

 luisFelippeM

fix the Long parameter list code smell using introduce object refactoring technique

```
public class StudyObjective extends Registry{ private String title; private String description; private String topic; private Integer practicedDays; private LocalDateTime startDate; private Double duration; private String objectiveInOneLine; private String objectiveFullDescription;
```

```
public String getTitle() {
    return title;
}

public String getTopic() {
    return topic;
}

public Integer getPracticedDays() {
    return practicedDays;
}

public LocalDateTime getStartDate() {
    return startDate;
}

public Double getDuration() {
    return duration;
}

public String getObjectiveInOneLine() {
    return objectiveInOneLine;
}

public String getObjectiveFullDescription() {
    return objectiveFullDescription;
}

public String getMotivation() {
    return motivation;
}

private String motivation;

@Override
public String toString(){
    return "StudyObjective [title:" + title + ", description:" + description + (topic != null ? ", topic" +
        + (practicedDays != null ? ", practicedDays:" + practicedDays : "") + (duration != null ? ", " +
        + (objectiveInOneLine != null ? ", objective summary:" + objectiveInOneLine : "") + (objecti +
        + (motivation != null ? ", motivation:" + motivation : "") + "]";
}
public StudyObjective(String title, String description) {
```

```
public StudyObjective(String title, String description) {
    this.title = title;
    this.description = description;
    this.name = title;
}

public void handleSetRegistry(Integer id, String name, Integer priority, boolean isActive){
    this.id=id;
    this.name=name;
    this.priority=priority;
    this.isActive=isActive;
}

public void handleSetTextualInfo(String title, String description, String topic, String objectiveInOneLine) {
    this.title=title;
    this.description=description;
    this.topic=topic;
    this.objectiveInOneLine=objectiveInOneLine;
    this.objectiveFullDescription=objectiveFullDescription;
    this.motivation=motivation;
}

public void handleSetTime(Integer practicedDays, int day, int month, int year, Double duration){
    this.practicedDays=practicedDays;
    this.duration=duration;
    this.startDate= LocalDateTime.of(year, month, day, 0, 0);
}

public void handleSetObjective(Integer id, Integer priority, Integer practicedDays, int day, int month,
                               handleSetRegistry(id, name, priority, isActive);
                               handleSetTextualInfo(title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation);
                               handleSetTime(practicedDays, day, month, year, duration);
}

public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String> stringProperties, Double duration) {
    handleSetObjective(intProperties.get(0), intProperties.get(1), intProperties.get(2), intProperties.get(3),
                      stringProperties.get(0), stringProperties.get(1), stringProperties.get(2), stringProperties.get(3));
    return intProperties.get(0);
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
```



GitHub Copilot

Let's refactor this code to reduce the parameter lists using objects. Here's the step-by-step plan:

1. Create data objects for related parameters:
 - o RegistryInfo: id, name, priority, isActive
 - o TextualInfo: title, description, topic, objectives, motivation
 - o TimelInfo: practicedDays, startDate, duration
2. Update the StudyObjective class to use these objects
3. Update the setter methods to accept the new objects

Here's the refactored code:

```
public class RegistryInfo {  
    private Integer id;  
    private String name;  
    private Integer priority;  
    private boolean isActive;  
  
    public RegistryInfo(Integer id, String name, Integer priority, boolean isActive) {  
        this.id = id;  
        this.name = name;  
        this.priority = priority;  
        this.isActive = isActive;  
    }  
  
    // Getters  
    public Integer getId() { return id; }  
    public String getName() { return name; }  
    public Integer getPriority() { return priority; }  
    public boolean isActive() { return isActive; }  
}  
  
public class TextualInfo {  
    private String title;  
    private String description;  
    private String topic;  
    private String objectiveInOneLine;  
    private String objectiveFullDescription;  
    private String motivation;
```

```
public TextualInfo(String title, String description, String topic,
                  String objectiveInOneLine, String objectiveFullDescription,
                  String motivation) {
    this.title = title;
    this.description = description;
    this.topic = topic;
    this.objectiveInOneLine = objectiveInOneLine;
    this.objectiveFullDescription = objectiveFullDescription;
    this.motivation = motivation;
}

// Getters
public String getTitle() { return title; }
public String getDescription() { return description; }
public String getTopic() { return topic; }
public String getObjectiveInOneLine() { return objectiveInOneLine; }
public String getObjectiveFullDescription() { return objectiveFullDescription; }
public String getMotivation() { return motivation; }
}

public class TimeInfo {
    private Integer practicedDays;
    private LocalDateTime startDate;
    private Double duration;

    public TimeInfo(Integer practicedDays, int day, int month, int year, Double duration) {
        this.practicedDays = practicedDays;
        this.startDate = LocalDateTime.of(year, month, day, 0, 0);
        this.duration = duration;
    }

    // Getters
    public Integer getPracticedDays() { return practicedDays; }
    public LocalDateTime getStartDate() { return startDate; }
    public Double getDuration() { return duration; }
}
```

```
public class StudyObjective extends Registry {
    private String title;
    private String description;
    private String topic;
    private Integer practicedDays;
    private LocalDateTime startDate;
    private Double duration;
    private String objectiveInOneLine;
    private String objectiveFullDescription;
    private String motivation;

    // ... existing getters ...

    public void handleSetRegistry(RegistryInfo registryInfo) {
        this.id = registryInfo.getId();
        this.name = registryInfo.getName();
        this.priority = registryInfo.getPriority();
        this.isActive = registryInfo.isActive();
    }

    public void handleSetTextualInfo(TextualInfo textualInfo) {
        this.title = textualInfo.getTitle();
        this.description = textualInfo.getDescription();
        this.topic = textualInfo.getTopic();
        this.objectiveInOneLine = textualInfo.getObjectiveInOneLine();
        this.objectiveFullDescription = textualInfo.getObjectiveFullDescription();
        this.motivation = textualInfo.getMotivation();
    }

    public void handleSetTime(TimeInfo timeInfo) {
        this.practicedDays = timeInfo.getPracticedDays();
        this.startDate = timeInfo.getStartDate();
        this.duration = timeInfo.getDuration();
    }

    public void handleSetObjective(RegistryInfo registryInfo, TextualInfo textualInfo, TimeInfo timeInfo)
        handleSetRegistry(registryInfo);
        handleSetTextualInfo(textualInfo);
        handleSetTime(timeInfo);
    }

    public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String> stringProperties,
                                         Double duration, boolean isActive) {
        RegistryInfo registryInfo = new RegistryInfo(
            intProperties.get(0), stringProperties.get(0), intProperties.get(1), isActive
        );
        TextualInfo textualInfo = new TextualInfo(

```

```

        TextualInfo textualInfo = new TextualInfo(
            stringProperties.get(1), stringProperties.get(2), stringProperties.get(3),
            stringProperties.get(4), stringProperties.get(5), stringProperties.get(6)
        );

        TimeInfo timeInfo = new TimeInfo(
            intProperties.get(2), intProperties.get(3), intProperties.get(4),
            intProperties.get(5), duration
        );

        handleSetObjective(registryInfo, textualInfo, timeInfo);
        return registryInfo.getId();
    }

    // ... rest of the existing code ...
}

```

This refactoring:

- Reduces parameter lists by grouping related data
- Improves code readability and maintenance
- Makes the code more object-oriented
- Maintains existing functionality while being more structured
- Makes it easier to add new properties to each category



 luisFelippeM

```

private void handleSetObjective(StudyObjective objective){ handleMethodHeader("Study Objective Edit"); System.out.println("Type the
following info: Integer id, Integer priority " + "Integer practicedDays, int day, int month, int year, String name, String title, String description, " +
"String topic, String objectiveInOneLine, String objectiveFullDescription, String motivation, " + "Double duration, boolean isActive \n");
objective.handleSetObjective(Integer.parseInt(getInput()),
Integer.parseInt(getInput()),Integer.parseInt(getInput()),Integer.parseInt(getInput()), Integer.parseInt(getInput()), Integer.parseInt(getInput()),
getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), getInput(), Double.parseDouble(getInput()),
Boolean.parseBoolean(getInput())); }

```

adapt this to the new calling of objective.handleSetObjective



GitHub Copilot

Step-by-step plan:

1. Create objects for each parameter group
2. Update method signature to use new objects
3. Group input parameters into respective objects
4. Call updated handleSetObjective method

```
private void handleSetObjective(StudyObjective objective) {  
    handleMethodHeader("(Study Objective Edit)");  
    System.out.println("Enter Registry Info:");  
    System.out.println("id, name, priority, isActive:");  
    RegistryInfo registryInfo = new RegistryInfo(  
        Integer.parseInt(getInput()),  
        getInput(),  
        Integer.parseInt(getInput()),  
        Boolean.parseBoolean(getInput())  
    );  
  
    System.out.println("Enter Textual Info:");  
    System.out.println("title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation");  
    TextualInfo textualInfo = new TextualInfo(  
        getInput(), // title  
        getInput(), // description  
        getInput(), // topic  
        getInput(), // objectiveInOneLine  
        getInput(), // objectiveFullDescription  
        getInput() // motivation  
    );  
  
    System.out.println("Enter Time Info:");  
    System.out.println("practicedDays, day, month, year, duration:");  
    TimeInfo timeInfo = new TimeInfo(  
        Integer.parseInt(getInput()), // practicedDays  
        Integer.parseInt(getInput()), // day  
        Integer.parseInt(getInput()), // month  
        Integer.parseInt(getInput()), // year  
        Double.parseDouble(getInput()) // duration  
    );  
  
    objective.handleSetObjective(registryInfo, textualInfo, timeInfo);  
}
```

 luisFelipeM

```
public class TextualInfo { private String title; private String description; private String topic; private String objectiveInOneLine; private String objectiveFullDescription; private String motivation;

    public TextualInfo(String title, String description, String topic,
                      String objectiveInOneLine, String objectiveFullDescription,
                      String motivation) {
        this.title = title;
        this.description = description;
        this.topic = topic;
        this.objectiveInOneLine = objectiveInOneLine;
        this.objectiveFullDescription = objectiveFullDescription;
        this.motivation = motivation;
    }

    // Getters
    public String getTitle() { return title; }
    public String getDescription() { return description; }
    public String getTopic() { return topic; }
    public String getObjectiveInOneLine() { return objectiveInOneLine; }
    public String getObjectiveFullDescription() { return objectiveFullDescription; }
    public String getMotivation() { return motivation; }
}
```

 GitHub Copilot



 luisFelipeM

```
public class TextualInfo { private String title; private String description; private String topic; private String objectiveInOneLine; private String objectiveFullDescription; private String motivation;

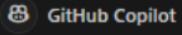
    public TextualInfo(String title, String description, String topic,
                      String objectiveInOneLine, String objectiveFullDescription,
                      String motivation) {
        this.title = title;
        this.description = description;
        this.topic = topic;
        this.objectiveInOneLine = objectiveInOneLine;
        this.objectiveFullDescription = objectiveFullDescription;
        this.motivation = motivation;
    }

    // Getters
    public String getTitle() { return title; }
```

```
public String getDescription() { return description; }
public String getTopic() { return topic; }
public String getObjectiveInOneLine() { return objectiveInOneLine; }
public String getObjectiveFullDescription() { return objectiveFullDescription; }
public String getMotivation() { return motivation; }

}
```

textualInfo is now a data class code smell.



Plan to refactor the TextualInfo data class:

1. Add behavior/methods related to text manipulation
2. Add validation logic
3. Add a builder pattern for flexible object creation
4. Add methods to format and combine the textual information
5. Add `toString` method for proper representation

```
public class TextualInfo {
    private static final int MAX_TITLE_LENGTH = 100;
    private static final int MAX_DESCRIPTION_LENGTH = 500;

    private final String title;
    private final String description;
    private final String topic;
    private final String objectiveInOneLine;
    private final String objectiveFullDescription;
    private final String motivation;

    private TextualInfo(Builder builder) {
        this.title = validateTitle(builder.title);
        this.description = validateDescription(builder.description);
        this.topic = builder.topic;
        this.objectiveInOneLine = builder.objectiveInOneLine;
        this.objectiveFullDescription = builder.objectiveFullDescription;
        this.motivation = builder.motivation;
    }

    private String validateTitle(String title) {
        if (title == null || title.trim().isEmpty()) {
            throw new IllegalArgumentException("Title cannot be empty");
        }
        if (title.length() > MAX_TITLE_LENGTH) {
            throw new IllegalArgumentException("Title exceeds maximum length of " + MAX_TITLE_LENGTH);
        }
        return title.trim();
    }

    private String validateDescription(String description) {
        if (description != null && description.length() > MAX_DESCRIPTION_LENGTH) {
            throw new IllegalArgumentException("Description exceeds maximum length of " + MAX_DESCRIPTION_LENGTH);
        }
        return description;
    }

    public String getFormattedObjective() {
        return String.format("%s: %s", title, objectiveInOneLine);
    }

    public String getDetailedInformation() {
        StringBuilder sb = new StringBuilder()
            .append("Title: ").append(title)
            .append("\nTopic: ").append(topic != null ? topic : "N/A")
            .append("\nDescription: ").append(description != null ? description : "N/A")
            .append("\nObjective: ").append(objectiveInOneLine != null ? objectiveInOneLine : "N/A")
            .append("\nDetailed Description: ").append(objectiveFullDescription != null ? objectiveFullDescription : "N/A")
            .append("\nMotivation: ").append(motivation != null ? motivation : "N/A");
        return sb.toString();
    }
}
```

```
        .append("\nMotivation: ").append(motivation != null ? motivation : "N/A");
    return sb.toString();
}

public boolean containsKeyword(String keyword) {
    return (title != null && title.toLowerCase().contains(keyword.toLowerCase())) ||
           (description != null && description.toLowerCase().contains(keyword.toLowerCase())) ||
           (topic != null && topic.toLowerCase().contains(keyword.toLowerCase()));
}

// Getters remain the same
public String getTitle() { return title; }
public String getDescription() { return description; }
public String getTopic() { return topic; }
public String getObjectiveInOneLine() { return objectiveInOneLine; }
public String getObjectiveFullDescription() { return objectiveFullDescription; }
public String getMotivation() { return motivation; }

public static class Builder {
    private String title;
    private String description;
    private String topic;
    private String objectiveInOneLine;
    private String objectiveFullDescription;
    private String motivation;

    public BuilderWithTitle(String title) {
        this.title = title;
        return this;
    }

    public Builder withDescription(String description) {
        this.description = description;
        return this;
    }

    public Builder withTopic(String topic) {
        this.topic = topic;
        return this;
    }

    public Builder withObjectiveInOneLine(String objectiveInOneLine) {
        this.objectiveInOneLine = objectiveInOneLine;
        return this;
    }

    public Builder withObjectiveFullDescription(String objectiveFullDescription) {
        this.objectiveFullDescription = objectiveFullDescription;
        return this;
    }
}
```

```

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

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

    @Override
    public String toString() {
        return getDetailedInformation();
    }
}

```

 luisFelippeM X

java: constructor TextualInfo in class org.example.studyregistry.TextualInfo cannot be applied to given types; required:
org.example.studyregistry.TextualInfo.Builder found:
java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String reason: actual and formal argument lists differ in length

```

private void handleSetObjective(StudyObjective objective) { handleMethodHeader("Study Objective Edit"); System.out.println("Enter Registry Info"); System.out.println("id, name, priority, isActive"); RegistryInfo registryInfo = new RegistryInfo( Integer.parseInt(getInput()), getInput(), Integer.parseInt(getInput()), Boolean.parseBoolean(getInput()) );

System.out.println("Enter Textual Info:");
System.out.println("title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation");
TextualInfo textualInfo = new TextualInfo(
    getInput(), // title
    getInput(), // description
    getInput(), // topic
    getInput(), // objectiveInOneLine
    getInput(), // objectiveFullDescription
    getInput() // motivation
);

System.out.println("Enter Time Info:");
System.out.println("practicedDays, day, month, year, duration:");
TimeInfo timeInfo = new TimeInfo(
    Integer.parseInt(getInput()), // practicedDays
    Integer.parseInt(getInput()), // day
    Integer.parseInt(getInput()), // month
    Integer.parseInt(getInput()), // year
    Double.parseDouble(getInput()) // duration
);

objective.handleSetObjective(registryInfo, textualInfo, timeInfo);
}

```



GitHub Copilot

Plan

1. Update the TextualInfo creation to use the Builder pattern
2. Modify handleSetObjective method to use builder pattern for TextualInfo
3. Keep other parts of the code unchanged

```
private void handleSetObjective(StudyObjective objective) {
    handleMethodHeader("(Study Objective Edit)");
    System.out.println("Enter Registry Info:");
    System.out.println("id, name, priority, isActive:");
    RegistryInfo registryInfo = new RegistryInfo(
        Integer.parseInt(getInput()),
        getInput(),
        Integer.parseInt(getInput()),
        Boolean.parseBoolean(getInput())
    );

    System.out.println("Enter Textual Info:");
    System.out.println("title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation");
    TextualInfo textualInfo = new TextualInfo.Builder()
        .withTitle(getInput())
        .withDescription(getInput())
        .withTopic(getInput())
        .withObjectiveInOneLine(getInput())
        .withObjectiveFullDescription(getInput())
        .withMotivation(getInput())
        .build();

    System.out.println("Enter Time Info:");
    System.out.println("practicedDays, day, month, year, duration:");
    TimeInfo timeInfo = new TimeInfo(
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Double.parseDouble(getInput())
    );

    objective.handleSetObjective(registryInfo, textualInfo, timeInfo);
}
```

 luisFelippeM

```
public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String> stringProperties, Double duration, boolean isActive) {  
    RegistryInfo registryInfo = new RegistryInfo( intProperties.get(0), stringProperties.get(0), intProperties.get(1), isActive );  
  
    TextualInfo textualInfo = new TextualInfo(  
        stringProperties.get(1), stringProperties.get(2), stringProperties.get(3),  
        stringProperties.get(4), stringProperties.get(5), stringProperties.get(6)  
    );  
  
    TimeInfo timeInfo = new TimeInfo(  
        intProperties.get(2), intProperties.get(3), intProperties.get(4),  
        intProperties.get(5), duration  
    );  
  
    handleSetObjective(registryInfo, textualInfo, timeInfo);  
    return registryInfo.getId();  
}
```

'TextualInfo(org.example.studyregistry.TextualInfo.Builder)' has private access in 'org.example.studyreg'

 GitHub Copilot

Plan to fix TextualInfo adapter:

1. Update `handleSetObjectiveAdapter` to use TextualInfo Builder pattern
2. Keep the same logic for parameter distribution
3. Maintain the same return value
4. Use builder methods for TextualInfo creation

```
public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String> stringProperties,  
    Double duration, boolean isActive) {  
    RegistryInfo registryInfo = new RegistryInfo(  
        intProperties.get(0), stringProperties.get(0), intProperties.get(1), isActive  
    );  
  
    TextualInfo textualInfo = new TextualInfo.Builder()  
        .WithTitle(stringProperties.get(1))  
        .WithDescription(stringProperties.get(2))  
        .WithTopic(stringProperties.get(3))  
        .WithObjectiveInOneLine(stringProperties.get(4))  
        .WithObjectiveFullDescription(stringProperties.get(5))  
        .WithMotivation(stringProperties.get(6))  
        .Build();  
  
    TimeInfo timeInfo = new TimeInfo(  
        intProperties.get(2), intProperties.get(3), intProperties.get(4),  
        intProperties.get(5), duration  
    );  
  
    handleSetObjective(registryInfo, textualInfo, timeInfo);  
    return registryInfo.getId();  
}
```

 luisFelippeM

X

```
private void handleSetObjective(StudyObjective objective) { handleMethodHeader("(Study Objective Edit)"); System.out.println("Enter Registry Info:"); System.out.println("id, name, priority, isActive:"); RegistryInfo registryInfo = new RegistryInfo( Integer.parseInt(getInput()), getInput(), Integer.parseInt(getInput()), Boolean.parseBoolean(getInput()) );  
  
    System.out.println("Enter Textual Info:");  
    System.out.println("title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation:");  
    TextualInfo textualInfo = new TextualInfo.Builder()  
        .withTitle(getInput())  
        .withDescription(getInput())  
        .withTopic(getInput())  
        .withObjectiveInOneLine(getInput())  
        .withObjectiveFullDescription(getInput())  
        .withMotivation(getInput())  
        .build();  
  
    System.out.println("Enter Time Info:");  
    System.out.println("practicedDays, day, month, year, duration:");  
    TimeInfo timeInfo = new TimeInfo(  
        Integer.parseInt(getInput()),  
        Integer.parseInt(getInput()),  
        Integer.parseInt(getInput()),  
        Integer.parseInt(getInput()),  
        Double.parseDouble(getInput())  
    );  
  
    objective.handleSetObjective(registryInfo, textualInfo, timeInfo);  
}  
  
this is now a long method code smell. fix it
```

 GitHub Copilot

Plan:

1. Extract method for Registry Info input handling
2. Extract method for Textual Info input handling
3. Extract method for Time Info input handling
4. Create a main method that orchestrates these calls
5. Keep error handling in each method

```
private void handleSetObjective(StudyObjective objective) {
    handleMethodHeader("({Study Objective Edit})");

    RegistryInfo registryInfo = getRegistryInfo();
    TextualInfo textualInfo = getTextualInfo();
    TimeInfo timeInfo = getTimeInfo();

    objective.handleSetObjective(registryInfo, textualInfo, timeInfo);
}

private RegistryInfo getRegistryInfo() {
    System.out.println("Enter Registry Info:");
    System.out.println("id, name, priority, isActive:");
    return new RegistryInfo(
        Integer.parseInt(getInput()),
        getInput(),
        Integer.parseInt(getInput()),
        Boolean.parseBoolean(getInput())
    );
}

private TextualInfo getTextualInfo() {
    System.out.println("Enter Textual Info:");
    System.out.println("title, description, topic, objectiveInOneLine, objectiveFullDescription, motivation");
    return new TextualInfo.Builder()
        .WithTitle(getInput())
        .WithDescription(getInput())
        .WithTopic(getInput())
        .WithObjectiveInOneLine(getInput())
        .WithObjectiveFullDescription(getInput())
        .WithMotivation(getInput())
        .Build();
}

private TimeInfo getTimeInfo() {
    System.out.println("Enter Time Info:");
    System.out.println("practicedDays, day, month, year, duration:");
    return new TimeInfo(
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Integer.parseInt(getInput()),
        Double.parseDouble(getInput())
    );
}
```