



luisFelipeM

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

```
public class StudyPlan extends Registry{ private StudyObjective objective; private List<String> steps;
```

```
    public StudyPlan(String planName, StudyObjective objective, List<StudyMaterial> materials) {
        this.name = planName;
        this.objective = objective;
        this.steps = new ArrayList<>();
    }

    @Override
    public String toString(){
        return "Plan: " + name + ",\nObjective: " + objective.getDescription() + ",\nSteps: " + String.join(
    }

    public List<String> getSteps() {
        return steps;
    }

    public StudyObjective getObjective() {
        return objective;
    }

    public void assignObjective(StudyObjective objective) {
        this.objective = objective;
    }

    public void addSingleStep(String toAdd){
        steps.add(toAdd);
    }

    public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep, String seas
        String basicSteps, String mainObjectiveTitle, String mainGoalTitle, String mainM
        String mainTask, Integer numberofSteps, boolean isImportant, LocalDateTime start
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

        this.steps = new ArrayList<>(Arrays.asList(firstStep, resetStudyMechanism, consistentStep, seasonalS
    }

    public void handleAssignSteps(List<String> stringProperties, Integer numberofSteps, boolean isImportant,
        assignSteps(stringProperties.get(0), stringProperties.get(1), stringProperties.get(2), stringPropert
    }

}
```



GitHub Copilot

Let's refactor this code using the Builder pattern to address the long parameter list smell.

Steps:

1. Create a static nested Builder class
2. Move parameters to builder class
3. Create builder methods for each parameter
4. Create build() method
5. Modify StudyPlan constructor
6. Update existing methods

```
public class StudyPlan extends Registry {  
    private StudyObjective objective;  
    private List<String> steps;  
  
    private StudyPlan(Builder builder) {  
        this.name = builder.planName;  
        this.objective = builder.objective;  
        this.steps = builder.steps;  
    }  
  
    public static class Builder {  
        private String planName;  
        private StudyObjective objective;  
        private List<String> steps = new ArrayList<>();  
        private DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");  
  
        public Builder(String planName) {  
            this.planName = planName;  
        }  
  
        public Builder objective(StudyObjective objective) {  
            this.objective = objective;  
            return this;  
        }  
  
        public Builder addStep(String step) {  
            this.steps.add(step);  
            return this;  
        }  
  
        public Builder setSteps(List<String> steps) {  
            this.steps = new ArrayList<>(steps);  
            return this;  
        }  
  
        public Builder actionDetailedSteps(String firstStep, String resetStudyMechanism  
    }
```

```
public Builder assignDetailedSteps(String firstStep, String resetStudyMechanism,
                                  String consistentStep, String seasonalSteps,
                                  String basicSteps, String mainObjectiveTitle,
                                  String mainGoalTitle, String mainMaterialTopic,
                                  String mainTask, Integer numberOfSteps,
                                  boolean isImportant, LocalDateTime startDate,
                                  LocalDateTime endDate) {
    this.steps = new ArrayList<>(Arrays.asList(
        firstStep, resetStudyMechanism, consistentStep, seasonalSteps,
        basicSteps, "Number of steps: " + numberOfSteps.toString(),
        "Is it important to you? " + isImportant,
        startDate.format(formatter), endDate.format(formatter),
        mainObjectiveTitle, mainGoalTitle, mainMaterialTopic, mainTask
    )));
    return this;
}

public StudyPlan build() {
    return new StudyPlan(this);
}
}

@Override
public String toString() {
    return "Plan: " + name + ",\nObjective: " + objective.getDescription() + ",\nSteps: " + String.j
}

public List<String> getSteps() {
    return steps;
}

public StudyObjective getObjective() {
    return objective;
}

public void assignObjective(StudyObjective objective) {
    this.objective = objective;
}

public void addSingleStep(String toAdd) {
    steps.add(toAdd);
}
}
```

Usage example:

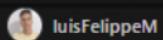
```

StudyPlan plan = new StudyPlan.Builder("My Study Plan")
    .objective(studyObjective)
    .assignDetailedSteps(
        "Step 1",
        "Reset",
        "Consistent",
        "Seasonal",
        "Basic",
        "Main Objective",
        "Main Goal",
        "Material Topic",
        "Main Task",
        5,
        true,
        LocalDateTime.now(),
        LocalDateTime.now().plusDays(30)
    )
    .build();

```

The builder pattern provides:

- More readable code
- Flexible object construction
- Immutable objects
- Method chaining
- Clear parameter organization



@TestMethodOrder(MethodOrderer.OrderAnnotation.class) class StudyPlanTest { StudyObjective studyObjective = new StudyObjective("Test Study Objective", "Test Study Objective"); StudyPlan studyPlan = new StudyPlan("Test Plan", studyObjective, new ArrayList<>()); LocalDateTime startDateTest = null; DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

```

@BeforeEach
void setUp() {
    assignSteps();
}

void assignSteps(){
    List<String> stringProperties = List.of("firstStep", "resetStudyMechanism", "consistentStep", "seasonalStep", "basicSteps", "mainObjectiveTitle", "mainGoalTitle", "mainMaterialTopic", "mainTask");
    Integer numberofSteps = 20;
    boolean isImportant = true;
    startDateTest = LocalDateTime.now();
    LocalDateTime startDate = startDateTest;
    LocalDateTime endDate = startDate.plusDays(10);
    studyPlan.handleAssignSteps(stringProperties, numberofSteps, isImportant, startDate, endDate);
}

boolean verifyStepsResponse(String step, List<String> response) {
    for(String object : response){
        if(object.contains(step)){
            return true;
        }
    }
    return false;
}

```

```
@Test
@Order(1)
@DisplayName("Handle Assign First Step Test")
void handleAssignStepsFirstStepTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("firstStep", steps));
}

@Test
@Order(2)
@DisplayName("Handle Assign Reset Study Mechanism Test")
void handleAssignStepsResetStudyMechanismTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("resetStudyMechanism", steps));
}

@Test
@Order(3)
@DisplayName("Handle Assign Reset Consistent Step Test")
void handleAssignStepsConsistentStepTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("consistentStep", steps));
}

@Test
@Order(4)
@DisplayName("Handle Assign Reset Seasonal Steps Test")
void handleAssignStepsSeasonalStepsTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("seasonalSteps", steps));
}

@Test
@Order(5)
@DisplayName("Handle Assign Reset Basic Steps Test")
void handleAssignStepsBasicStepsTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("basicSteps", steps));
}

@Test
@Order(6)
@DisplayName("Handle Assign Reset Main Objective Title Test")
void handleAssignStepsMainObjectiveTitleTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("mainObjectiveTitle", steps));
}
```

```
@Test
@Order(7)
@DisplayName("Handle Assign Reset Main Goal Title Test")
void handleAssignStepsMainGoalTitleTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("mainGoalTitle", steps));
}

@Test
@Order(8)
@DisplayName("Handle Assign Reset Main Material Topic Test")
void handleAssignStepsMainMaterialTopicTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("mainMaterialTopic", steps));
}

@Test
@Order(9)
@DisplayName("Handle Assign Reset Main Task Test")
void handleAssignStepsMainTaskTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("mainTask", steps));
}

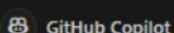
@Test
@Order(10)
@DisplayName("Handle Assign Number Of Steps Test")
void handleAssignStepsNumberOfStepsTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse("20", steps));
}

@Test
@Order(11)
@DisplayName("Handle Assign Start Date Test")
void handleAssignStepsStartDateTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse(startDateTest.format(formatter), steps));
}

@Test
@Order(12)
@DisplayName("Handle Assign End Date Test")
void handleAssignStepsEndDateTest() {
    List<String> steps = studyPlan.getSteps();
    assertTrue(verifyStepsResponse(startDateTest.plusDays(10).format(formatter), steps));
}
```

```
@Test  
 @Order(13)  
 @DisplayName("Handle Assign Is Important Test")  
 void handleAssignStepsIsImportantTest() {  
     List<String> steps = studyPlan.getSteps();  
     assertTrue(verifyStepsResponse("true", steps));  
 }
```

this is the test class for StudyPlan and it can't be changed



Let's plan the Builder pattern refactoring while maintaining test compatibility:

Steps:

1. Create inner StepsBuilder class for step-specific properties
2. Create inner StudyPlanBuilder class for main plan construction
3. Keep existing methods for test compatibility
4. Add builder methods for gradual construction
5. Create build() methods

Implementation Plan:

1. StudyPlanSteps - to handle step-related data
2. StudyPlan - main class with builder support
3. Maintain original method signatures
4. Add builder pattern while keeping existing functionality

```
public class StudyPlan extends Registry {
    private StudyObjective objective;
    private List<String> steps;
    private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    // Keep original constructor for test compatibility
    public StudyPlan(String planName, StudyObjective objective, List<StudyMaterial> materials) {
        this.name = planName;
        this.objective = objective;
        this.steps = new ArrayList<>();
    }

    // Private constructor for builder
    private StudyPlan(StudyPlanBuilder builder) {
        this.name = builder.name;
        this.objective = builder.objective;
        this.steps = builder.stepsBuilder.build();
    }

    // Inner class for steps management
    private static class StepsBuilder {
        private final List<String> steps = new ArrayList<>();
        private final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

        public void addStep(String step) {
            steps.add(step);
        }

        public void setDetailedSteps(String firstStep, String resetStudyMechanism,
                                    String consistentStep, String seasonalSteps,
                                    String basicSteps, String mainObjectiveTitle,
                                    String mainGoalTitle, String mainMaterialTopic,
                                    String mainTask, Integer numberOfSteps,
                                    boolean isImportant, LocalDateTime startDate,
                                    LocalDateTime endDate) {
            steps.clear();
            steps.addAll(Arrays.asList(
                firstStep, resetStudyMechanism, consistentStep, seasonalSteps,
                basicSteps, "Number of steps: " + numberOfSteps,
                "Is it important to you? " + isImportant,
                startDate.format(formatter), endDate.format(formatter),
                mainObjectiveTitle, mainGoalTitle, mainMaterialTopic, mainTask
            ));
        }
    }

    public List<String> build() {
        return new ArrayList<>(steps);
    }
}
```

```
// Builder class
public static class StudyPlanBuilder {
    private String name;
    private StudyObjective objective;
    private final StepsBuilder stepsBuilder = new StepsBuilder();

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

    public StudyPlanBuilder objective(StudyObjective objective) {
        this.objective = objective;
        return this;
    }

    public StudyPlanBuilder addStep(String step) {
        stepsBuilder.addStep(step);
        return this;
    }

    public StudyPlan build() {
        return new StudyPlan(this);
    }
}

// Original methods for test compatibility
@Override
public String toString() {
    return "Plan: " + name + ",\nObjective: " + objective.getDescription() + ",\nSteps: " + String.j
}

public List<String> getSteps() {
    return steps;
}

public StudyObjective getObjective() {
    return objective;
}

public void assignObjective(StudyObjective objective) {
    this.objective = objective;
}

public void addSingleStep(String toAdd) {
    steps.add(toAdd);
}

public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep,
    String seasonalSteps, String basicSteps, String mainObjectiveTitle,
```

```

    public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep,
                           String seasonalSteps, String basicSteps, String mainObjectiveTitle,
                           String mainGoalTitle, String mainMaterialTopic, String mainTask,
                           Integer numberOfSteps, boolean isImportant,
                           LocalDateTime startDate, LocalDateTime endDate) {
        StepsBuilder stepsBuilder = new StepsBuilder();
        stepsBuilder.setDetailedSteps(firstStep, resetStudyMechanism, consistentStep,
                                      seasonalSteps, basicSteps, mainObjectiveTitle,
                                      mainGoalTitle, mainMaterialTopic, mainTask,
                                      numberOfSteps, isImportant, startDate, endDate);
        this.steps = stepsBuilder.build();
    }

    public void handleAssignSteps(List<String> stringProperties, Integer numberOfSteps,
                                 boolean isImportant, LocalDateTime startDate, LocalDateTime endDate) {
        assignSteps(stringProperties.get(0), stringProperties.get(1), stringProperties.get(2),
                   stringProperties.get(3), stringProperties.get(4), stringProperties.get(5),
                   stringProperties.get(6), stringProperties.get(7), stringProperties.get(8),
                   numberOfSteps, isImportant, startDate, endDate);
    }
}

```

 luisFelipeM X

```

public void setDetailedSteps(String firstStep, String resetStudyMechanism, String consistentStep, String seasonalSteps, String basicSteps, String mainObjectiveTitle, String mainGoalTitle, String mainMaterialTopic, String mainTask, Integer numberOfSteps, boolean isImportant, LocalDateTime startDate, LocalDateTime endDate) { steps.clear(); steps.addAll(Arrays.asList( firstStep, resetStudyMechanism, consistentStep, seasonalSteps, basicSteps, "Number of steps: " + numberOfSteps, "Is it important to you? " + isImportant, startDate.format(formatter), endDate.format(formatter), mainObjectiveTitle, mainGoalTitle, mainMaterialTopic, mainTask )); }

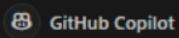
```

```

    public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep,
                           String seasonalSteps, String basicSteps, String mainObjectiveTitle,
                           String mainGoalTitle, String mainMaterialTopic, String mainTask,
                           Integer numberOfSteps, boolean isImportant,
                           LocalDateTime startDate, LocalDateTime endDate) {
        StepsBuilder stepsBuilder = new StepsBuilder();
        stepsBuilder.setDetailedSteps(firstStep, resetStudyMechanism, consistentStep,
                                      seasonalSteps, basicSteps, mainObjectiveTitle,
                                      mainGoalTitle, mainMaterialTopic, mainTask,
                                      numberOfSteps, isImportant, startDate, endDate);
        this.steps = stepsBuilder.build();
    }
}

```

now there are two code smells. fix them



Plan to Fix Code Smells

Identified Code Smells:

1. Long Parameter List in both methods
2. Duplicate Parameter Lists

Solution Steps:

1. Create a StepDetails class to encapsulate all parameters
2. Use the StepDetails class in both methods
3. Remove duplicate parameter lists

```
public class StepDetails {  
    private final String firstStep;  
    private final String resetStudyMechanism;  
    private final String consistentStep;  
    private final String seasonalSteps;  
    private final String basicSteps;  
    private final String mainObjectiveTitle;  
    private final String mainGoalTitle;  
    private final String mainMaterialTopic;  
    private final String mainTask;  
    private final Integer numberOfSteps;  
    private final boolean isImportant;  
    private final LocalDateTime startDate;  
    private final LocalDateTime endDate;  
  
    private StepDetails(Builder builder) {  
        this.firstStep = builder.firstStep;  
        this.resetStudyMechanism = builder.resetStudyMechanism;  
        this.consistentStep = builder.consistentStep;  
        this.seasonalSteps = builder.seasonalSteps;  
        this.basicSteps = builder.basicSteps;  
        this.mainObjectiveTitle = builder.mainObjectiveTitle;  
        this.mainGoalTitle = builder.mainGoalTitle;  
        this.mainMaterialTopic = builder.mainMaterialTopic;  
        this.mainTask = builder.mainTask;  
        this.numberOfSteps = builder.numberOfSteps;  
        this.isImportant = builder.isImportant;  
        this.startDate = builder.startDate;  
        this.endDate = builder.endDate;  
    }  
  
    public static class Builder {  
        private String firstStep;  
        private String resetStudyMechanism;
```

```
public static class Builder {
    private String firstStep;
    private String resetStudyMechanism;
    private String consistentStep;
    private String seasonalSteps;
    private String basicSteps;
    private String mainObjectiveTitle;
    private String mainGoalTitle;
    private String mainMaterialTopic;
    private String mainTask;
    private Integer numberOfSteps;
    private boolean isImportant;
    private LocalDateTime startDate;
    private LocalDateTime endDate;

    public Builder firstStep(String val) { firstStep = val; return this; }
    public Builder resetStudyMechanism(String val) { resetStudyMechanism = val; return this; }
    public Builder consistentStep(String val) { consistentStep = val; return this; }
    public Builder seasonalSteps(String val) { seasonalSteps = val; return this; }
    public Builder basicSteps(String val) { basicSteps = val; return this; }
    public Builder mainObjectiveTitle(String val) { mainObjectiveTitle = val; return this; }
    public Builder mainGoalTitle(String val) { mainGoalTitle = val; return this; }
    public Builder mainMaterialTopic(String val) { mainMaterialTopic = val; return this; }
    public Builder mainTask(String val) { mainTask = val; return this; }
    public Builder numberOfSteps(Integer val) { numberOfSteps = val; return this; }
    public Builder isImportant(boolean val) { isImportant = val; return this; }
    public Builder startDate(LocalDateTime val) { startDate = val; return this; }
    public Builder endDate(LocalDateTime val) { endDate = val; return this; }

    public StepDetails build() {
        return new StepDetails(this);
    }
}

// Getters
public String getFirstStep() { return firstStep; }
public String getResetStudyMechanism() { return resetStudyMechanism; }
public String getConsistentStep() { return consistentStep; }
public String getSeasonalSteps() { return seasonalSteps; }
public String getBasicSteps() { return basicSteps; }
public String getMainObjectiveTitle() { return mainObjectiveTitle; }
public String getMainGoalTitle() { return mainGoalTitle; }
public String getMainMaterialTopic() { return mainMaterialTopic; }
public String getMainTask() { return mainTask; }
public Integer getNumberOfSteps() { return numberOfSteps; }
public boolean isImportant() { return isImportant; }
public LocalDateTime getStartDate() { return startDate; }
public LocalDateTime getEndDate() { return endDate; }
```

```

// Refactored methods
public void setDetailedSteps(StepDetails details) {
    steps.clear();
    steps.addAll(Arrays.asList(
        details.getFirstStep(),
        details.getResetStudyMechanism(),
        details.getConsistentStep(),
        details.getSeasonalSteps(),
        details.getBasicSteps(),
        "Number of steps: " + details.getNumberOfSteps(),
        "Is it important to you? " + details.isImportant(),
        details.getStartDate().format(formatter),
        details.getEndDate().format(formatter),
        details.getMainObjectiveTitle(),
        details.getMainGoalTitle(),
        details.getMainMaterialTopic(),
        details.getMainTask()
    ));
}

public void assignSteps(StepDetails details) {
    StepsBuilder stepsBuilder = new StepsBuilder();
    stepsBuilder.setDetailedSteps(details);
    this.steps = stepsBuilder.build();
}

// Method to maintain compatibility with existing tests
public void handleAssignSteps(List<String> stringProperties, Integer numberOfSteps,
                               boolean isImportant, LocalDateTime startDate, LocalDateTime endDate) {
    StepDetails details = new StepDetails.Builder()
        .firstStep(stringProperties.get(0))
        .resetStudyMechanism(stringProperties.get(1))
        .consistentStep(stringProperties.get(2))
        .seasonalSteps(stringProperties.get(3))
        .basicSteps(stringProperties.get(4))
        .mainObjectiveTitle(stringProperties.get(5))
        .mainGoalTitle(stringProperties.get(6))
        .mainMaterialTopic(stringProperties.get(7))
        .mainTask(stringProperties.get(8))
        .numberOfSteps(numberOfSteps)
        .isImportant(isImportant)
        .startDate(startDate)
        .endDate(endDate)
        .build();

    assignSteps(details);
}

```

 luisFelipeM

fix this code to the new logic, deleting what is not needed and refactoring what is needed:

```
public class StudyPlan extends Registry { private StudyObjective objective; private List<String> steps; private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    // Keep original constructor for test compatibility
    public StudyPlan(String planName, StudyObjective objective, List<StudyMaterial> materials) {
        this.name = planName;
        this.objective = objective;
        this.steps = new ArrayList<>();
    }

    // Private constructor for builder
    private StudyPlan(StudyPlanBuilder builder) {
        this.name = builder.name;
        this.objective = builder.objective;
        this.steps = builder.stepsBuilder.build();
    }

    // Inner class for steps management
    private static class StepsBuilder {
        private final List<String> steps = new ArrayList<>();
        private final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

        public void addStep(String step) {
            steps.add(step);
        }

        public void setDetailedSteps(String firstStep, String resetStudyMechanism,
                                    String consistentStep, String seasonalSteps,
                                    String basicSteps, String mainObjectiveTitle,
                                    String mainGoalTitle, String mainMaterialTopic,
                                    String mainTask, Integer numberofSteps,
                                    boolean isImportant, LocalDateTime startDate,
                                    LocalDateTime endDate) {
            steps.clear();
            steps.addAll(Arrays.asList(
                firstStep, resetStudyMechanism, consistentStep, seasonalSteps,
                basicSteps, "Number of steps: " + numberofSteps,
                "Is it important to you? " + isImportant,
                startDate.format(formatter), endDate.format(formatter),
                mainObjectiveTitle, mainGoalTitle, mainMaterialTopic, mainTask
            ));
        }

        public List<String> build() {
            return new ArrayList<>(steps);
        }
    }
}
```

```
// Builder class
public static class StudyPlanBuilder {
    private String name;
    private StudyObjective objective;
    private final StepsBuilder stepsBuilder = new StepsBuilder();

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

    public StudyPlanBuilder objective(StudyObjective objective) {
        this.objective = objective;
        return this;
    }

    public StudyPlanBuilder addStep(String step) {
        stepsBuilder.addStep(step);
        return this;
    }

    public StudyPlan build() {
        return new StudyPlan(this);
    }
}

// Original methods for test compatibility
@Override
public String toString() {
    return "Plan: " + name + ",\nObjective: " + objective.getDescription() + ",\nSteps: " + String.join(
}

public List<String> getSteps() {
    return steps;
}

public StudyObjective getObjective() {
    return objective;
}

public void assignObjective(StudyObjective objective) {
    this.objective = objective;
}

public void addSingleStep(String toAdd) {
    steps.add(toAdd);
}

public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep,
    String seasonalSteps, String basicSteps, String mainObjectiveTitle)
```

```
public void assignSteps(String firstStep, String resetStudyMechanism, String consistentStep,
                      String seasonalSteps, String basicSteps, String mainObjectiveTitle,
                      String mainGoalTitle, String mainMaterialTopic, String mainTask,
                      Integer numberOfSteps, boolean isImportant,
                      LocalDateTime startDate, LocalDateTime endDate) {
    StepsBuilder stepsBuilder = new StepsBuilder();
    stepsBuilder.setDetailedSteps(firstStep, resetStudyMechanism, consistentStep,
                                 seasonalSteps, basicSteps, mainObjectiveTitle,
                                 mainGoalTitle, mainMaterialTopic, mainTask,
                                 numberOfSteps, isImportant, startDate, endDate);
    this.steps = stepsBuilder.build();
}

public class StepDetails {
    private final String firstStep;
    private final String resetStudyMechanism;
    private final String consistentStep;
    private final String seasonalSteps;
    private final String basicSteps;
    private final String mainObjectiveTitle;
    private final String mainGoalTitle;
    private final String mainMaterialTopic;
    private final String mainTask;
    private final Integer numberOfSteps;
    private final boolean isImportant;
    private final LocalDateTime startDate;
    private final LocalDateTime endDate;

    private StepDetails(Builder builder) {
        this.firstStep = builder.firstStep;
        this.resetStudyMechanism = builder.resetStudyMechanism;
        this.consistentStep = builder.consistentStep;
        this.seasonalSteps = builder.seasonalSteps;
        this.basicSteps = builder.basicSteps;
        this.mainObjectiveTitle = builder.mainObjectiveTitle;
        this.mainGoalTitle = builder.mainGoalTitle;
        this.mainMaterialTopic = builder.mainMaterialTopic;
        this.mainTask = builder.mainTask;
        this.numberOfSteps = builder.numberOfSteps;
        this.isImportant = builder.isImportant;
        this.startDate = builder.startDate;
        this.endDate = builder.endDate;
    }
}

public static class Builder {
    private String firstStep;
    private String resetStudyMechanism;
    private String consistentStep;
    private String seasonalSteps;
    private String basicSteps;
    private String mainObjectiveTitle;
    private String mainGoalTitle;
    private String mainMaterialTopic;
    private String mainTask;
    private Integer numberOfSteps;
    private Boolean isImportant;
    private LocalDateTime startDate;
    private LocalDateTime endDate;
}
```

```
public static class Builder {
    private String firstStep;
    private String resetStudyMechanism;
    private String consistentStep;
    private String seasonalSteps;
    private String basicSteps;
    private String mainObjectiveTitle;
    private String mainGoalTitle;
    private String mainMaterialTopic;
    private String mainTask;
    private Integer numberOfSteps;
    private boolean isImportant;
    private LocalDateTime startDate;
    private LocalDateTime endDate;

    public Builder firstStep(String val) { firstStep = val; return this; }
    public Builder resetStudyMechanism(String val) { resetStudyMechanism = val; return this; }
    public Builder consistentStep(String val) { consistentStep = val; return this; }
    public Builder seasonalSteps(String val) { seasonalSteps = val; return this; }
    public Builder basicSteps(String val) { basicSteps = val; return this; }
    public Builder mainObjectiveTitle(String val) { mainObjectiveTitle = val; return this; }
    public Builder mainGoalTitle(String val) { mainGoalTitle = val; return this; }
    public Builder mainMaterialTopic(String val) { mainMaterialTopic = val; return this; }
    public Builder mainTask(String val) { mainTask = val; return this; }
    public Builder numberOfSteps(Integer val) { numberOfSteps = val; return this; }
    public Builder isImportant(boolean val) { isImportant = val; return this; }
    public Builder startDate(LocalDateTime val) { startDate = val; return this; }
    public Builder endDate(LocalDateTime val) { endDate = val; return this; }

    public StepDetails build() {
        return new StepDetails(this);
    }
}

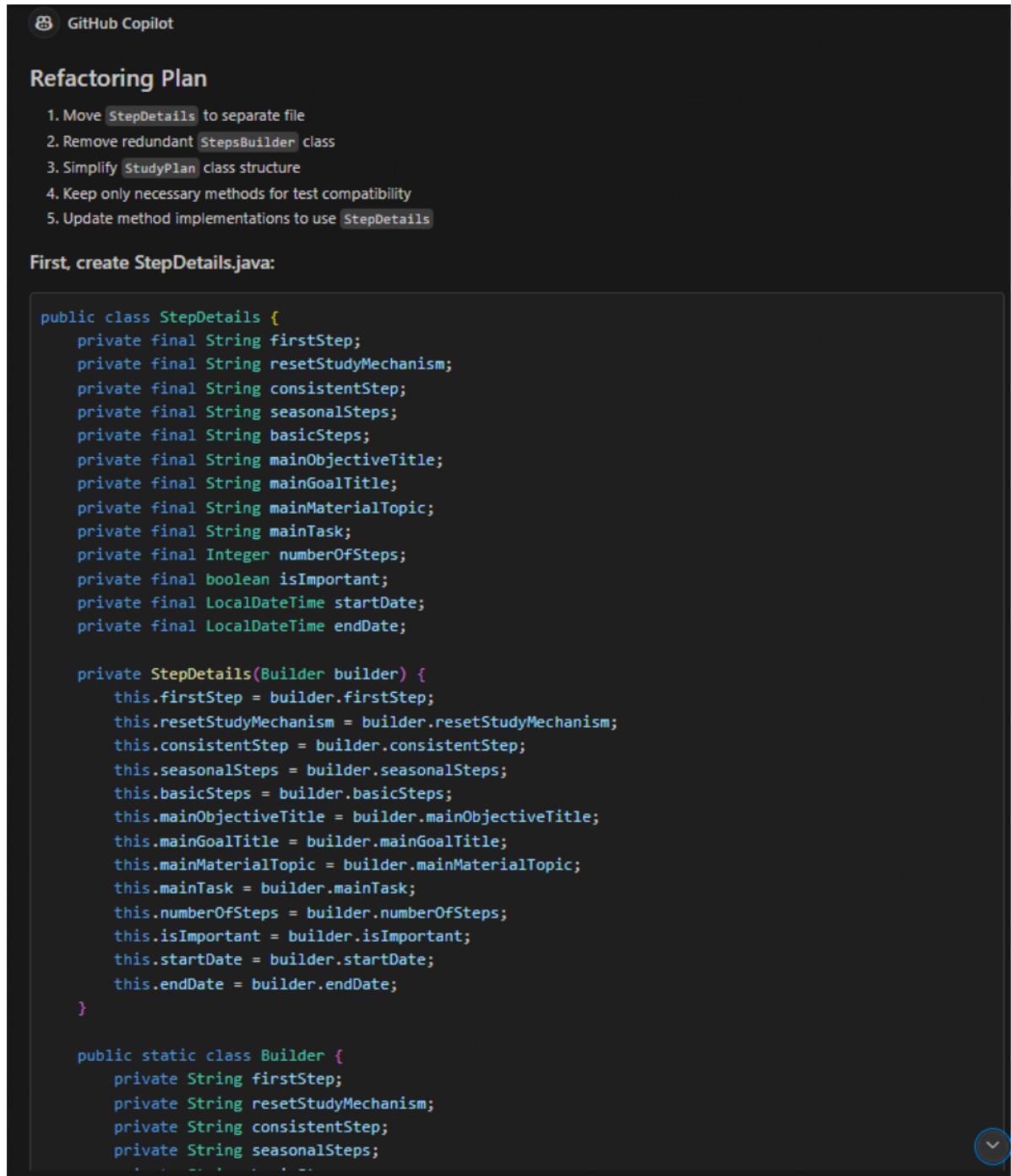
// Getters
public String getFirstStep() { return firstStep; }
public String getResetStudyMechanism() { return resetStudyMechanism; }
public String getConsistentStep() { return consistentStep; }
public String getSeasonalSteps() { return seasonalSteps; }
public String getBasicSteps() { return basicSteps; }
public String getMainObjectiveTitle() { return mainObjectiveTitle; }
public String getMainGoalTitle() { return mainGoalTitle; }
public String getMainMaterialTopic() { return mainMaterialTopic; }
public String getMainTask() { return mainTask; }
public Integer getNumberOfSteps() { return numberOfSteps; }
public boolean isImportant() { return isImportant; }
public LocalDateTime getStartDate() { return startDate; }
public LocalDateTime getEndDate() { return endDate; }
}
```

```
// Refactored methods
public void setDetailedSteps(StepDetails details) {
    steps.clear();
    steps.addAll(Arrays.asList(
        details.getFirstStep(),
        details.getResetStudyMechanism(),
        details.getConsistentStep(),
        details.getSeasonalSteps(),
        details.getBasicSteps(),
        "Number of steps: " + details.getNumberOfSteps(),
        "Is it important to you? " + details.isImportant(),
        details.getStartDate().format(formatter),
        details.getEndDate().format(formatter),
        details.getMainObjectiveTitle(),
        details.getMainGoalTitle(),
        details.getMainMaterialTopic(),
        details.getMainTask()
    ));
}

public void assignSteps(StepDetails details) {
    StepsBuilder stepsBuilder = new StepsBuilder();
    stepsBuilder.setDetailedSteps(details);
    this.steps = stepsBuilder.build();
}

// Method to maintain compatibility with existing tests
public void handleAssignSteps(List<String> stringProperties, Integer numberOfSteps,
                             boolean isImportant, LocalDateTime startDate, LocalDateTime endDate) {
    StepDetails details = new StepDetails.Builder()
        .firstStep(stringProperties.get(0))
        .resetStudyMechanism(stringProperties.get(1))
        .consistentStep(stringProperties.get(2))
        .seasonalSteps(stringProperties.get(3))
        .basicSteps(stringProperties.get(4))
        .mainObjectiveTitle(stringProperties.get(5))
        .mainGoalTitle(stringProperties.get(6))
        .mainMaterialTopic(stringProperties.get(7))
        .mainTask(stringProperties.get(8))
        .numberOfSteps(numberOfSteps)
        .isImportant(isImportant)
        .startDate(startDate)
        .endDate(endDate)
        .build();

    assignSteps(details);
}
```



```
private String consistentStep;
private String seasonalSteps;
private String basicSteps;
private String mainObjectiveTitle;
private String mainGoalTitle;
private String mainMaterialTopic;
private String mainTask;
private Integer numberOfSteps;
private boolean isImportant;
private LocalDateTime startDate;
private LocalDateTime endDate;

public Builder firstStep(String val) { firstStep = val; return this; }
public Builder resetStudyMechanism(String val) { resetStudyMechanism = val; return this; }
public Builder consistentStep(String val) { consistentStep = val; return this; }
public Builder seasonalSteps(String val) { seasonalSteps = val; return this; }
public Builder basicSteps(String val) { basicSteps = val; return this; }
public Builder mainObjectiveTitle(String val) { mainObjectiveTitle = val; return this; }
public Builder mainGoalTitle(String val) { mainGoalTitle = val; return this; }
public Builder mainMaterialTopic(String val) { mainMaterialTopic = val; return this; }
public Builder mainTask(String val) { mainTask = val; return this; }
public Builder numberOfSteps(Integer val) { numberOfSteps = val; return this; }
public Builder isImportant(boolean val) { isImportant = val; return this; }
public Builder startDate(LocalDateTime val) { startDate = val; return this; }
public Builder endDate(LocalDateTime val) { endDate = val; return this; }

public StepDetails build() {
    return new StepDetails(this);
}
}

// Getters
public String getFirstStep() { return firstStep; }
public String getResetStudyMechanism() { return resetStudyMechanism; }
public String getConsistentStep() { return consistentStep; }
public String getSeasonalSteps() { return seasonalSteps; }
public String getBasicSteps() { return basicSteps; }
public String getMainObjectiveTitle() { return mainObjectiveTitle; }
public String getMainGoalTitle() { return mainGoalTitle; }
public String getMainMaterialTopic() { return mainMaterialTopic; }
public String getMainTask() { return mainTask; }
public Integer getNumberOfSteps() { return numberOfSteps; }
public boolean isImportant() { return isImportant; }
public LocalDateTime getStartDate() { return startDate; }
public LocalDateTime getEndDate() { return endDate; }
}
```

Then, update StudyPlan.java:

```
public class StudyPlan extends Registry {
    private StudyObjective objective;
    private List<String> steps;
    private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss")

    public StudyPlan(String planName, StudyObjective objective, List<StudyMaterial> materials) {
        this.name = planName;
        this.objective = objective;
        this.steps = new ArrayList<>();
    }

    @Override
    public String toString() {
        return "Plan: " + name + ",\nObjective: " + objective.getDescription() + ",\nSteps: " + String.j
    }

    public List<String> getSteps() {
        return steps;
    }

    public StudyObjective getObjective() {
        return objective;
    }

    public void assignObjective(StudyObjective objective) {
        this.objective = objective;
    }

    public void addSingleStep(String toAdd) {
        steps.add(toAdd);
    }

    private List<String> createStepsList(StepDetails details) {
        return Arrays.asList(
            details.getFirstStep(),
            details.getResetStudyMechanism(),
            details.getConsistentStep(),
            details.getSeasonalSteps(),
            details.getBasicSteps(),
            "Number of steps: " + details.getNumberOfSteps(),
            "Is it important to you? " + details.isImportant(),
            details.getStartDate().format(formatter),
            details.getEndDate().format(formatter),
            details.getMainObjectiveTitle(),
            details.getMainGoalTitle(),
            details.getMainMaterialTopic(),
            details.getMainTopic()
        );
    }
}
```

```

        details.getMainMaterialTopic(),
        details.getMainTask()
    );
}

public void assignSteps(StepDetails details) {
    this.steps = new ArrayList<>(createStepsList(details));
}

public void handleAssignSteps(List<String> stringProperties, Integer numberOfSteps,
                             boolean isImportant, LocalDateTime startDate, LocalDateTime endDate) {
    StepDetails details = new StepDetails.Builder()
        .firstStep(stringProperties.get(0))
        .resetStudyMechanism(stringProperties.get(1))
        .consistentStep(stringProperties.get(2))
        .seasonalSteps(stringProperties.get(3))
        .basicSteps(stringProperties.get(4))
        .mainObjectiveTitle(stringProperties.get(5))
        .mainGoalTitle(stringProperties.get(6))
        .mainMaterialTopic(stringProperties.get(7))
        .mainTask(stringProperties.get(8))
        .numberOfSteps(numberOfSteps)
        .isImportant(isImportant)
        .startDate(startDate)
        .endDate(endDate)
        .build();

    assignSteps(details);
}
}

```

 luisFelipeM X

```
private void handleSetSteps(StudyPlan studyPlan){ handleMethodHeader("Study Plan Edit"); System.out.println("Type the following info: String firstStep, String resetStudyMechanism, String consistentStep, " + "String seasonalSteps, String basicSteps, String mainObjectiveTitle, String mainGoalTitle, String mainMaterialTopic, " + "String mainTask, @NotNull Integer numberOfSteps, boolean isImportant, " + "The Date to start is today, the date to end is x days from now, type the quantity of days\n"); LocalDateTime createdAT = LocalDateTime.now(); studyPlan.assignSteps(getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), Integer.parseInt(getInput()), Boolean.parseBoolean(getInput()), createdAT, createdAT.plusDays(Long.parseLong(getInput())))); }
```

now use the new assignSteps here

 GitHub Copilot

Refactoring Plan

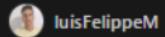
1. Create StepDetails.Builder
2. Collect user inputs
3. Build StepDetails object
4. Call new assignSteps method

```
private void handleSetSteps(StudyPlan studyPlan) {
    handleMethodHeader("(Study Plan Edit)");
    System.out.println("Type the following info: String firstStep, String resetStudyMechanism, String co
        "String seasonalSteps, String basicSteps, String mainObjectiveTitle, String mainGoalTitle, S
        "String mainTask, @NotNull Integer numberOfSteps, boolean isImportant. " +
        "The Date to start is today, the date to end is x days from now, type the quantity of days\n

    LocalDateTime createdAT = LocalDateTime.now();

    StepDetails details = new StepDetails.Builder()
        .firstStep(getInput())
        .resetStudyMechanism(getInput())
        .consistentStep(getInput())
        .seasonalSteps(getInput())
        .basicSteps(getInput())
        .mainObjectiveTitle(getInput())
        .mainGoalTitle(getInput())
        .mainMaterialTopic(getInput())
        .mainTask(getInput())
        .numberOfSteps(Integer.parseInt(getInput()))
        .isImportant(Boolean.parseBoolean(getInput()))
        .startDate(createdAT)
        .endDate(createdAT.plusDays(Long.parseLong(getInput())))
        .build();

    studyPlan.assignSteps(details);
}
```



luisFelipeM

this code smell was fixed, but now there is two new code smells on StepDetails.

he is now a data class, and has a long method. fix them



GitHub Copilot

Refactoring Plan for StepDetails

1. Split into smaller classes:

- Create `StudyStepsInfo` for basic step information
- Create `StudyObjectiveInfo` for objective-related information
- Create `StudyTimeInfo` for time-related data

2. Use composition to reduce complexity

3. Implement value object pattern

```
public record StudyStepsInfo(
    String firstStep,
    String resetStudyMechanism,
    String consistentStep,
    String seasonalSteps,
    String basicSteps
) {}

public record StudyObjectiveInfo(
    String mainObjectiveTitle,
    String mainGoalTitle,
    String mainMaterialTopic,
    String mainTask,
    Integer numberOfSteps,
    boolean isImportant
) {}

public record StudyTimeInfo(
    LocalDateTime startDate,
    LocalDateTime endDate
) {}

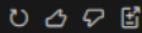
public record StepDetails(
    StudyStepsInfo stepsInfo,
    StudyObjectiveInfo objectiveInfo,
    StudyTimeInfo timeInfo
) {
    public static class Builder {
        private String firstStep;
        private String resetStudyMechanism;
        private String consistentStep;
        private String seasonalSteps;
        private String basicSteps;
        private String mainObjectiveTitle;
        private String mainGoalTitle;
        private String mainMaterialTopic;
        private String mainTask;
        private Integer numberOfSteps;
        private boolean isImportant;
        private LocalDateTime startDate;
        private LocalDateTime endDate;

        public Builder firstStep(String val) { firstStep = val; return this; }
        public Builder resetStudyMechanism(String val) { resetStudyMechanism = val; return this; }
        public Builder consistentStep(String val) { consistentStep = val; return this; }
```

```
public Builder firstStep(String val) { firstStep = val; return this; }
public Builder resetStudyMechanism(String val) { resetStudyMechanism = val; return this; }
public Builder consistentStep(String val) { consistentStep = val; return this; }
public Builder seasonalSteps(String val) { seasonalSteps = val; return this; }
public Builder basicSteps(String val) { basicSteps = val; return this; }
public Builder mainObjectiveTitle(String val) { mainObjectiveTitle = val; return this; }
public Builder mainGoalTitle(String val) { mainGoalTitle = val; return this; }
public Builder mainMaterialTopic(String val) { mainMaterialTopic = val; return this; }
public Builder mainTask(String val) { mainTask = val; return this; }
public Builder numberofSteps(Integer val) { numberofSteps = val; return this; }
public Builder isImportant(boolean val) { isImportant = val; return this; }
public Builder startDate(LocalDateTime val) { startDate = val; return this; }
public Builder endDate(LocalDateTime val) { endDate = val; return this; }

public StepDetails build() {
    return new StepDetails(
        new StudyStepsInfo(firstStep, resetStudyMechanism, consistentStep, seasonalSteps, basicSteps),
        new StudyObjectiveInfo(mainObjectiveTitle, mainGoalTitle, mainMaterialTopic, mainTask, numberofSteps),
        new StudyTimeInfo(startDate, endDate)
    );
}

// Convenience methods to access nested properties
public String getFirstStep() { return stepsInfo.firstStep(); }
public String getResetStudyMechanism() { return stepsInfo.resetStudyMechanism(); }
public String getConsistentStep() { return stepsInfo.consistentStep(); }
public String getSeasonalSteps() { return stepsInfo.seasonalSteps(); }
public String getBasicSteps() { return stepsInfo.basicSteps(); }
public String getMainObjectiveTitle() { return objectiveInfo.mainObjectiveTitle(); }
public String getMainGoalTitle() { return objectiveInfo.mainGoalTitle(); }
public String getMainMaterialTopic() { return objectiveInfo.mainMaterialTopic(); }
public String getMainTask() { return objectiveInfo.mainTask(); }
public Integer getNumberofSteps() { return objectiveInfo.numberofSteps(); }
public boolean isImportant() { return objectiveInfo.isImportant(); }
public LocalDateTime getStartDate() { return timeInfo.startDate(); }
public LocalDateTime getEndDate() { return timeInfo.endDate(); }
}
```



Ask Copilot

@ 🌐 💡

Claude 3.5 Sonnet (Preview) ▶