StudyObjective

```
public void handleSetObjective(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){
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, boolean isActive){
handleSetObjective(intProperties.get(0), intProperties.get(1), intProperties.get(2),
intProperties.get(3), intProperties.get(4), intProperties.get(5),
stringProperties.get(0), stringProperties.get(1), stringProperties.get(2),
stringProperties.get(3), stringProperties.get(4), stringProperties.get(5),
stringProperties.get(6), duration, isActive);
return intProperties.get(0);
}
```

fix the Long parameter list code smell using the "Introduce Parameter Object." refactoring technique in handleSetObjective

Java

```java
public class Objective {

    private Integer id;
    private Integer priority;
    private Integer practicedDays;
    private int day;
    private int month;
    private int year;
    private String name;
    private String title;
    private String description;
    private String topic;
    private String objectiveInOneLine;
    private String objectiveFullDescription;
    private String motivation;
    private Double duration;
    private boolean isActive;

    // Constructor
    public Objective(Integer id, Integer priority, Integer practicedDays, int day,
        this.id = id;
        this.priority = priority;
        this.practicedDays = practicedDays;
        this.day = day;
        this.month = month;
        this.year = year;
        this.name = name;
        this.title = title;
        this.description = description;
        this.topic = topic;
        this.objectiveInOneLine = objectiveInOneLine;
        this.objectiveFullDescription = objectiveFullDescription;
        this.motivation = motivation;
        this.duration = duration;
        this.isActive = isActive;
    }

    // Getters and Setters (omitted for brevity)

}

public class ObjectiveService {

    public void handleSetObjective(Objective objective) {
        handleSetRegistry(objective.getId(), objective.getName(), objective.getPri
        handleSetTextualInfo(objective.getTitle(), objective.getDescription(), obj
        handleSetTime(objective.getPracticedDays(), objective.getDay(), objective.
    }

    public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String>
        Objective objective = new Objective(
                intProperties.get(0),
                intProperties.get(1),
                intProperties.get(2),
                intProperties.get(3),
                intProperties.get(4),
                intProperties.get(5),
                stringProperties.get(0),
                stringProperties.get(1),
                stringProperties.get(2),
                stringProperties.get(3),
                stringProperties.get(4),
                stringProperties.get(5),
                stringProperties.get(6),
                duration,
                isActive
        );
        handleSetObjective(objective);
        return intProperties.get(0);
    }

    // Other methods (handleSetRegistry, handleSetTextualInfo, handleSetTime)
}
```

Explanation:

Explanation:

1. **Create an Objective Class:**
   - A new `Objective` class is created to encapsulate all the attributes related to an objective.
   - This class holds fields for each parameter in the original `handleSetObjective` method.
   - A constructor is provided to initialize an `Objective` instance with the given values.
2. **Modify handleSetObjective:**
   - The `handleSetObjective` method is modified to accept an `Objective` object as its sole parameter.
   - This significantly reduces the number of parameters passed to the method.
3. **Modify handleSetObjectiveAdapter:**
   - The `handleSetObjectiveAdapter` method is updated to create an `Objective` instance using the provided lists and other parameters.
   - Then, it calls the modified `handleSetObjective` method with the created `Objective` object.

Benefits:

- **Reduced Parameter List:** The `handleSetObjective` method now has a single parameter, making it more readable and maintainable.
- **Improved Code Organization:** The `Objective` class encapsulates all the relevant data related to an objective, improving code organization and readability.
- **Enhanced Reusability:** The `Objective` object can be used in other parts of the application, potentially improving code reusability.
- **Easier to Modify:** Adding or removing parameters in the future becomes easier as it only requires modifying the `Objective` class.

This refactoring addresses the "Long Parameter List" code smell by introducing a dedicated object to represent the objective, resulting in cleaner and more maintainable code.

make obejctive as a record

Java

```java
public record Objective(
        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
) {
}

public class ObjectiveService {

    public void handleSetObjective(Objective objective) {
        handleSetRegistry(objective.id(), objective.name(), objective.priority(),
        handleSetTextualInfo(objective.title(), objective.description(), objective
        handleSetTime(objective.practicedDays(), objective.day(), objective.month(
    }

    public int handleSetObjectiveAdapter(List<Integer> intProperties, List<String>
        Objective objective = new Objective(
                intProperties.get(0),
                intProperties.get(1),
                intProperties.get(2),
                intProperties.get(3),
                intProperties.get(4),
                intProperties.get(5),
                stringProperties.get(0),
                stringProperties.get(1),
                stringProperties.get(2),
                stringProperties.get(3),
                stringProperties.get(4),
                stringProperties.get(5),
                stringProperties.get(6),
                duration,
                isActive
        );
        handleSetObjective(objective);
        return intProperties.get(0);
    }

    // Other methods (handleSetRegistry, handleSetTextualInfo, handleSetTime)
}
```

Explanation:

- **Record Class:**
  - The `Objective` class is now defined as a `record`.
  - Records are a concise way to define immutable classes in Java.
  - They automatically generate a constructor, getters, `equals`, `hashCode`, and `toString` methods.
- **Benefits of Using Records:**
  - **Conciseness:** Records provide a more concise syntax compared to traditional classes, reducing boilerplate code.
  - **Immutability:** Records are inherently immutable, which can improve the safety and predictability of your code.
  - **Readability:** The compact syntax of records can make your code more readable and easier to understand.

This version leverages the features of Java records to further simplify the `Objective` class and enhance code maintainability.

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()));
}

refactor this related method without change the parsing to use the new method signature

Java

```java
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, Str
            "String topic, String objectiveInOneLine, String objectiveFullDescript
            "Double duration, boolean isActive  \n");

    objective.handleSetObjective(
            new Objective(
                    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())
            )
    );
}
```

Explanation:

- **Create an Objective Instance:**
  - Inside the `handleSetObjective` method, an instance of the `Objective` class is created using the `new Objective(...)` constructor.
  - This constructor takes all the necessary parameters to create an `Objective` object.
- **Pass Objective to handleSetObjective:**
  - The created `Objective` instance is then passed as the argument to the `objective.handleSetObjective()` method.
  - This utilizes the new method signature that accepts an `Objective` object.

Key Changes:

- The `objective.handleSetObjective()` method now receives an `Objective` object instead of individual parameters.
- The code within `handleSetObjective` is simplified by creating an `Objective` instance and passing it to the method.

This refactored code adheres to the new method signature while maintaining the original parsing logic.