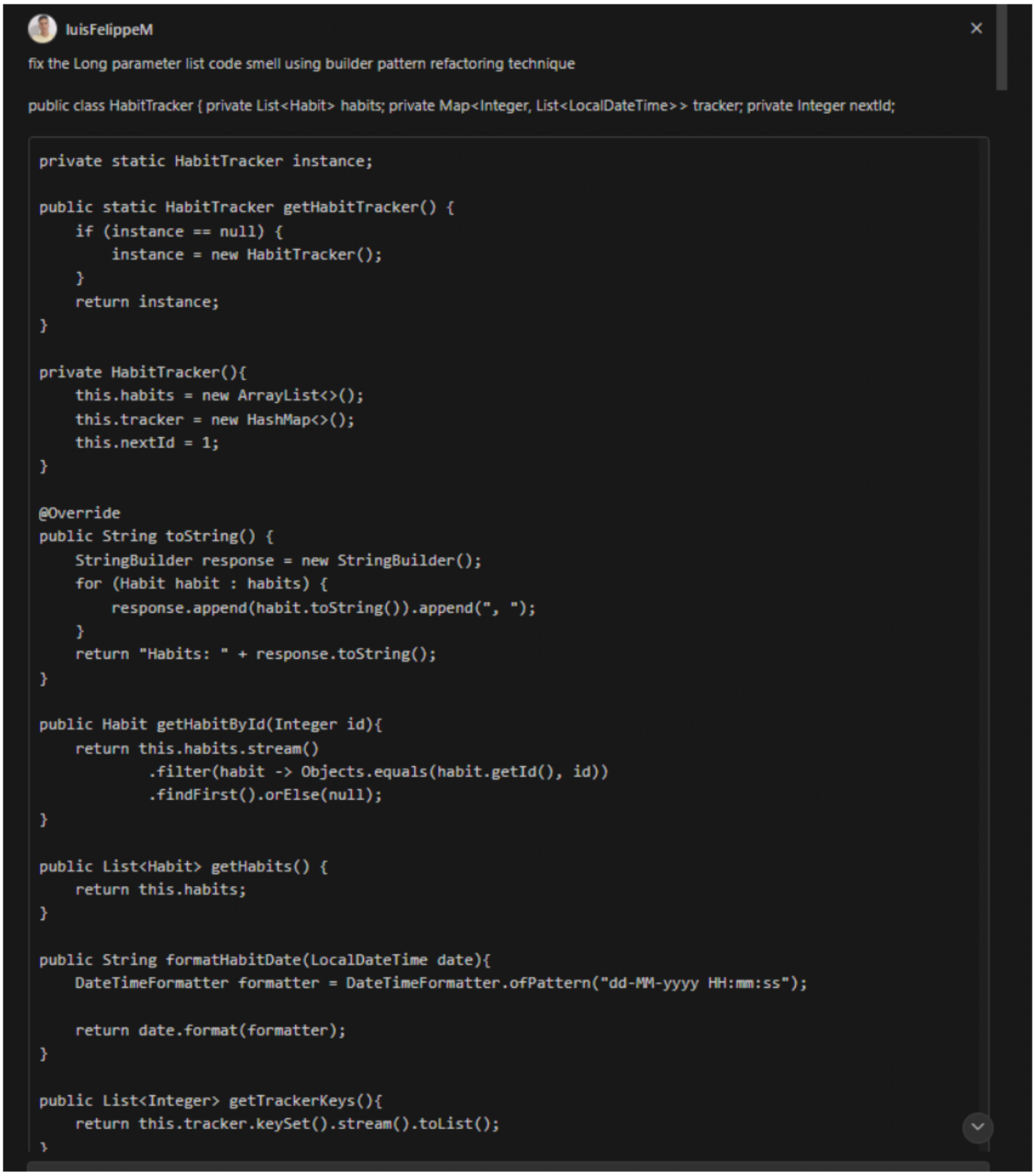
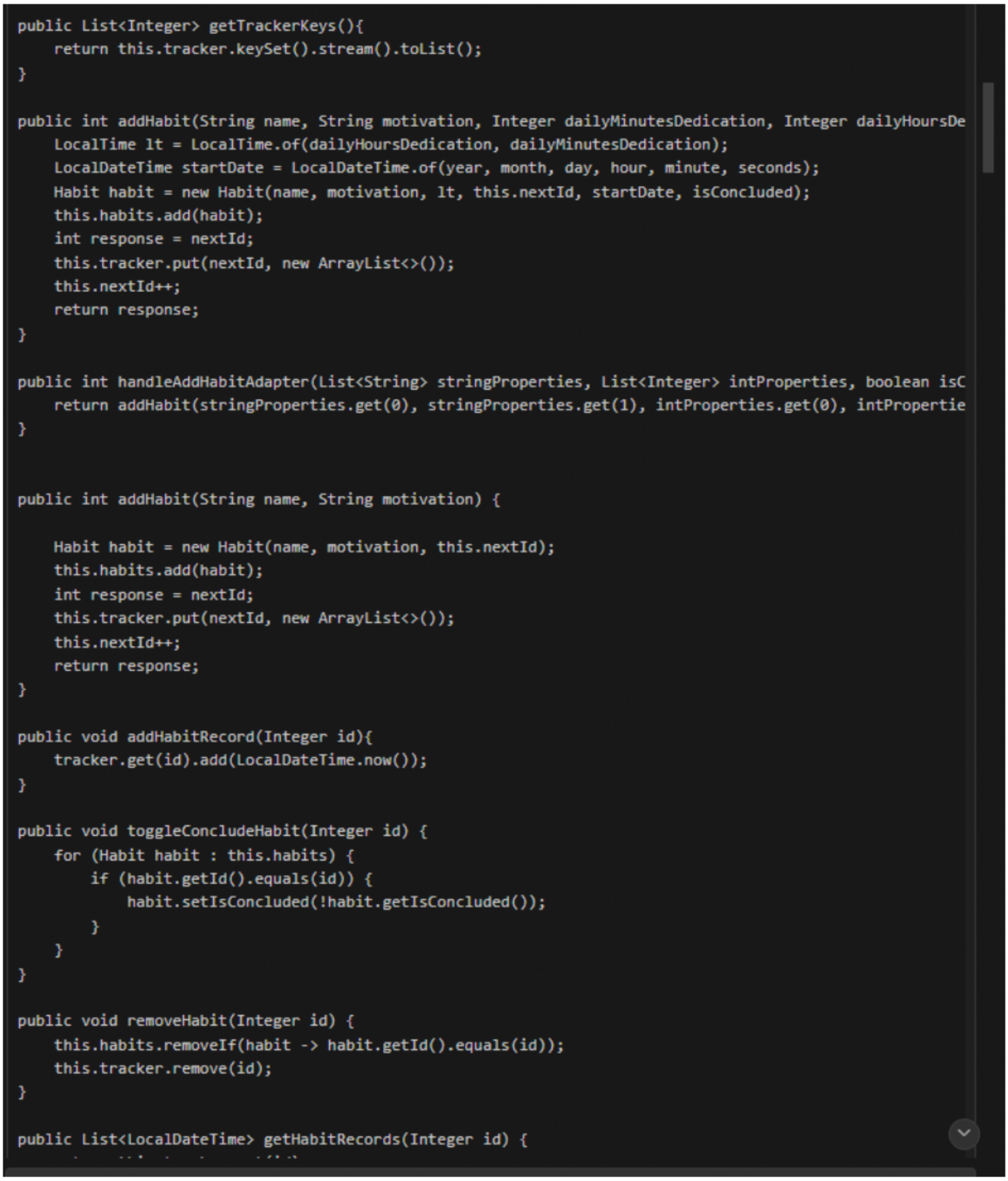
- 

luisFelippeM

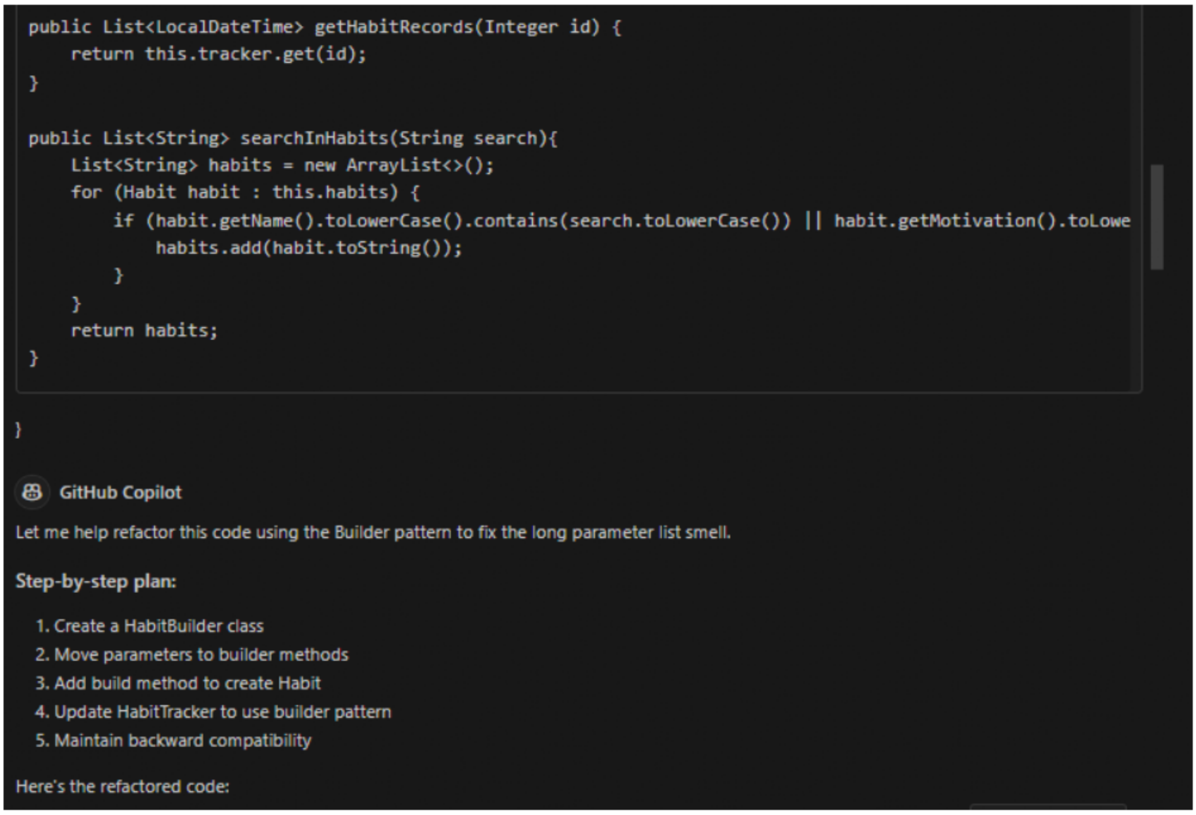
fix the Long parameter list code smell using builder pattern refactoring technique

public class HabitTracker { private List<Habit> habits; private Map<Integer, List<LocalDateTime>> tracker; private Integer nextId;

```
private static HabitTracker instance;

public static HabitTracker getHabitTracker() {
    if (instance == null) {
        instance = new HabitTracker();
    }
    return instance;
}

private HabitTracker(){
    this.habits = new ArrayList<>();
    this.tracker = new HashMap<>();
    this.nextId = 1;
}

@Override
public String toString() {
    StringBuilder response = new StringBuilder();
    for (Habit habit : habits) {
        response.append(habit.toString()).append(", ");
    }
    return "Habits: " + response.toString();
}

public Habit getHabitById(Integer id){
    return this.habits.stream()
            .filter(habit -> Objects.equals(habit.getId(), id))
            .findFirst().orElse(null);
}

public List<Habit> getHabits() {
    return this.habits;
}

public String formatHabitDate(LocalDateTime date){
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    return date.format(formatter);
}

public List<Integer> getTrackerKeys(){
    return this.tracker.keySet().stream().toList();
}
```



```java
public List<Integer> getTrackerKeys(){
    return this.tracker.keySet().stream().toList();
}

public int addHabit(String name, String motivation, Integer dailyMinutesDedication, Integer dailyHoursDe
    LocalTime lt = LocalTime.of(dailyHoursDedication, dailyMinutesDedication);
    LocalDateTime startDate = LocalDateTime.of(year, month, day, hour, minute, seconds);
    Habit habit = new Habit(name, motivation, lt, this.nextId, startDate, isConcluded);
    this.habits.add(habit);
    int response = nextId;
    this.tracker.put(nextId, new ArrayList<>());
    this.nextId++;
    return response;
}

public int handleAddHabitAdapter(List<String> stringProperties, List<Integer> intProperties, boolean isC
    return addHabit(stringProperties.get(0), stringProperties.get(1), intProperties.get(0), intPropertie
}


public int addHabit(String name, String motivation) {

    Habit habit = new Habit(name, motivation, this.nextId);
    this.habits.add(habit);
    int response = nextId;
    this.tracker.put(nextId, new ArrayList<>());
    this.nextId++;
    return response;
}

public void addHabitRecord(Integer id){
    tracker.get(id).add(LocalDateTime.now());
}

public void toggleConcludeHabit(Integer id) {
    for (Habit habit : this.habits) {
        if (habit.getId().equals(id)) {
            habit.setIsConcluded(!habit.getIsConcluded());
        }
    }
}

public void removeHabit(Integer id) {
    this.habits.removeIf(habit -> habit.getId().equals(id));
    this.tracker.remove(id);
}

public List<LocalDateTime> getHabitRecords(Integer id) {
```



```
public List<LocalDateTime> getHabitRecords(Integer id) {
    return this.tracker.get(id);
}

public List<String> searchInHabits(String search){
    List<String> habits = new ArrayList<>();
    for (Habit habit : this.habits) {
        if (habit.getName().toLowerCase().contains(search.toLowerCase()) || habit.getMotivation().toLowe
            habits.add(habit.toString());
        }
    }
    return habits;
}

}
```

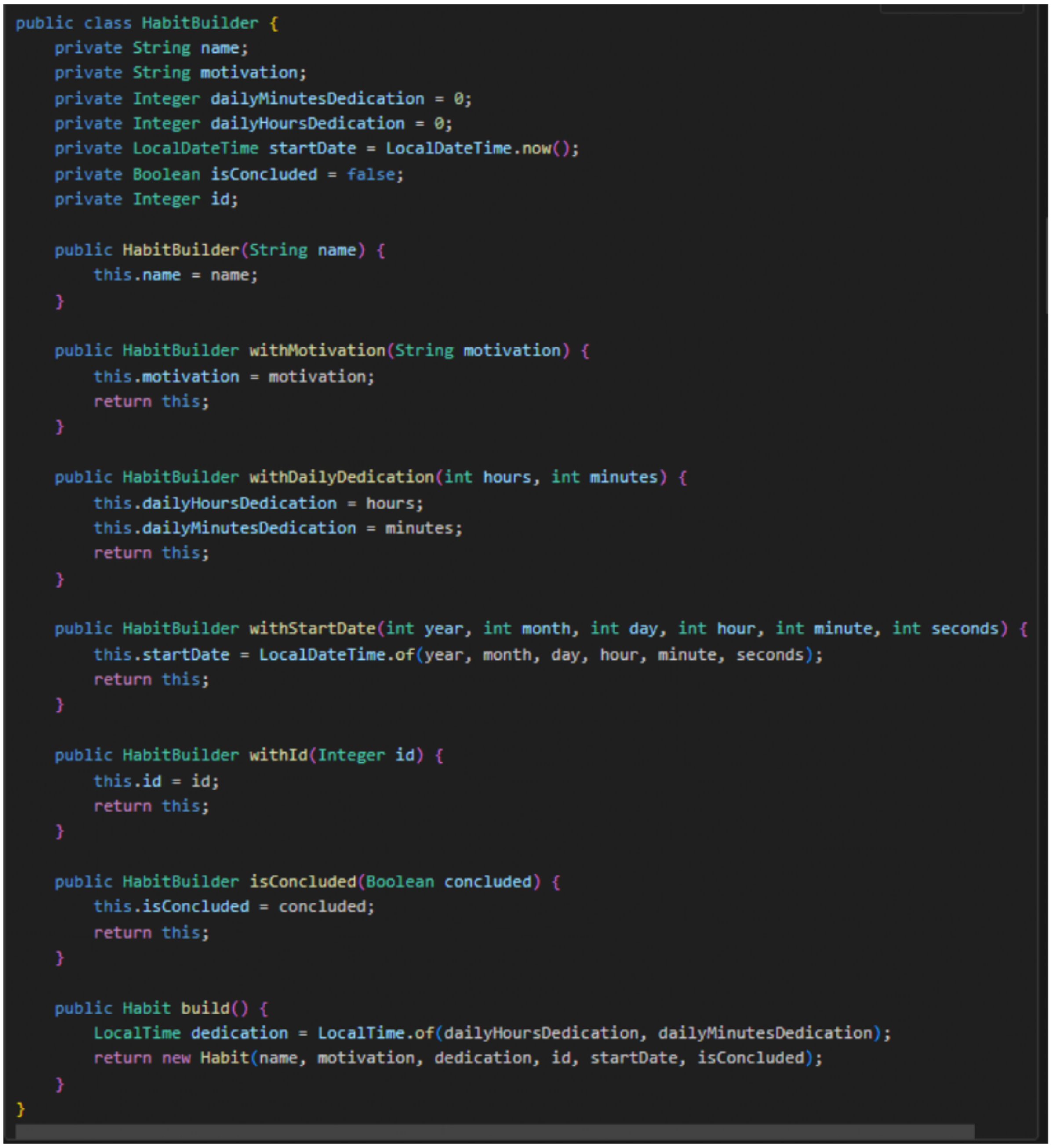
GitHub Copilot

Let me help refactor this code using the Builder pattern to fix the long parameter list smell.

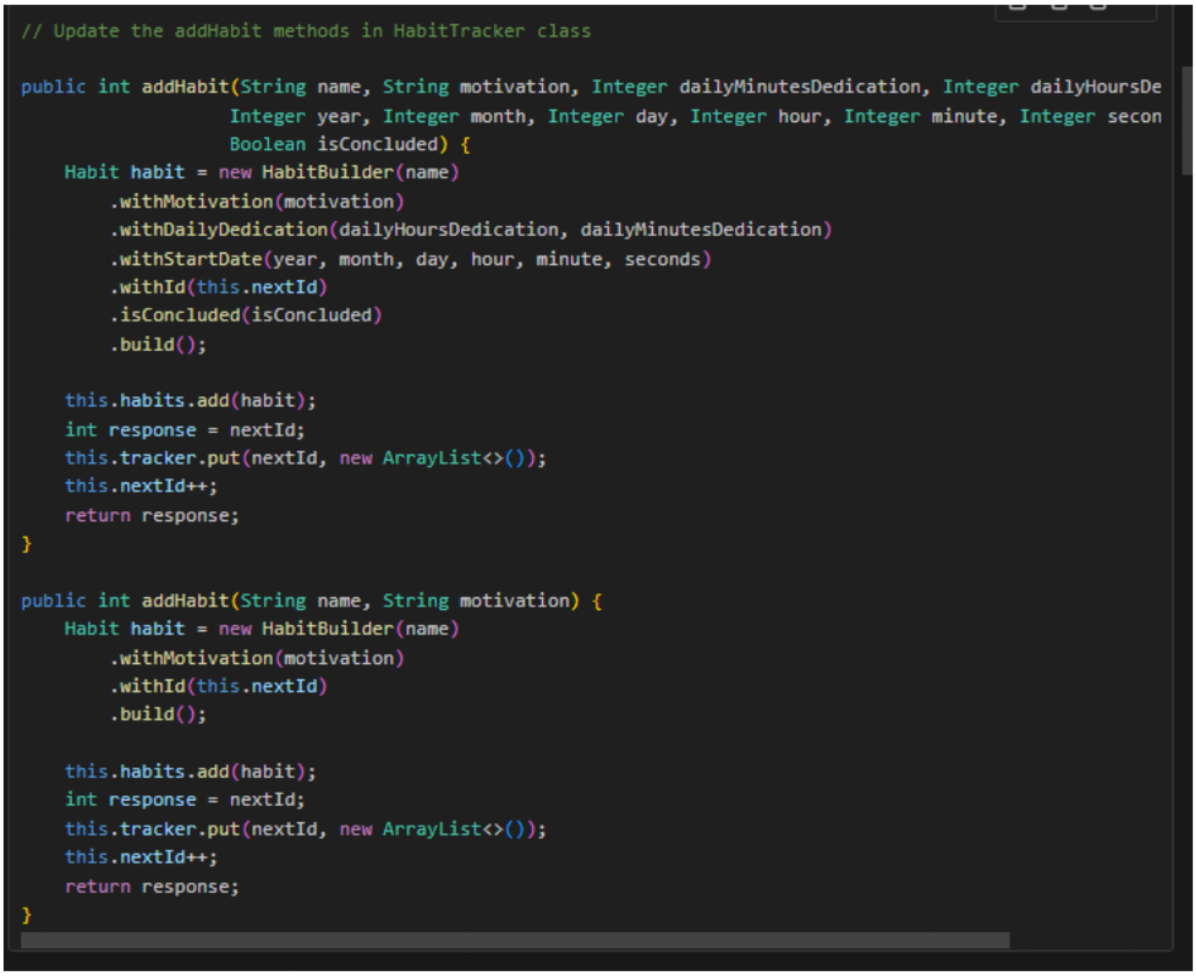
Step-by-step plan:

1. Create a HabitBuilder class
2. Move parameters to builder methods
3. Add build method to create Habit
4. Update HabitTracker to use builder pattern
5. Maintain backward compatibility

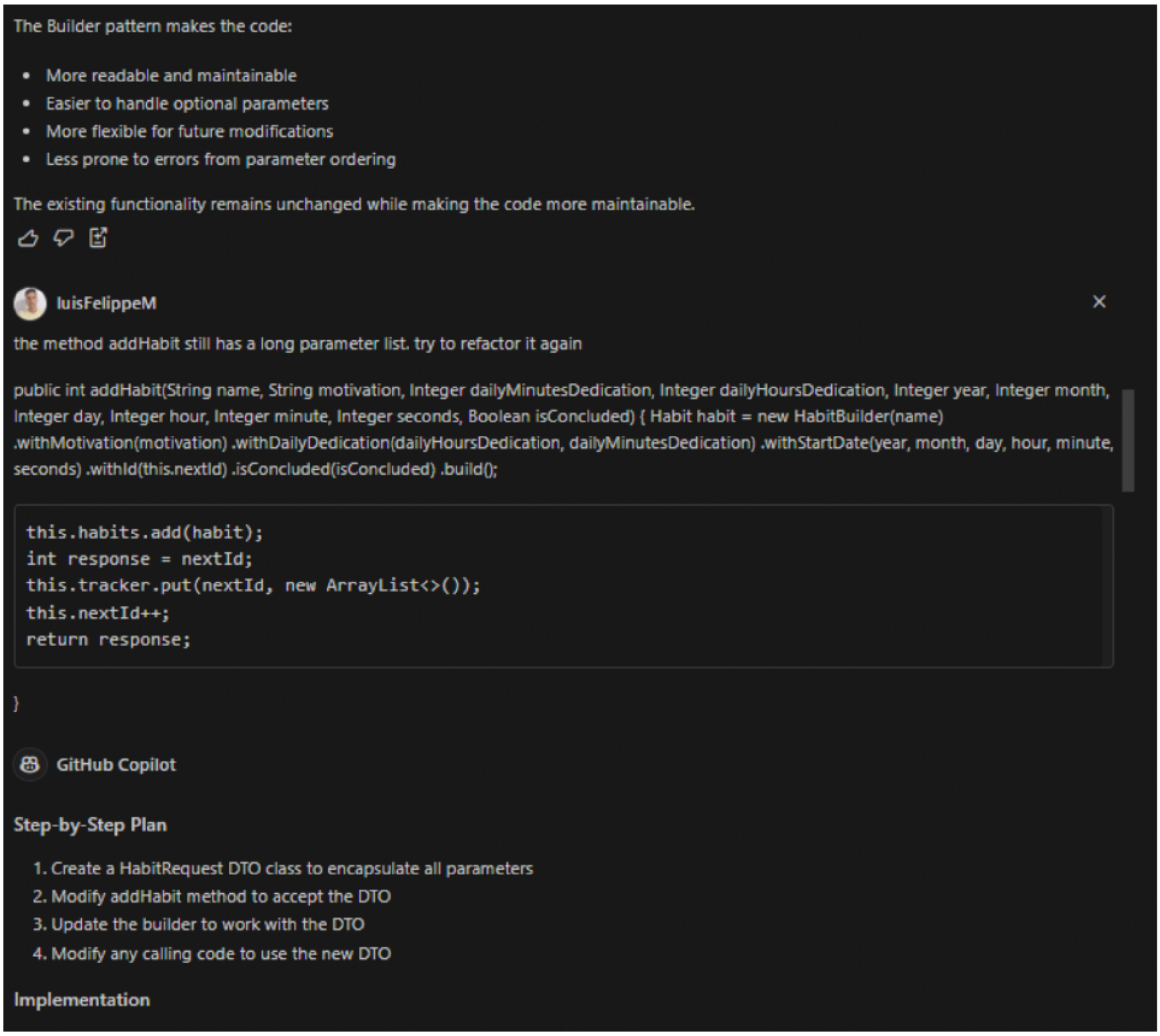
Here's the refactored code:



```java
public class HabitBuilder {
    private String name;
    private String motivation;
    private Integer dailyMinutesDedication = 0;
    private Integer dailyHoursDedication = 0;
    private LocalDateTime startDate = LocalDateTime.now();
    private Boolean isConcluded = false;
    private Integer id;

    public HabitBuilder(String name) {
        this.name = name;
    }

    public HabitBuilder withMotivation(String motivation) {
        this.motivation = motivation;
        return this;
    }

    public HabitBuilder withDailyDedication(int hours, int minutes) {
        this.dailyHoursDedication = hours;
        this.dailyMinutesDedication = minutes;
        return this;
    }

    public HabitBuilder withStartDate(int year, int month, int day, int hour, int minute, int seconds) {
        this.startDate = LocalDateTime.of(year, month, day, hour, minute, seconds);
        return this;
    }

    public HabitBuilder withId(Integer id) {
        this.id = id;
        return this;
    }

    public HabitBuilder isConcluded(Boolean concluded) {
        this.isConcluded = concluded;
        return this;
    }

    public Habit build() {
        LocalTime dedication = LocalTime.of(dailyHoursDedication, dailyMinutesDedication);
        return new Habit(name, motivation, dedication, id, startDate, isConcluded);
    }
}
```



```
// Update the addHabit methods in HabitTracker class

public int addHabit(String name, String motivation, Integer dailyMinutesDedication, Integer dailyHoursDe
                    Integer year, Integer month, Integer day, Integer hour, Integer minute, Integer secon
                    Boolean isConcluded) {
    Habit habit = new HabitBuilder(name)
        .withMotivation(motivation)
        .withDailyDedication(dailyHoursDedication, dailyMinutesDedication)
        .withStartDate(year, month, day, hour, minute, seconds)
        .withId(this.nextId)
        .isConcluded(isConcluded)
        .build();

    this.habits.add(habit);
    int response = nextId;
    this.tracker.put(nextId, new ArrayList<>());
    this.nextId++;
    return response;
}

public int addHabit(String name, String motivation) {
    Habit habit = new HabitBuilder(name)
        .withMotivation(motivation)
        .withId(this.nextId)
        .build();

    this.habits.add(habit);
    int response = nextId;
    this.tracker.put(nextId, new ArrayList<>());
    this.nextId++;
    return response;
}
```



The Builder pattern makes the code:

- More readable and maintainable
- Easier to handle optional parameters
- More flexible for future modifications
- Less prone to errors from parameter ordering

The existing functionality remains unchanged while making the code more maintainable.

luisFelippeM

the method addHabit still has a long parameter list. try to refactor it again

public int addHabit(String name, String motivation, Integer dailyMinutesDedication, Integer dailyHoursDedication, Integer year, Integer month, Integer day, Integer hour, Integer minute, Integer seconds, Boolean isConcluded) { Habit habit = new HabitBuilder(name) .withMotivation(motivation) .withDailyDedication(dailyHoursDedication, dailyMinutesDedication) .withStartDate(year, month, day, hour, minute, seconds) .withId(this.nextId) .isConcluded(isConcluded) .build();

```
this.habits.add(habit);
int response = nextId;
this.tracker.put(nextId, new ArrayList<>());
this.nextId++;
return response;
```

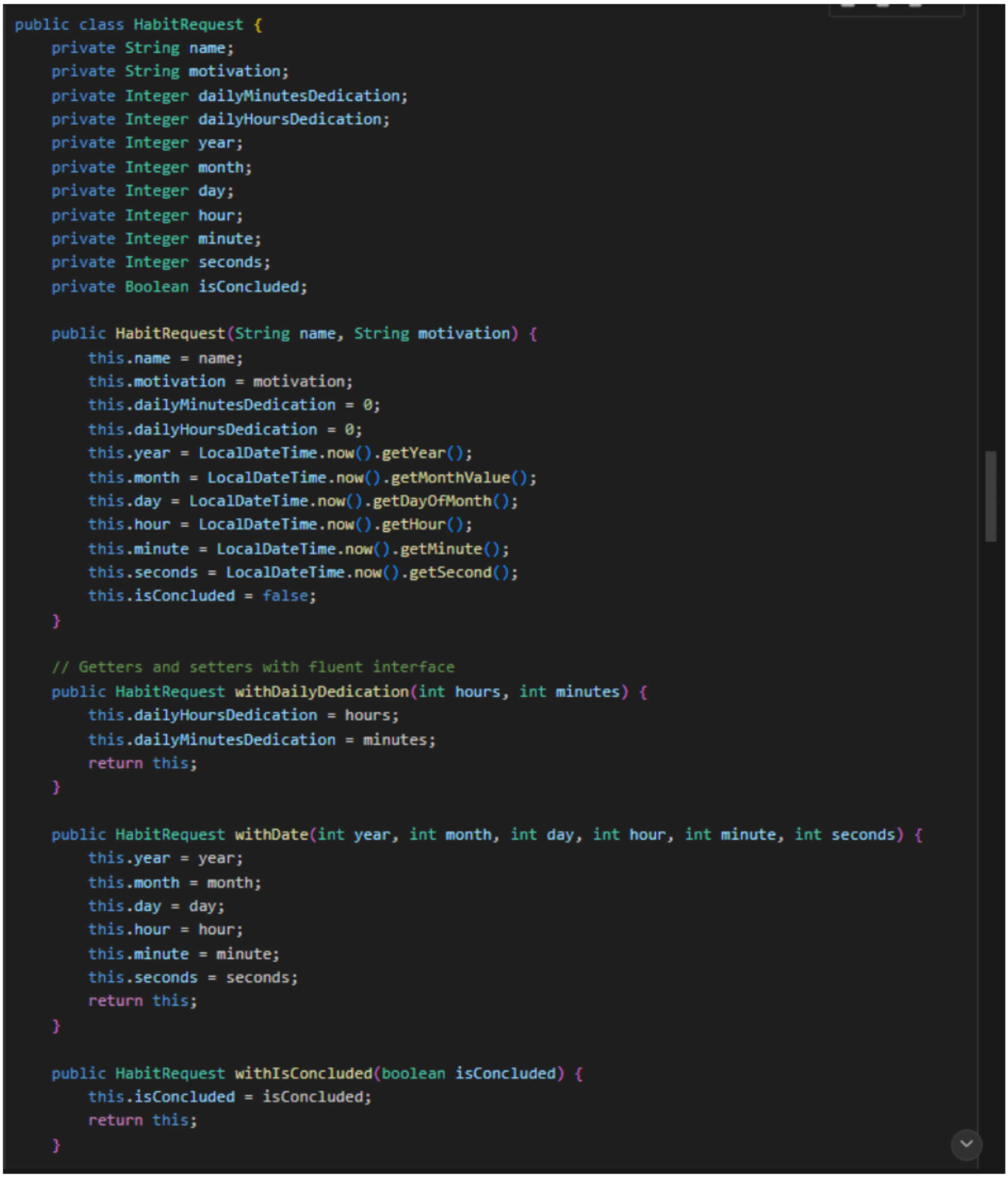
}

GitHub Copilot

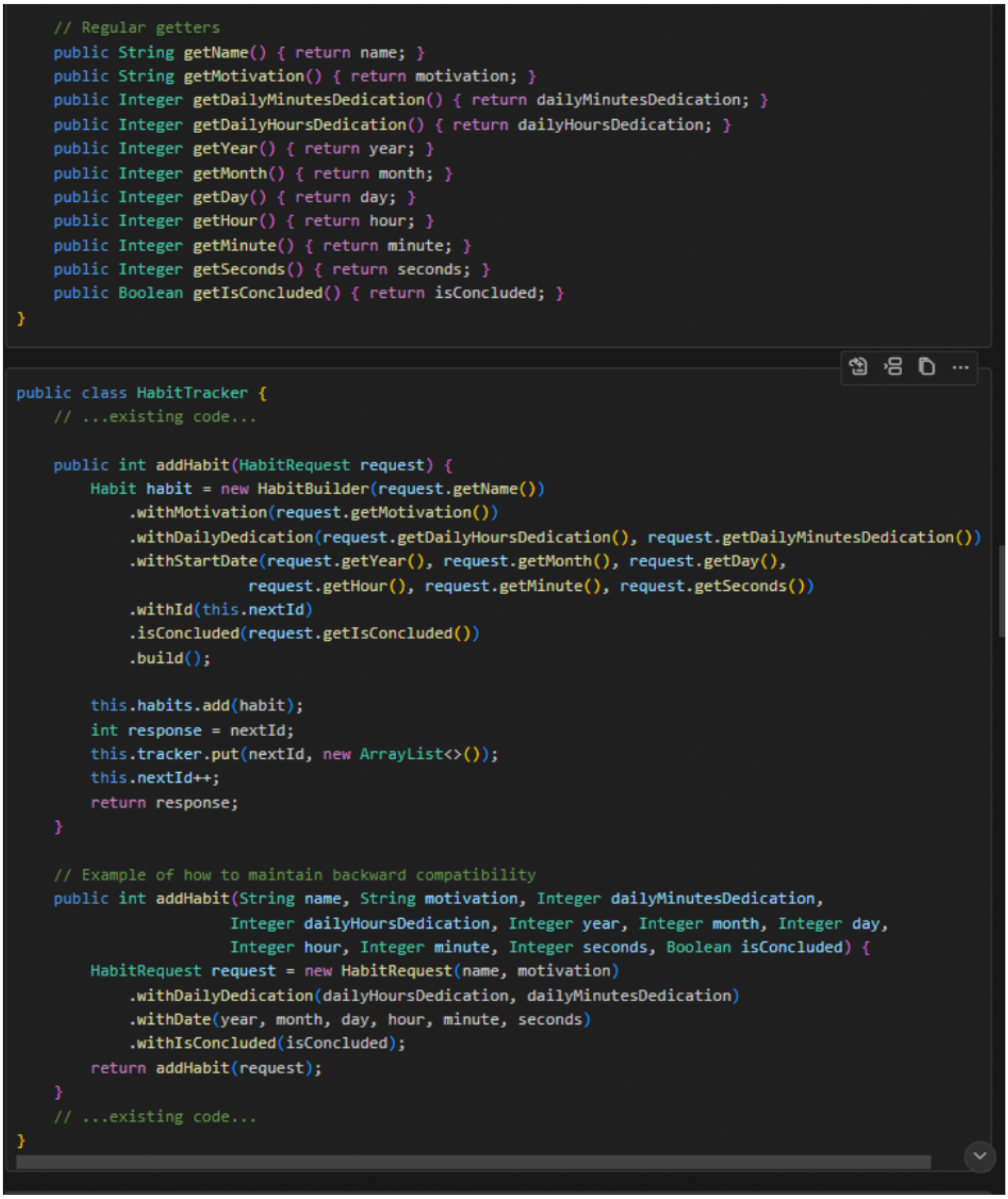
Step-by-Step Plan

1. Create a HabitRequest DTO class to encapsulate all parameters
2. Modify addHabit method to accept the DTO
3. Update the builder to work with the DTO
4. Modify any calling code to use the new DTO

Implementation

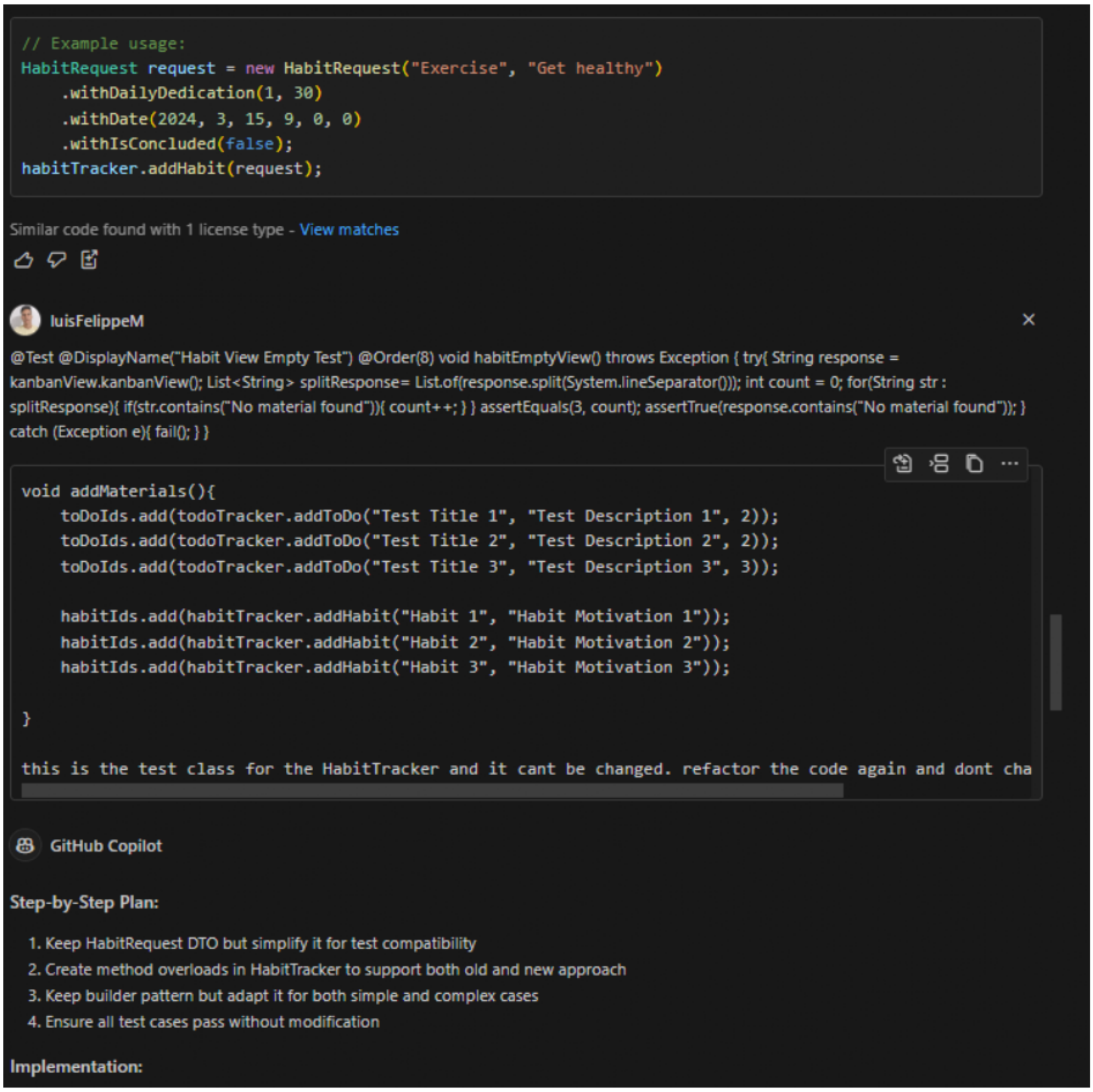


```java
public class HabitRequest {
    private String name;
    private String motivation;
    private Integer dailyMinutesDedication;
    private Integer dailyHoursDedication;
    private Integer year;
    private Integer month;
    private Integer day;
    private Integer hour;
    private Integer minute;
    private Integer seconds;
    private Boolean isConcluded;

    public HabitRequest(String name, String motivation) {
        this.name = name;
        this.motivation = motivation;
        this.dailyMinutesDedication = 0;
        this.dailyHoursDedication = 0;
        this.year = LocalDateTime.now().getYear();
        this.month = LocalDateTime.now().getMonthValue();
        this.day = LocalDateTime.now().getDayOfMonth();
        this.hour = LocalDateTime.now().getHour();
        this.minute = LocalDateTime.now().getMinute();
        this.seconds = LocalDateTime.now().getSecond();
        this.isConcluded = false;
    }

    // Getters and setters with fluent interface
    public HabitRequest withDailyDedication(int hours, int minutes) {
        this.dailyHoursDedication = hours;
        this.dailyMinutesDedication = minutes;
        return this;
    }

    public HabitRequest withDate(int year, int month, int day, int hour, int minute, int seconds) {
        this.year = year;
        this.month = month;
        this.day = day;
        this.hour = hour;
        this.minute = minute;
        this.seconds = seconds;
        return this;
    }

    public HabitRequest withIsConcluded(boolean isConcluded) {
        this.isConcluded = isConcluded;
        return this;
    }
```



```java
    // Regular getters
    public String getName() { return name; }
    public String getMotivation() { return motivation; }
    public Integer getDailyMinutesDedication() { return dailyMinutesDedication; }
    public Integer getDailyHoursDedication() { return dailyHoursDedication; }
    public Integer getYear() { return year; }
    public Integer getMonth() { return month; }
    public Integer getDay() { return day; }
    public Integer getHour() { return hour; }
    public Integer getMinute() { return minute; }
    public Integer getSeconds() { return seconds; }
    public Boolean getIsConcluded() { return isConcluded; }
}
```

```java
public class HabitTracker {
    // ...existing code...

    public int addHabit(HabitRequest request) {
        Habit habit = new HabitBuilder(request.getName())
            .withMotivation(request.getMotivation())
            .withDailyDedication(request.getDailyHoursDedication(), request.getDailyMinutesDedication())
            .withStartDate(request.getYear(), request.getMonth(), request.getDay(),
                         request.getHour(), request.getMinute(), request.getSeconds())
            .withId(this.nextId)
            .isConcluded(request.getIsConcluded())
            .build();

        this.habits.add(habit);
        int response = nextId;
        this.tracker.put(nextId, new ArrayList<>());
        this.nextId++;
        return response;
    }

    // Example of how to maintain backward compatibility
    public int addHabit(String name, String motivation, Integer dailyMinutesDedication,
                       Integer dailyHoursDedication, Integer year, Integer month, Integer day,
                       Integer hour, Integer minute, Integer seconds, Boolean isConcluded) {
        HabitRequest request = new HabitRequest(name, motivation)
            .withDailyDedication(dailyHoursDedication, dailyMinutesDedication)
            .withDate(year, month, day, hour, minute, seconds)
            .withIsConcluded(isConcluded);
        return addHabit(request);
    }
    // ...existing code...
}
```



```
// Example usage:
HabitRequest request = new HabitRequest("Exercise", "Get healthy")
    .withDailyDedication(1, 30)
    .withDate(2024, 3, 15, 9, 0, 0)
    .withIsConcluded(false);
habitTracker.addHabit(request);
```

Similar code found with 1 license type - View matches

luisFelippeM

@Test @DisplayName("Habit View Empty Test") @Order(8) void habitEmptyView() throws Exception { try{ String response = kanbanView.kanbanView(); List<String> splitResponse= List.of(response.split(System.lineSeparator())); int count = 0; for(String str : splitResponse){ if(str.contains("No material found")){ count++; } } assertEquals(3, count); assertTrue(response.contains("No material found")); } catch (Exception e){ fail(); } }

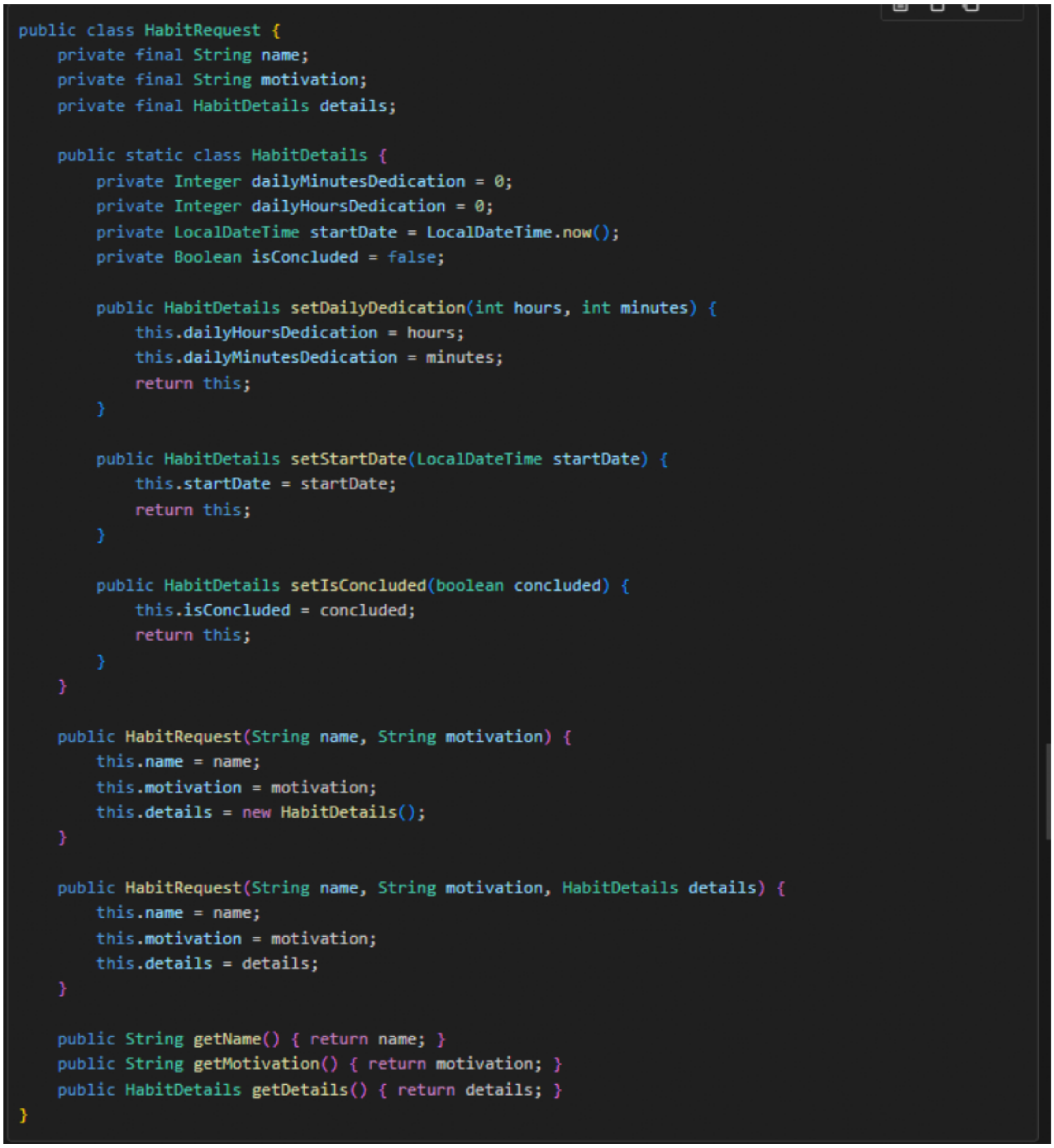
```
void addMaterials(){
    toDoIds.add(todoTracker.addToDo("Test Title 1", "Test Description 1", 2));
    toDoIds.add(todoTracker.addToDo("Test Title 2", "Test Description 2", 2));
    toDoIds.add(todoTracker.addToDo("Test Title 3", "Test Description 3", 3));

    habitIds.add(habitTracker.addHabit("Habit 1", "Habit Motivation 1"));
    habitIds.add(habitTracker.addHabit("Habit 2", "Habit Motivation 2"));
    habitIds.add(habitTracker.addHabit("Habit 3", "Habit Motivation 3"));

}

this is the test class for the HabitTracker and it cant be changed. refactor the code again and dont cha
```

GitHub Copilot

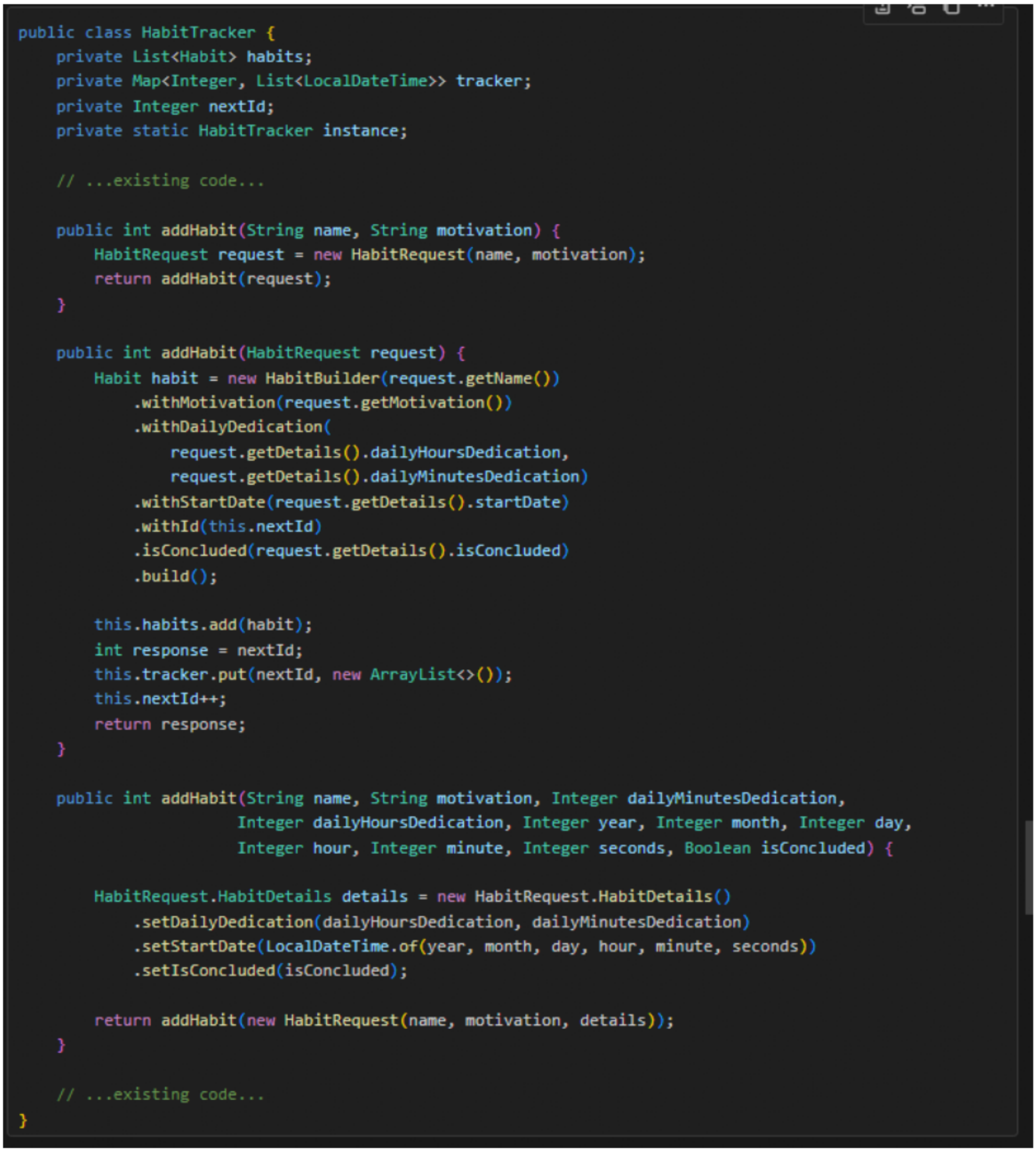
Step-by-Step Plan:

1. Keep HabitRequest DTO but simplify it for test compatibility
2. Create method overloads in HabitTracker to support both old and new approach
3. Keep builder pattern but adapt it for both simple and complex cases
4. Ensure all test cases pass without modification

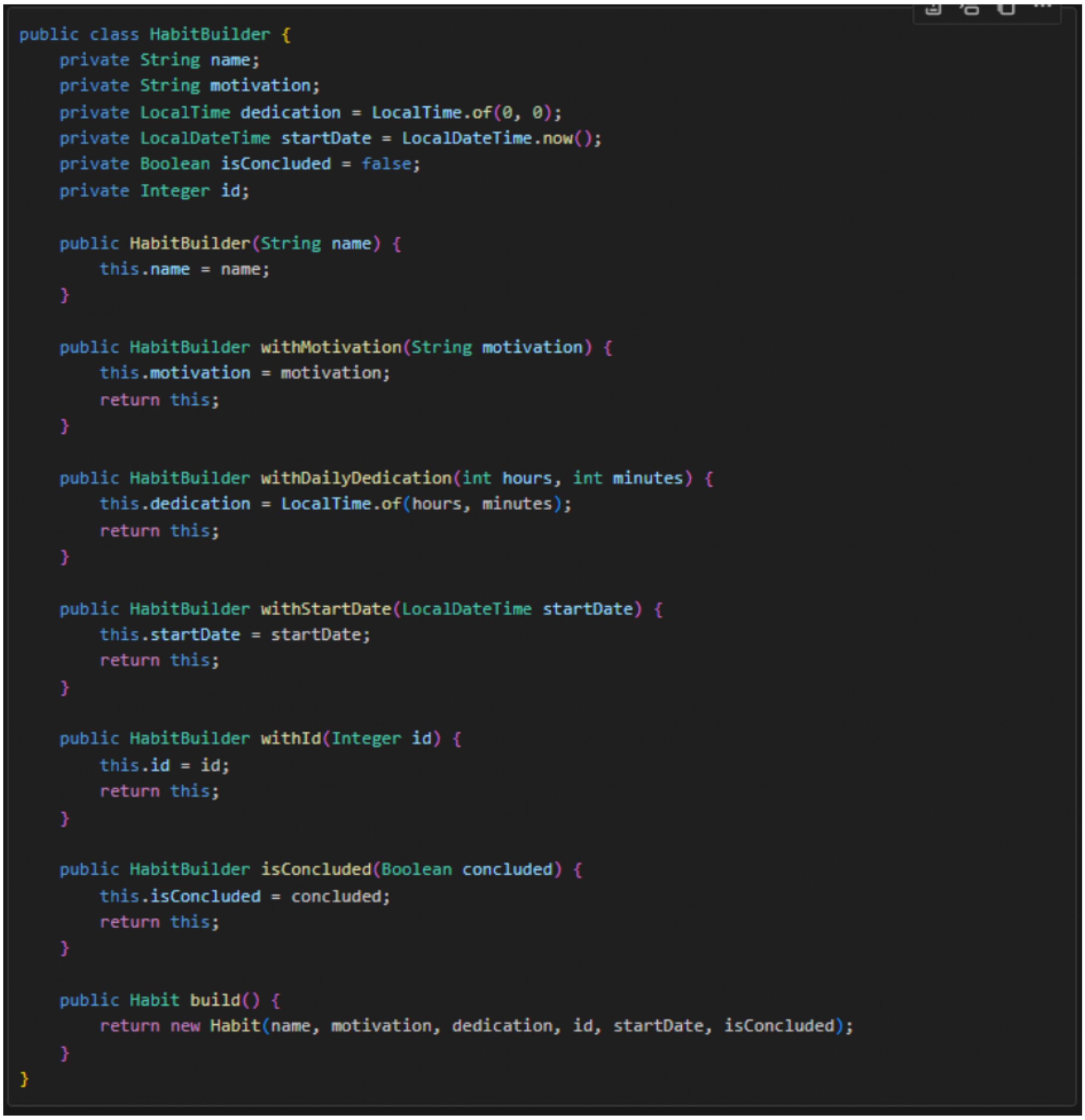
Implementation:



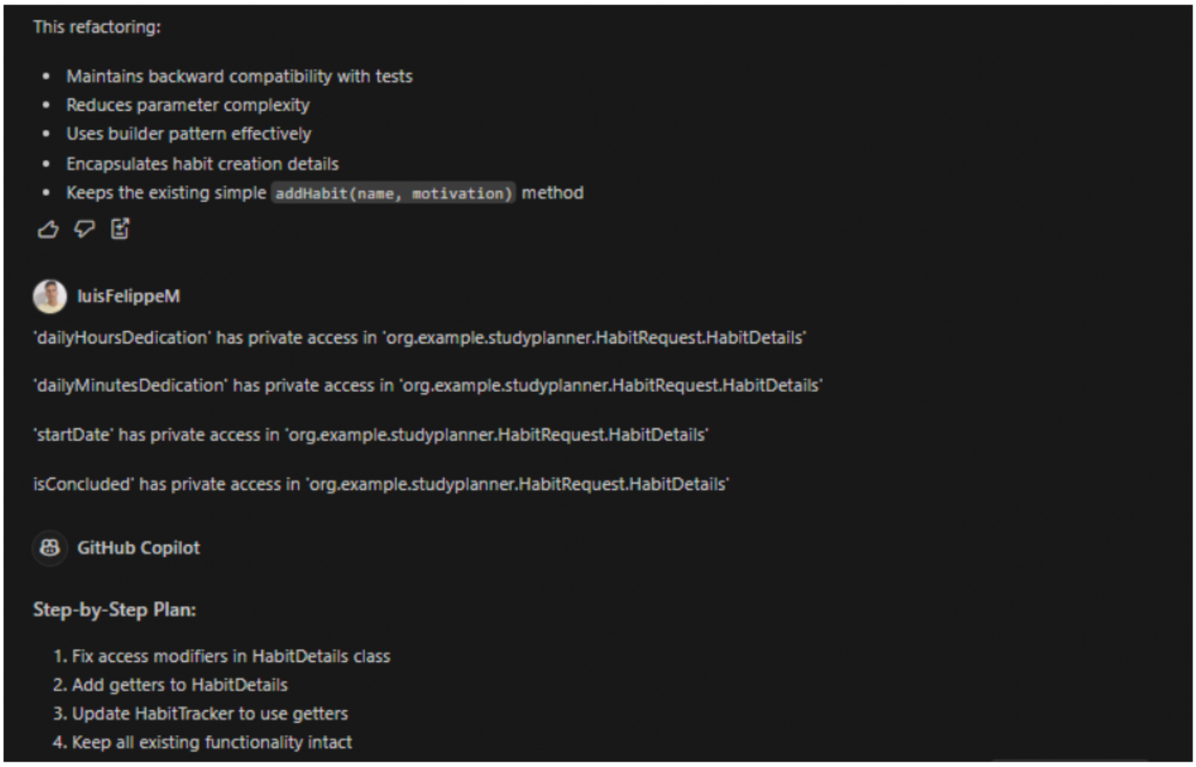
```
public class HabitRequest {
    private final String name;
    private final String motivation;
    private final HabitDetails details;

    public static class HabitDetails {
        private Integer dailyMinutesDedication = 0;
        private Integer dailyHoursDedication = 0;
        private LocalDateTime startDate = LocalDateTime.now();
        private Boolean isConcluded = false;

        public HabitDetails setDailyDedication(int hours, int minutes) {
            this.dailyHoursDedication = hours;
            this.dailyMinutesDedication = minutes;
            return this;
        }

        public HabitDetails setStartDate(LocalDateTime startDate) {
            this.startDate = startDate;
            return this;
        }

        public HabitDetails setIsConcluded(boolean concluded) {
            this.isConcluded = concluded;
            return this;
        }
    }

    public HabitRequest(String name, String motivation) {
        this.name = name;
        this.motivation = motivation;
        this.details = new HabitDetails();
    }

    public HabitRequest(String name, String motivation, HabitDetails details) {
        this.name = name;
        this.motivation = motivation;
        this.details = details;
    }

    public String getName() { return name; }
    public String getMotivation() { return motivation; }
    public HabitDetails getDetails() { return details; }
}
```



```
public class HabitTracker {
    private List<Habit> habits;
    private Map<Integer, List<LocalDateTime>> tracker;
    private Integer nextId;
    private static HabitTracker instance;

    // ...existing code...

    public int addHabit(String name, String motivation) {
        HabitRequest request = new HabitRequest(name, motivation);
        return addHabit(request);
    }

    public int addHabit(HabitRequest request) {
        Habit habit = new HabitBuilder(request.getName())
            .withMotivation(request.getMotivation())
            .withDailyDedication(
                request.getDetails().dailyHoursDedication,
                request.getDetails().dailyMinutesDedication)
            .withStartDate(request.getDetails().startDate)
            .withId(this.nextId)
            .isConcluded(request.getDetails().isConcluded)
            .build();

        this.habits.add(habit);
        int response = nextId;
        this.tracker.put(nextId, new ArrayList<>());
        this.nextId++;
        return response;
    }

    public int addHabit(String name, String motivation, Integer dailyMinutesDedication,
                       Integer dailyHoursDedication, Integer year, Integer month, Integer day,
                       Integer hour, Integer minute, Integer seconds, Boolean isConcluded) {

        HabitRequest.HabitDetails details = new HabitRequest.HabitDetails()
            .setDailyDedication(dailyHoursDedication, dailyMinutesDedication)
            .setStartDate(LocalDateTime.of(year, month, day, hour, minute, seconds))
            .setIsConcluded(isConcluded);

        return addHabit(new HabitRequest(name, motivation, details));
    }

    // ...existing code...
}
```



```java
public class HabitBuilder {
    private String name;
    private String motivation;
    private LocalTime dedication = LocalTime.of(0, 0);
    private LocalDateTime startDate = LocalDateTime.now();
    private Boolean isConcluded = false;
    private Integer id;

    public HabitBuilder(String name) {
        this.name = name;
    }

    public HabitBuilder withMotivation(String motivation) {
        this.motivation = motivation;
        return this;
    }

    public HabitBuilder withDailyDedication(int hours, int minutes) {
        this.dedication = LocalTime.of(hours, minutes);
        return this;
    }

    public HabitBuilder withStartDate(LocalDateTime startDate) {
        this.startDate = startDate;
        return this;
    }

    public HabitBuilder withId(Integer id) {
        this.id = id;
        return this;
    }

    public HabitBuilder isConcluded(Boolean concluded) {
        this.isConcluded = concluded;
        return this;
    }

    public Habit build() {
        return new Habit(name, motivation, dedication, id, startDate, isConcluded);
    }
}
```



This refactoring:

- Maintains backward compatibility with tests
- Reduces parameter complexity
- Uses builder pattern effectively
- Encapsulates habit creation details
- Keeps the existing simple `addHabit(name, motivation)` method

luisFelippeM

'dailyHoursDedication' has private access in 'org.example.studyplanner.HabitRequest.HabitDetails'

'dailyMinutesDedication' has private access in 'org.example.studyplanner.HabitRequest.HabitDetails'

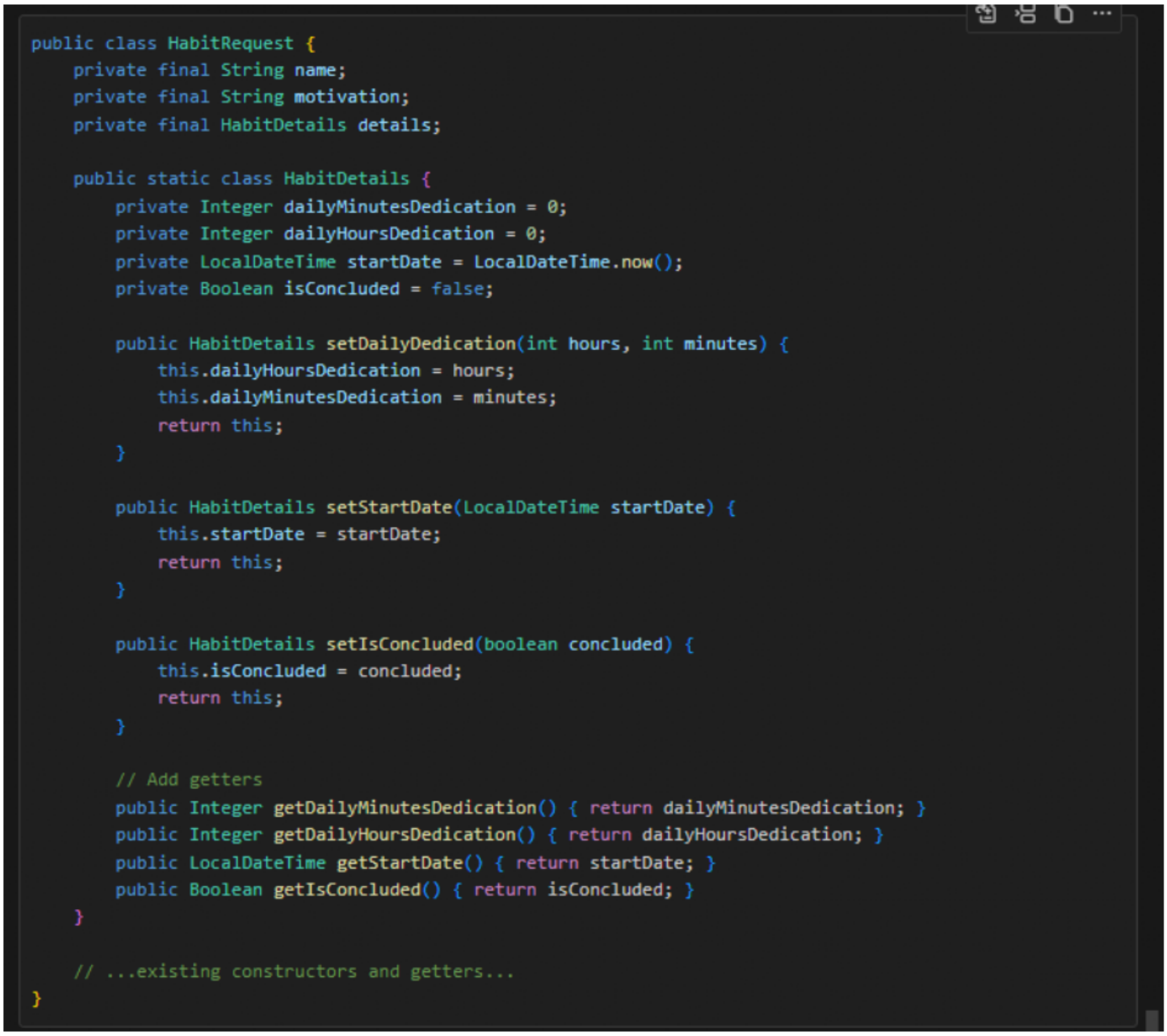
'startDate' has private access in 'org.example.studyplanner.HabitRequest.HabitDetails'

isConcluded' has private access in 'org.example.studyplanner.HabitRequest.HabitDetails'

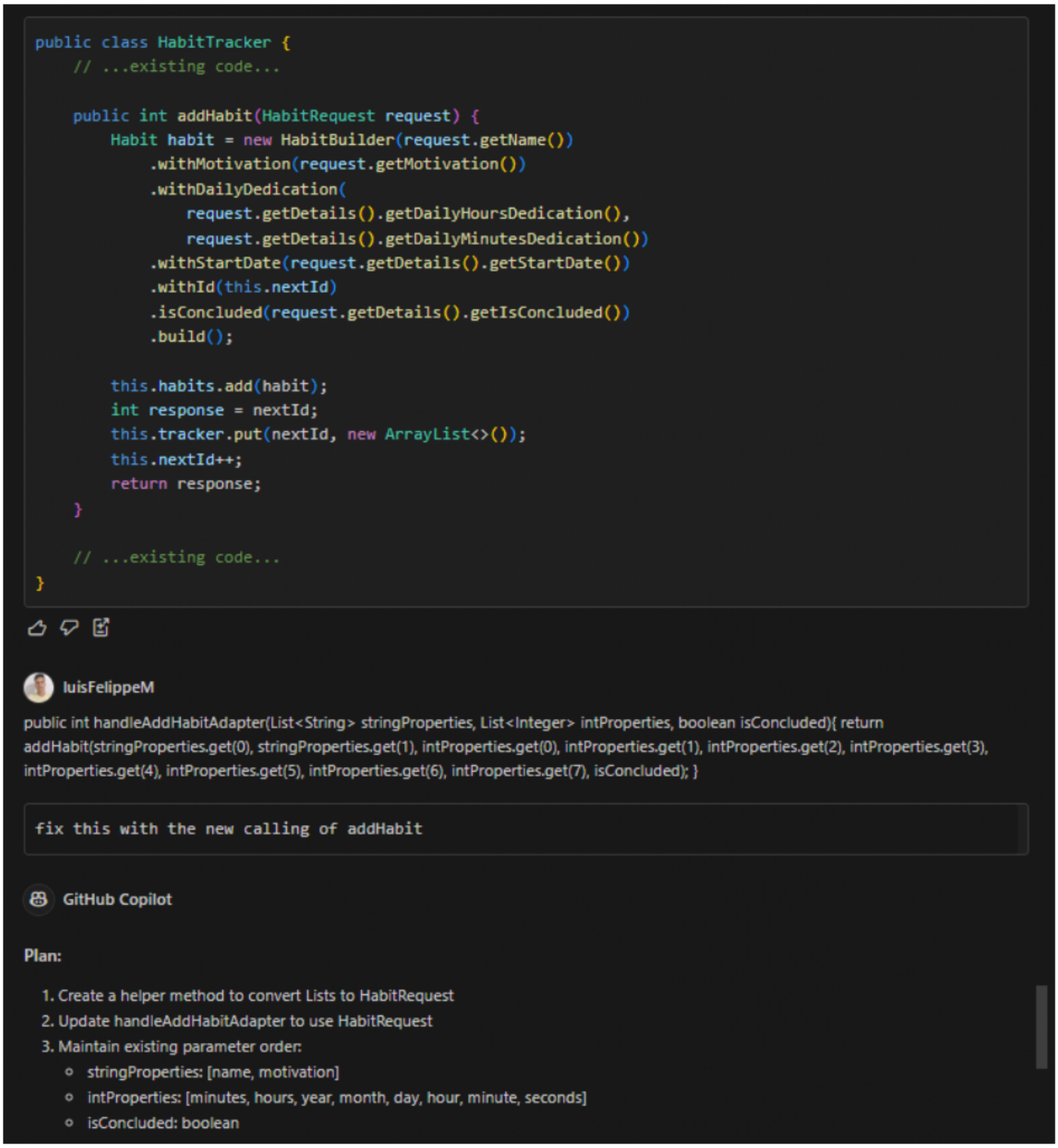
GitHub Copilot

Step-by-Step Plan:

1. Fix access modifiers in HabitDetails class
2. Add getters to HabitDetails
3. Update HabitTracker to use getters
4. Keep all existing functionality intact



```
public class HabitRequest {
    private final String name;
    private final String motivation;
    private final HabitDetails details;

    public static class HabitDetails {
        private Integer dailyMinutesDedication = 0;
        private Integer dailyHoursDedication = 0;
        private LocalDateTime startDate = LocalDateTime.now();
        private Boolean isConcluded = false;

        public HabitDetails setDailyDedication(int hours, int minutes) {
            this.dailyHoursDedication = hours;
            this.dailyMinutesDedication = minutes;
            return this;
        }

        public HabitDetails setStartDate(LocalDateTime startDate) {
            this.startDate = startDate;
            return this;
        }

        public HabitDetails setIsConcluded(boolean concluded) {
            this.isConcluded = concluded;
            return this;
        }

        // Add getters
        public Integer getDailyMinutesDedication() { return dailyMinutesDedication; }
        public Integer getDailyHoursDedication() { return dailyHoursDedication; }
        public LocalDateTime getStartDate() { return startDate; }
        public Boolean getIsConcluded() { return isConcluded; }
    }

    // ...existing constructors and getters...
}
```



```java
public class HabitTracker {
    // ...existing code...

    public int addHabit(HabitRequest request) {
        Habit habit = new HabitBuilder(request.getName())
            .withMotivation(request.getMotivation())
            .withDailyDedication(
                request.getDetails().getDailyHoursDedication(),
                request.getDetails().getDailyMinutesDedication())
            .withStartDate(request.getDetails().getStartDate())
            .withId(this.nextId)
            .isConcluded(request.getDetails().getIsConcluded())
            .build();

        this.habits.add(habit);
        int response = nextId;
        this.tracker.put(nextId, new ArrayList<>());
        this.nextId++;
        return response;
    }

    // ...existing code...
}
```

luisFelippeM

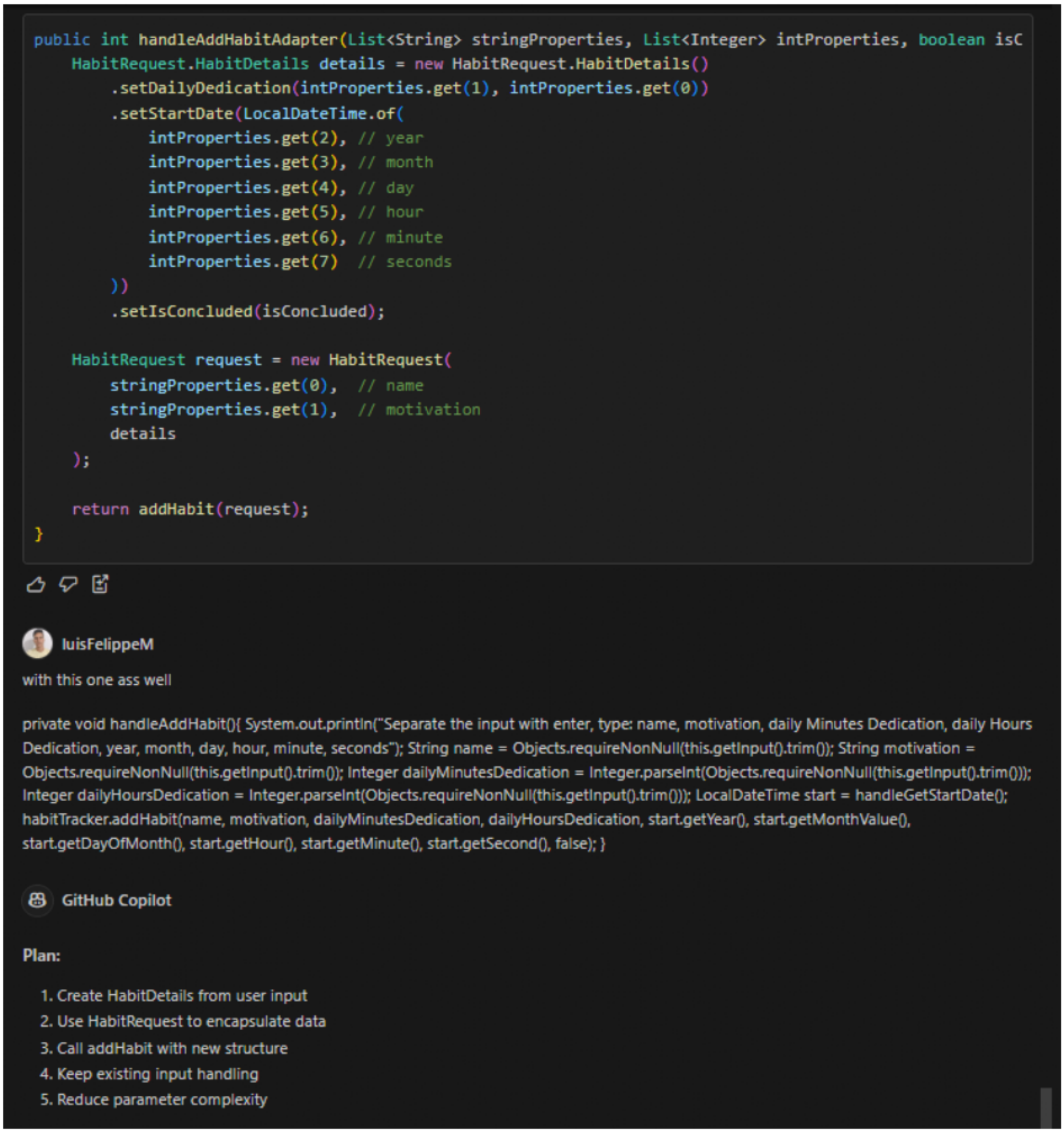
public int handleAddHabitAdapter(List<String> stringProperties, List<Integer> intProperties, boolean isConcluded){ return addHabit(stringProperties.get(0), stringProperties.get(1), intProperties.get(0), intProperties.get(1), intProperties.get(2), intProperties.get(3), intProperties.get(4), intProperties.get(5), intProperties.get(6), intProperties.get(7), isConcluded); }

```
fix this with the new calling of addHabit
```

GitHub Copilot

Plan:

1. Create a helper method to convert Lists to HabitRequest
2. Update handleAddHabitAdapter to use HabitRequest
3. Maintain existing parameter order:
    - stringProperties: [name, motivation]
    - intProperties: [minutes, hours, year, month, day, hour, minute, seconds]
    - isConcluded: boolean



```
public int handleAddHabitAdapter(List<String> stringProperties, List<Integer> intProperties, boolean isC
    HabitRequest.HabitDetails details = new HabitRequest.HabitDetails()
        .setDailyDedication(intProperties.get(1), intProperties.get(0))
        .setStartDate(LocalDateTime.of(
            intProperties.get(2), // year
            intProperties.get(3), // month
            intProperties.get(4), // day
            intProperties.get(5), // hour
            intProperties.get(6), // minute
            intProperties.get(7)  // seconds
        ))
        .setIsConcluded(isConcluded);

    HabitRequest request = new HabitRequest(
        stringProperties.get(0),  // name
        stringProperties.get(1),  // motivation
        details
    );

    return addHabit(request);
}
```

luisFelippeM

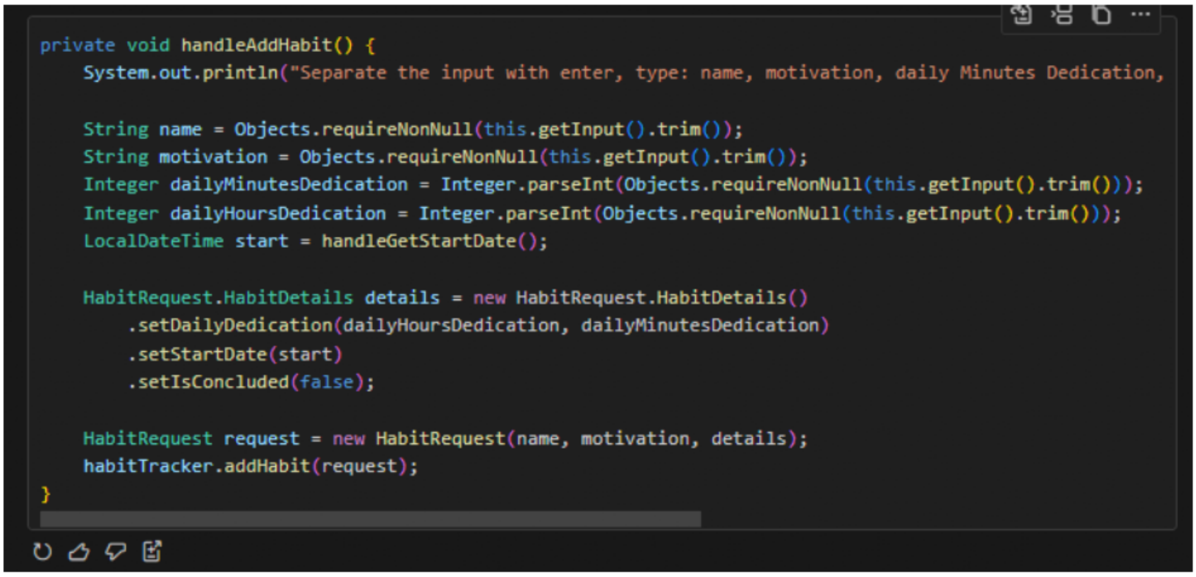
with this one ass well

private void handleAddHabit(){ System.out.println("Separate the input with enter, type: name, motivation, daily Minutes Dedication, daily Hours Dedication, year, month, day, hour, minute, seconds"); String name = Objects.requireNonNull(this.getInput().trim()); String motivation = Objects.requireNonNull(this.getInput().trim()); Integer dailyMinutesDedication = Integer.parseInt(Objects.requireNonNull(this.getInput().trim())); Integer dailyHoursDedication = Integer.parseInt(Objects.requireNonNull(this.getInput().trim())); LocalDateTime start = handleGetStartDate(); habitTracker.addHabit(name, motivation, dailyMinutesDedication, dailyHoursDedication, start.getYear(), start.getMonthValue(), start.getDayOfMonth(), start.getHour(), start.getMinute(), start.getSecond(), false); }

GitHub Copilot

Plan:

1. Create HabitDetails from user input
2. Use HabitRequest to encapsulate data
3. Call addHabit with new structure
4. Keep existing input handling
5. Reduce parameter complexity



```
private void handleAddHabit() {
    System.out.println("Separate the input with enter, type: name, motivation, daily Minutes Dedication,

    String name = Objects.requireNonNull(this.getInput().trim());
    String motivation = Objects.requireNonNull(this.getInput().trim());
    Integer dailyMinutesDedication = Integer.parseInt(Objects.requireNonNull(this.getInput().trim()));
    Integer dailyHoursDedication = Integer.parseInt(Objects.requireNonNull(this.getInput().trim()));
    LocalDateTime start = handleGetStartDate();

    HabitRequest.HabitDetails details = new HabitRequest.HabitDetails()
        .setDailyDedication(dailyHoursDedication, dailyMinutesDedication)
        .setStartDate(start)
        .setIsConcluded(false);

    HabitRequest request = new HabitRequest(name, motivation, details);
    habitTracker.addHabit(request);
}
```