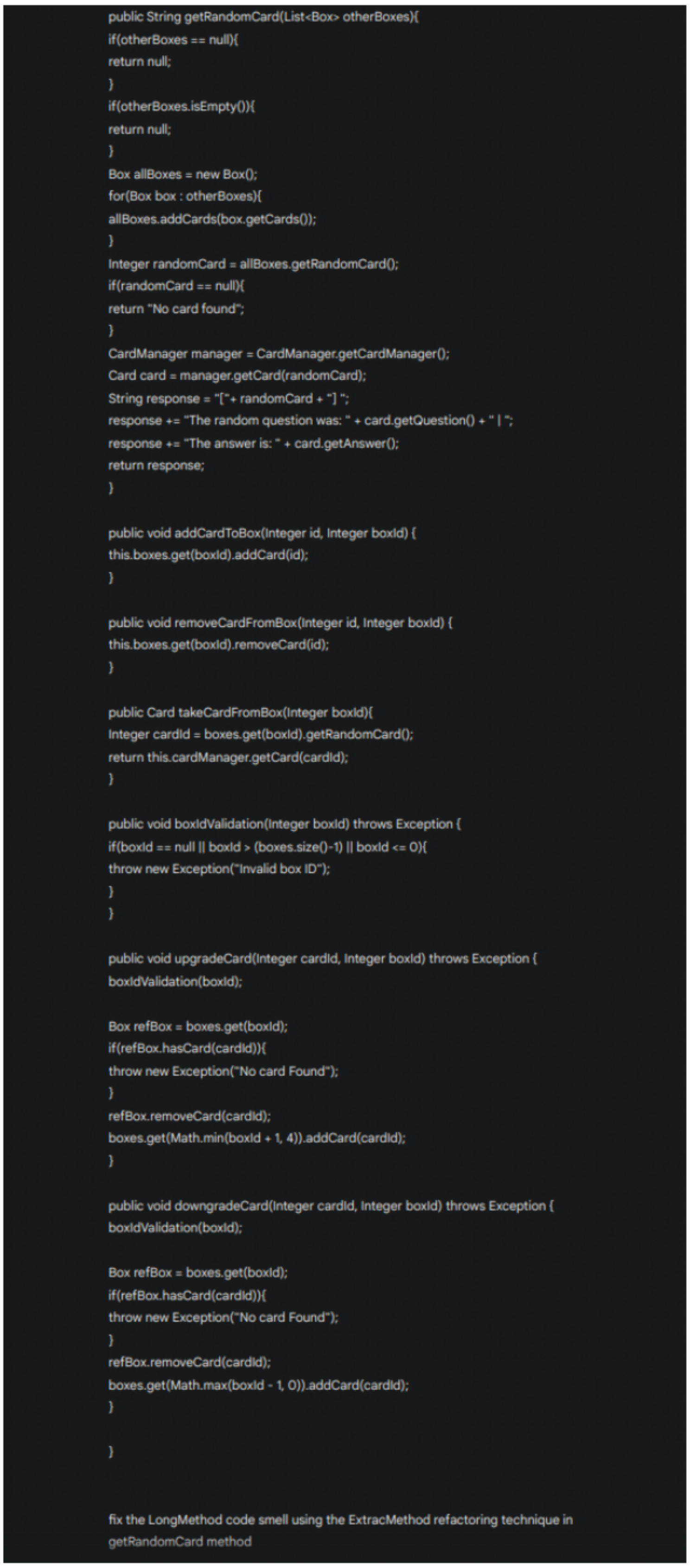
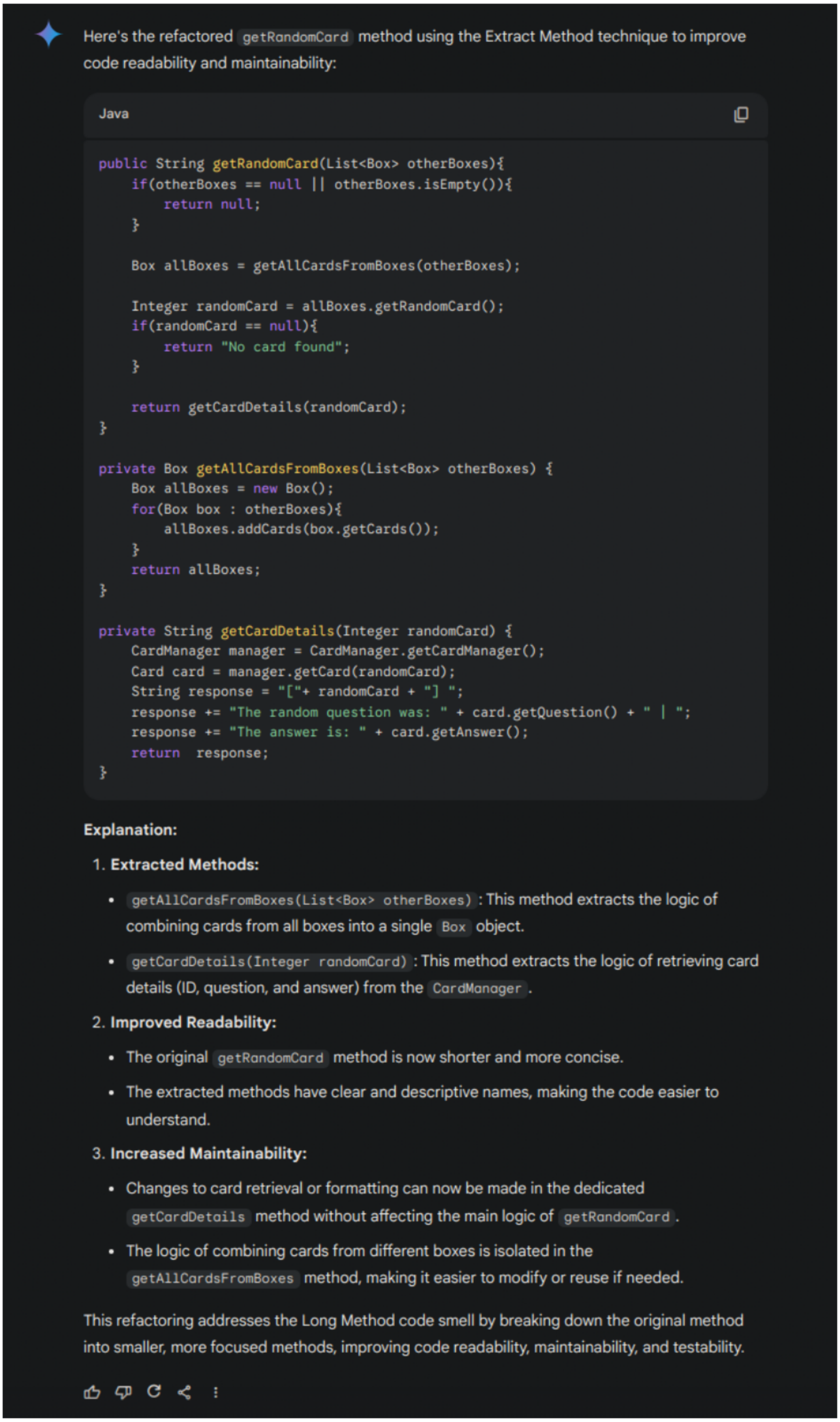
LeitnerSystem



```
public String getRandomCard(List<Box> otherBoxes){
if(otherBoxes == null){
return null;
}
if(otherBoxes.isEmpty()){
return null;
}
Box allBoxes = new Box();
for(Box box : otherBoxes){
allBoxes.addCards(box.getCards());
}
Integer randomCard = allBoxes.getRandomCard();
if(randomCard == null){
return "No card found";
}
CardManager manager = CardManager.getCardManager();
Card card = manager.getCard(randomCard);
String response = "["+ randomCard + "] ";
response += "The random question was: " + card.getQuestion() + " | ";
response += "The answer is: " + card.getAnswer();
return response;
}

public void addCardToBox(Integer id, Integer boxId) {
this.boxes.get(boxId).addCard(id);
}

public void removeCardFromBox(Integer id, Integer boxId) {
this.boxes.get(boxId).removeCard(id);
}

public Card takeCardFromBox(Integer boxId){
Integer cardId = boxes.get(boxId).getRandomCard();
return this.cardManager.getCard(cardId);
}

public void boxIdValidation(Integer boxId) throws Exception {
if(boxId == null || boxId > (boxes.size()-1) || boxId <= 0){
throw new Exception("Invalid box ID");
}
}

public void upgradeCard(Integer cardId, Integer boxId) throws Exception {
boxIdValidation(boxId);

Box refBox = boxes.get(boxId);
if(refBox.hasCard(cardId)){
throw new Exception("No card Found");
}
refBox.removeCard(cardId);
boxes.get(Math.min(boxId + 1, 4)).addCard(cardId);
}

public void downgradeCard(Integer cardId, Integer boxId) throws Exception {
boxIdValidation(boxId);

Box refBox = boxes.get(boxId);
if(refBox.hasCard(cardId)){
throw new Exception("No card Found");
}
refBox.removeCard(cardId);
boxes.get(Math.max(boxId - 1, 0)).addCard(cardId);
}

}
```

fix the LongMethod code smell using the ExtracMethod refactoring technique in getRandomCard method


Here's the refactored `getRandomCard` method using the Extract Method technique to improve code readability and maintainability:

Java

```java
public String getRandomCard(List<Box> otherBoxes){
    if(otherBoxes == null || otherBoxes.isEmpty()){
        return null;
    }

    Box allBoxes = getAllCardsFromBoxes(otherBoxes);

    Integer randomCard = allBoxes.getRandomCard();
    if(randomCard == null){
        return "No card found";
    }

    return getCardDetails(randomCard);
}

private Box getAllCardsFromBoxes(List<Box> otherBoxes) {
    Box allBoxes = new Box();
    for(Box box : otherBoxes){
        allBoxes.addCards(box.getCards());
    }
    return allBoxes;
}

private String getCardDetails(Integer randomCard) {
    CardManager manager = CardManager.getCardManager();
    Card card = manager.getCard(randomCard);
    String response = "["+ randomCard + "] ";
    response += "The random question was: " + card.getQuestion() + " | ";
    response += "The answer is: " + card.getAnswer();
    return  response;
}
```

Explanation:

1. **Extracted Methods:**
   - `getAllCardsFromBoxes(List<Box> otherBoxes)`: This method extracts the logic of combining cards from all boxes into a single `Box` object.
   - `getCardDetails(Integer randomCard)`: This method extracts the logic of retrieving card details (ID, question, and answer) from the `CardManager`.
2. **Improved Readability:**
   - The original `getRandomCard` method is now shorter and more concise.
   - The extracted methods have clear and descriptive names, making the code easier to understand.
3. **Increased Maintainability:**
   - Changes to card retrieval or formatting can now be made in the dedicated `getCardDetails` method without affecting the main logic of `getRandomCard`.
   - The logic of combining cards from different boxes is isolated in the `getAllCardsFromBoxes` method, making it easier to modify or reuse if needed.

This refactoring addresses the Long Method code smell by breaking down the original method into smaller, more focused methods, improving code readability, maintainability, and testability.