

## StudyTaskManager



```
public void setUpWeek(String planName, String objectiveTitle, String objectiveDescription,
String materialTopic,
String materialFormat, String goal, String reminderTitle, String reminderDescription,
String mainTaskTitle, String mainHabit, String mainCardStudy){
    this.weekResponsibilities = new ArrayList<>();
    this.weekResponsibilities.addAll(Arrays.asList(planName, objectiveTitle,
objectiveDescription, materialTopic, materialFormat, goal, reminderTitle,
reminderDescription, mainTaskTitle, mainHabit, mainCardStudy));
}
```

```
public void handleSetUpWeek(List<String> stringProperties){
    setUpWeek(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), stringProperties.get(9), stringProperties.get(10));
}
```

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

Java



```
public class WeekSetup {

    private static class WeekSetupData {
        private String planName;
        private String objectiveTitle;
        private String objectiveDescription;
        private String materialTopic;
        private String materialFormat;
        private String goal;
        private String reminderTitle;
        private String reminderDescription;
        private String mainTaskTitle;
        private String mainHabit;
        private String mainCardStudy;

        // Constructor
        public WeekSetupData(String planName, String objectiveTitle, String objectiveDescription,
                               String materialFormat, String goal, String reminderTitle,
                               String mainTaskTitle, String mainHabit, String mainCardStudy) {
            this.planName = planName;
            this.objectiveTitle = objectiveTitle;
            this.objectiveDescription = objectiveDescription;
            this.materialTopic = materialTopic;
            this.materialFormat = materialFormat;
            this.goal = goal;
            this.reminderTitle = reminderTitle;
            this.reminderDescription = reminderDescription;
            this.mainTaskTitle = mainTaskTitle;
            this.mainHabit = mainHabit;
            this.mainCardStudy = mainCardStudy;
        }
    }

    private List<String> weekResponsibilities;

    public void setUpWeek(WeekSetupData weekSetupData) {
        this.weekResponsibilities = new ArrayList<>();
        this.weekResponsibilities.addAll(Arrays.asList(
            weekSetupData.planName,
            weekSetupData.objectiveTitle,
            weekSetupData.objectiveDescription,
            weekSetupData.materialTopic,
            weekSetupData.materialFormat,
            weekSetupData.goal,
            weekSetupData.reminderTitle,
            weekSetupData.reminderDescription,
            weekSetupData.mainTaskTitle,
            weekSetupData.mainHabit,
            weekSetupData.mainCardStudy
        ));
    }

    public void handleSetUpWeek(List<String> stringProperties) {
        WeekSetupData weekSetupData = new WeekSetupData(
            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),
            stringProperties.get(9),
            stringProperties.get(10)
        );
        setUpWeek(weekSetupData);
    }
}
```

#### Explanation:

##### 1. Create a Parameter Object:

- A new inner class `WeekSetupData` is created to hold all the parameters of the `setUpWeek` method.
- This class encapsulates the data, making the code more readable and maintainable.

##### 2. Modify `setUpWeek` Method:

- The `setUpWeek` method is modified to accept an instance of `WeekSetupData` instead of individual parameters.
- The method now accesses the required data from the `weekSetupData` object.

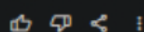
##### 3. Modify `handleSetUpWeek` Method:

- The `handleSetUpWeek` method is modified to create an instance of `WeekSetupData` using the provided `stringProperties` list.
- It then calls the `setUpWeek` method with the created `weekSetupData` object.

#### Benefits:

- **Reduced Parameter List:** The original `setUpWeek` method had a long parameter list, which can be difficult to read, understand, and maintain. By introducing the `WeekSetupData` object, the parameter list is reduced to a single object, making the code more concise and easier to follow.
- **Improved Readability:** The code becomes more readable as the meaning of the parameters is now encapsulated within the `WeekSetupData` object.
- **Enhanced Maintainability:** If you need to add or remove parameters, you only need to modify the `WeekSetupData` class, which is a localized change.
- **Increased Flexibility:** The `WeekSetupData` object can be reused in other parts of the code if needed.

This refactoring improves the code's readability, maintainability, and overall quality.



since a dataclass in this context is considered a code smell, use a record instead



Java



```
public class WeekSetup {

    record WeekSetupData(
        String planName,
        String objectiveTitle,
        String objectiveDescription,
        String materialTopic,
        String materialFormat,
        String goal,
        String reminderTitle,
        String reminderDescription,
        String mainTaskTitle,
        String mainHabit,
        String mainCardStudy
    ) {}

    private List<String> weekResponsibilities;

    public void setUpWeek(WeekSetupData weekSetupData) {
        this.weekResponsibilities = new ArrayList<>();
        this.weekResponsibilities.addAll(Arrays.asList(
            weekSetupData.planName(),
            weekSetupData.objectiveTitle(),
            weekSetupData.objectiveDescription(),
            weekSetupData.materialTopic(),
            weekSetupData.materialFormat(),
            weekSetupData.goal(),
            weekSetupData.reminderTitle(),
            weekSetupData.reminderDescription(),
            weekSetupData.mainTaskTitle(),
            weekSetupData.mainHabit(),
            weekSetupData.mainCardStudy()
        ));
    }

    public void handleSetUpWeek(List<String> stringProperties) {
        WeekSetupData weekSetupData = new WeekSetupData(
            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),
            stringProperties.get(9),
            stringProperties.get(10)
        );
        setUpWeek(weekSetupData);
    }
}
```

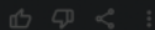
#### Explanation:

- **Record Declaration:** The `WeekSetupData` class is declared as a `record`. Records are a special kind of class introduced in Java 14 that are concise and immutable. They are designed for cases where a class primarily holds data.
- **Automatic Generation:** The Java compiler automatically generates several methods for records, including:
  - `toString()`
  - `hashCode()`
  - `equals()`
  - Accessor methods (e.g., `planName()`, `objectiveTitle()`, etc.)
- **Immutability:** Records are immutable by default, meaning their fields cannot be modified after creation. This ensures data integrity and can help prevent unintended side effects.

#### Key Advantages of Using a Record:

- **Conciseness:** Records provide a more concise syntax compared to traditional classes, reducing boilerplate code.
- **Immutability:** Immutability enhances the safety and predictability of your code.
- **Automatic Generation:** The compiler automatically generates essential methods, saving you time and effort.

By using a record instead of a traditional class or a `dataclass`, you achieve a more concise and potentially more efficient solution while maintaining the benefits of encapsulation and data organization.





```
private void getWeekInfo(){
    System.out.println("(Study Task Manager Week Set Up) Type the following info: String
    planName, String objectiveTitle, " +
    "String objectiveDescription, String materialTopic, String materialFormat, String goal, String
    reminderTitle, " +
    "String reminderDescription, String mainTaskTitle, String mainHabit, String
    mainCardStudy");
    studyTaskManager.setUpWeek(getInput(), getInput(), getInput(), getInput(), getInput(),
    getInput(), getInput(), getInput(),
    getInput(), getInput(), getInput());
}
```

nice, now refactor this method without changing the message and the parsing logic, just adapt it to the new setUpWeek signature



Java



```
private void getWeekInfo() {
    System.out.println("(Study Task Manager Week Set Up) Type the following info: {
        \"String objectiveDescription, String materialTopic, String materialForm
        \"String reminderDescription, String mainTaskTitle, String mainHabit, S

    // Get input for each field
    String planName = getInput();
    String objectiveTitle = getInput();
    String objectiveDescription = getInput();
    String materialTopic = getInput();
    String materialFormat = getInput();
    String goal = getInput();
    String reminderTitle = getInput();
    String reminderDescription = getInput();
    String mainTaskTitle = getInput();
    String mainHabit = getInput();
    String mainCardStudy = getInput();

    // Create WeekSetupData object
    WeekSetupData weekSetupData = new WeekSetupData(
        planName, objectiveTitle, objectiveDescription,
        materialTopic, materialFormat, goal,
        reminderTitle, reminderDescription,
        mainTaskTitle, mainHabit, mainCardStudy
    );

    // Call setUpWeek with the WeekSetupData object
    studyTaskManager.setUpWeek(weekSetupData);
}
```

#### Explanation:

1. **Get Input:** The original code remains unchanged for getting the input for each field using the `getInput()` method.
2. **Create WeekSetupData Object:**
  - A new `WeekSetupData` object is created using the collected input for each field.
3. **Call setUpWeek:**
  - The `studyTaskManager.setUpWeek()` method is called, passing the newly created `weekSetupData` object as the argument.

This refactored version adapts the `getWeekInfo()` method to the new `setUpWeek` signature while preserving the original message and input parsing logic.

make the method with less lines



Java



```
private void getWeekInfo() {
    System.out.println("(Study Task Manager Week Set Up) Type the following info: ");
    "String objectiveDescription, String materialTopic, String materialForm
    "String reminderDescription, String mainTaskTitle, String mainHabit, S

    studyTaskManager.setUpWeek(
        new WeekSetupData(
            getInput(), getInput(), getInput(),
            getInput(), getInput(), getInput(),
            getInput(), getInput(), getInput(),
            getInput(), getInput()
        )
    );
}
```

#### Explanation:

- **Concise Object Creation:** The creation of the `WeekSetupData` object is now done within a single line, improving readability and reducing the number of lines of code.

This version maintains the same functionality as the previous one while being more concise.

