



luisFelippeM

X

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

```
public class AudioReference extends Reference { public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH } private AudioQuality audioQuality;
```

```
    public AudioReference(AudioQuality quality) {
        this.audioQuality = quality;
    }

    public void editBasic(String title, String description, String link) {
        updateMetadata(title, description, link);
    }

    public void editAudioAdapter(List<String> properties, List<Integer> intProperties,
                                AudioQuality audioQuality, boolean isDownloadable) {
        editAudio(audioQuality, isDownloadable,
                  properties.get(0), // title
                  properties.get(1), // description
                  properties.get(2), // link
                  properties.get(3), // accessRights
                  properties.get(4), // license
                  properties.get(5), // language
                  intProperties.get(0), // rating
                  intProperties.get(1), // viewCount
                  intProperties.get(2) // shareCount
        );
    }

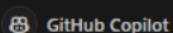
    public void editAudio(AudioQuality audioQuality, boolean isDownloadable,
                        String title, String description, String link,
                        String accessRights, String license, String language,
                        int rating, int viewCount, int shareCount) {
        editBasic(title, description, link);
        updateAccessControl(accessRights, license, isDownloadable);
        setStats(rating, language, viewCount, shareCount);
        this.audioQuality = audioQuality;
    }

    public static AudioQuality audioQualityAdapter(String quality) {
        return switch (quality.toLowerCase()) {
            case "low" -> AudioQuality.LOW;
            case "medium" -> AudioQuality.MEDIUM;
            case "high" -> AudioQuality.HIGH;
            case "very_high" -> AudioQuality.VERY_HIGH;
            default -> throw new IllegalArgumentException("Invalid audio quality");
        };
    }
}
```

```
    public AudioQuality getAudioQuality() {
        return audioQuality;
    }
```

} and this is a method of other class that calls the before mentioned editAudio:

```
private void editAudio(AudioReference audioReference) handleMethodHeader("(Audio Edit)"); System.out.println("Type the following info:  
AudioReference. AudioQuality audioQuality, boolean isDownloadable, " + "String title, String description, String accessRights, String  
license, String language, int rating, " + "int viewCount, int shareCount \n"); AudioReference.AudioQuality quality  
=AudioReference.audioQualityAdapter(getInput()); audioReference.editAudio(quality, Boolean.parseBoolean(getInput()), getInput(), getInput(),  
getInput(), getInput(), getInput(), Integer.parseInt(getInput()), Integer.parseInt(getInput()), Integer.parseInt(getInput())); }
```



GitHub Copilot

Step-by-step plan:

1. Create an AudioReferenceBuilder class
2. Add all necessary fields to builder
3. Create builder methods for each parameter
4. Add build method
5. Modify AudioReference to use builder
6. Update the client code to use builder pattern

Implementation:

```
public class AudioReferenceBuilder {
    private AudioReference.AudioQuality audioQuality;
    private boolean isDownloadable;
    private String title;
    private String description;
    private String link;
    private String accessRights;
    private String license;
    private String language;
    private int rating;
    private int viewCount;
    private int shareCount;

    public AudioReferenceBuilder withAudioQuality(AudioReference.AudioQuality audioQuality) {
        this.audioQuality = audioQuality;
        return this;
    }

    public AudioReferenceBuilder withDownloadable(boolean isDownloadable) {
        this.isDownloadable = isDownloadable;
        return this;
    }

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

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

    public AudioReferenceBuilder withLink(String link) {
        this.link = link;
        return this;
    }

    public AudioReferenceBuilder withAccessRights(String accessRights) {
        this.accessRights = accessRights;
        return this;
    }

    public AudioReferenceBuilder withLicense(String license) {
        this.license = license;
        return this;
    }
}
```

```
public AudioReferenceBuilder withLanguage(String language) {
    this.language = language;
    return this;
}

public AudioReferenceBuilder withRating(int rating) {
    this.rating = rating;
    return this;
}

public AudioReferenceBuilder withViewCount(int viewCount) {
    this.viewCount = viewCount;
    return this;
}

public AudioReferenceBuilder withShareCount(int shareCount) {
    this.shareCount = shareCount;
    return this;
}

public void buildInto(AudioReference audioReference) {
    audioReference.editAudio(audioQuality, isDownloadable, title, description,
        link, accessRights, license, language, rating, viewCount, shareCount);
}
}

public class AudioReference extends Reference {
    // ...existing code...

    public static AudioReferenceBuilder builder() {
        return new AudioReferenceBuilder();
    }

    // Keep existing methods but mark editAudio as package-private
    void editAudio(AudioQuality audioQuality, boolean isDownloadable,
        String title, String description, String link,
        String accessRights, String license, String language,
        int rating, int viewCount, int shareCount) {
        // ...existing implementation...
    }
}
```

```
private void editAudio(AudioReference audioReference) {
    handleMethodHeader("(Audio Edit)");
    System.out.println("Type the following info: AudioQuality, isDownloadable, title, description, link,
        "accessRights, license, language, rating, viewCount, shareCount\n");

    AudioReferenceBuilder builder = AudioReference.builder()
        .withAudioQuality(AudioReference.audioQualityAdapter(getInput()))
        .withDownloadable(Boolean.parseBoolean(getInput()))
        .WithTitle(getInput())
        .withDescription(getInput())
        .withLink(getInput())
        .withAccessRights(getInput())
        .withLicense(getInput())
        .withLanguage(getInput())
        .withRating(Integer.parseInt(getInput()))
        .withViewCount(Integer.parseInt(getInput()))
        .withShareCount(Integer.parseInt(getInput()));

    builder.buildInto(audioReference);
}
```



luisFelipeM

X

the AudioReference class still has the code smell

GitHub Copilot

Plan for Builder Pattern Implementation:

1. Add a static nested Builder class inside AudioReference
2. Move all edit parameters to Builder class
3. Create builder methods for each parameter
4. Add build method in Builder
5. Make AudioReference constructor private
6. Create static factory method to get Builder instance

Code Changes:

```
public class AudioReference extends Reference {
    public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH }
    private AudioQuality audioQuality;

    private AudioReference(AudioQuality quality) {
        this.audioQuality = quality;
    }

    public static Builder builder(AudioQuality quality) {
        return new Builder(quality);
    }

    public static class Builder {
        private final AudioQuality audioQuality;
        private boolean isDownloadable;
        private String title;
        private String description;
        private String link;
        private String accessRights;
        private String license;
        private String language;
        private int rating;
        private int viewCount;
        private int shareCount;

        private Builder(AudioQuality quality) {
            this.audioQuality = quality;
        }

        public Builder downloadable(boolean isDownloadable) {
            this.isDownloadable = isDownloadable;
            return this;
        }

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

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

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

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

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

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

public Builder rating(int rating) {
    this.rating = rating;
    return this;
}

public Builder viewCount(int viewCount) {
    this.viewCount = viewCount;
    return this;
}

public Builder shareCount(int shareCount) {
    this.shareCount = shareCount;
    return this;
}

public AudioReference build() {
    AudioReference reference = new AudioReference(audioQuality);
    reference.editBasic(title, description, link);
    reference.updateAccessControl(accessRights, license, isDownloadable);
    reference.setStats(rating, language, viewCount, shareCount);
    return reference;
}
}

public static AudioQuality audioQualityAdapter(String quality) {
    // ...existing code...
}

public AudioQuality getAudioQuality() {
    return audioQuality;
}
}
```

```
private void editAudio(AudioReference audioReference) {
    handleMethodHeader("(Audio Edit)");
    System.out.println("Type the following info: AudioQuality, isDownloadable, title, description, link,
        "accessRights, license, language, rating, viewCount, shareCount\n");

    AudioReference newReference = AudioReference.builder(AudioReference.audioQualityAdapter(getInput()))
        .downloadable(Boolean.parseBoolean(getInput()))
        .title(getInput())
        .description(getInput())
        .link(getInput())
        .accessRights(getInput())
        .license(getInput())
        .language(getInput())
        .rating(Integer.parseInt(getInput()))
        .viewCount(Integer.parseInt(getInput()))
        .shareCount(Integer.parseInt(getInput()))
        .build();
}
```

Similar code found with 1 license type - [View matches](#)



luisFelippeM

java: AudioReference(org.example.studymaterial.AudioReference.AudioQuality) has private access in
org.example.studymaterial.AudioReference

GitHub Copilot

Plan:

1. Fix accessibility levels
2. Make sure builder creates new instances
3. Add methods to update existing AudioReference objects
4. Remove old edit methods
5. Preserve necessary public methods

Code Changes:

```
public class AudioReference extends Reference {
    public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH }
    private AudioQuality audioQuality;

    // Make constructor public again since we need to create instances
    public AudioReference(AudioQuality quality) {
        this.audioQuality = quality;
    }

    public static Builder builder() {
        return new Builder();
    }

    public static class Builder {
        private AudioQuality audioQuality;
        private boolean isDownloadable;
        private String title;
        private String description;
        private String link;
        private String accessRights;
        private String license;
        private String language;
        private int rating;
        private int viewCount;
        private int shareCount;

        public Builder audioQuality(AudioQuality quality) {
            this.audioQuality = quality;
            return this;
        }

        // ... other builder setters remain the same ...

        public void updateExisting(AudioReference reference) {
            reference.editBasic(title, description, link);
            reference.updateAccessControl(accessRights, license, isDownloadable);
            reference.setStats(rating, language, viewCount, shareCount);
            reference.audioQuality = this.audioQuality;
        }

        public AudioReference build() {
            AudioReference reference = new AudioReference(audioQuality);
            updateExisting(reference);
            return reference;
        }
    }
}

// Keep only necessary public methods
```

```
// Keep only necessary public methods
public void editBasic(String title, String description, String link) {
    updateMetadata(title, description, link);
}

public AudioQuality getAudioQuality() {
    return audioQuality;
}

public static AudioQuality audioQualityAdapter(String quality) {
    // ... existing code ...
}
}
```

```
private void editAudio(AudioReference audioReference) {
    handleMethodHeader("(Audio Edit)");
    System.out.println("Type the following info: AudioQuality, isDownloadable, title, description, link,
        "accessRights, license, language, rating, viewCount, shareCount\n");

    AudioReference.Builder builder = AudioReference.builder()
        .audioQuality(AudioReference.audioQualityAdapter(getInput()))
        .downloadable(Boolean.parseBoolean(getInput()))
        .title(getInput())
        .description(getInput())
        .link(getInput())
        .accessRights(getInput())
        .license(getInput())
        .language(getInput())
        .rating(Integer.parseInt(getInput()))
        .viewCount(Integer.parseInt(getInput()))
        .shareCount(Integer.parseInt(getInput()));

    builder.updateExisting(audioReference);
}
```



 luisFelippeM

```
public class AudioReference extends Reference { public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH } private AudioQuality audioQuality;

// Make constructor public again since we need to create instances
public AudioReference(AudioQuality quality) {
    this.audioQuality = quality;
}

public static Builder builder() {
    return new Builder();
}

public static class Builder {
    private AudioQuality audioQuality;
    private boolean isDownloadable;
    private String title;
    private String description;
    private String link;
    private String accessRights;
    private String license;
    private String language;
    private int rating;
    private int viewCount;
    private int shareCount;

    public Builder audioQuality(AudioQuality quality) {
        this.audioQuality = quality;
        return this;
    }

    public Builder downloadable(boolean isDownloadable) {
        this.isDownloadable = isDownloadable;
        return this;
    }

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

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

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

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

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

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

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

public Builder rating(int rating) {
    this.rating = rating;
    return this;
}

public Builder viewCount(int viewCount) {
    this.viewCount = viewCount;
    return this;
}

public Builder shareCount(int shareCount) {
    this.shareCount = shareCount;
    return this;
}

public void updateExisting(AudioReference reference) {
    reference.editBasic(title, description, link);
    reference.updateAccessControl(accessRights, license, isDownloadable);
    reference.setStats(rating, language, viewCount, shareCount);
    reference.audioQuality = this.audioQuality;
}

public AudioReference build() {
    AudioReference reference = new AudioReference(audioQuality);
    updateExisting(reference);
    return reference;
}
}
```

```
public void editBasic(String title, String description, String link) {
    updateMetadata(title, description, link);
}

public void editAudioAdapter(List<String> properties, List<Integer> intProperties,
                            AudioQuality audioQuality, boolean isDownloadable) {
    editAudio(audioQuality, isDownloadable,
              properties.get(0), // title
              properties.get(1), // description
              properties.get(2), // link
              properties.get(3), // accessRights
              properties.get(4), // license
              properties.get(5), // language
              intProperties.get(0), // rating
              intProperties.get(1), // viewCount
              intProperties.get(2) // shareCount
    );
}

public static AudioQuality audioQualityAdapter(String quality) {
    return switch (quality.toLowerCase()) {
        case "low" -> AudioQuality.LOW;
        case "medium" -> AudioQuality.MEDIUM;
        case "high" -> AudioQuality.HIGH;
        case "very_high" -> AudioQuality.VERY_HIGH;
        default -> throw new IllegalArgumentException("Invalid audio quality");
    };
}

public AudioQuality getAudioQuality() {
    return audioQuality;
}
```

```
}
```

java: cannot find symbol symbol: method
editAudio(org.example.studymaterial.AudioReference.AudioQuality,boolean,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.String,java.lang.Integer,java.lang.Integer,java.lang.Integer) location: class org.example.studymaterial.AudioReference



GitHub Copilot

Step-by-Step Plan:

1. Remove the old `editAudio` method
2. Move logic from `editAudioAdapter` to Builder
3. Make Builder handle both direct and adapter-style construction
4. Clean up unnecessary methods
5. Ensure all required functionality is preserved

```
public class AudioReference extends Reference {  
    public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH }  
    private AudioQuality audioQuality;  
  
    public AudioReference(AudioQuality quality) {  
        this.audioQuality = quality;  
    }  
  
    public static Builder builder() {  
        return new Builder();  
    }  
  
    public static class Builder {  
        private AudioQuality audioQuality;  
        private boolean isDownloadable;  
        private String title;  
        private String description;  
        private String link;  
        private String accessRights;  
        private String license;  
        private String language;  
        private int rating;  
        private int viewCount;  
        private int shareCount;  
  
        // Regular builder methods  
        public Builder audioQuality(AudioQuality quality) {  
            this.audioQuality = quality;  
            return this;  
        }  
  
        // ... other existing builder methods ...  
  
        // Adapter-style builder method  
        public Builder fromLists(List<String> properties, List<Integer> intProperties,  
                               AudioQuality audioQuality, boolean isDownloadable) {  
            this.audioQuality = audioQuality;  
            ...  
        }  
    }  
}
```

```
public Builder fromLists(List<String> properties, List<Integer> intProperties,
                        AudioQuality audioQuality, boolean isDownloadable) {
    this.audioQuality = audioQuality;
    this.isDownloadable = isDownloadable;
    this.title = properties.get(0);
    this.description = properties.get(1);
    this.link = properties.get(2);
    this.accessRights = properties.get(3);
    this.license = properties.get(4);
    this.language = properties.get(5);
    this.rating = intProperties.get(0);
    this.viewCount = intProperties.get(1);
    this.shareCount = intProperties.get(2);
    return this;
}

public void updateExisting(AudioReference reference) {
    reference.editBasic(title, description, link);
    reference.updateAccessControl(accessRights, license, isDownloadable);
    reference.setStats(rating, language, viewCount, shareCount);
    reference.audioQuality = this.audioQuality;
}

public AudioReference build() {
    AudioReference reference = new AudioReference(audioQuality);
    updateExisting(reference);
    return reference;
}
}

public void editBasic(String title, String description, String link) {
    updateMetadata(title, description, link);
}

public void editAudioAdapter(List<String> properties, List<Integer> intProperties,
                            AudioQuality audioQuality, boolean isDownloadable) {
    Builder builder = builder()
        .fromLists(properties, intProperties, audioQuality, isDownloadable);
    builder.updateExisting(this);
}

public static AudioQuality audioQualityAdapter(String quality) {
    return switch (quality.toLowerCase()) {
        case "low" -> AudioQuality.LOW;
        case "medium" -> AudioQuality.MEDIUM;
        case "high" -> AudioQuality.HIGH;
        case "very_high" -> AudioQuality.VERY_HIGH;
        default -> throw new IllegalArgumentException("Invalid audio quality");
    };
}
```

```
    public AudioQuality getAudioQuality() {
        return audioQuality;
    }
}

✍️ 🗑️ 🖌️
```

luisFelippeM X

Cannot resolve method 'editAudio' in 'AudioReference'

```
public class AudioReferenceBuilder { private AudioReference.AudioQuality audioQuality; private boolean isDownloadable; private String title; private String description; private String link; private String accessRights; private String license; private String language; private int rating; private int viewCount; private int shareCount;
```

```
public AudioReferenceBuilder withAudioQuality(AudioReference.AudioQuality audioQuality) {
    this.audioQuality = audioQuality;
    return this;
}

public AudioReferenceBuilder withDownloadable(boolean isDownloadable) {
    this.isDownloadable = isDownloadable;
    return this;
}

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

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

public AudioReferenceBuilder withLink(String link) {
    this.link = link;
    return this;
}

public AudioReferenceBuilder withAccessRights(String accessRights) {
    this.accessRights = accessRights;
    return this;
}

public AudioReferenceBuilder withLicense(String license) {
    this.license = license;
    return this;
}
```

```
public AudioReferenceBuilder withLanguage(String language) {
    this.language = language;
    return this;
}

public AudioReferenceBuilder withRating(int rating) {
    this.rating = rating;
    return this;
}

public AudioReferenceBuilder withViewCount(int viewCount) {
    this.viewCount = viewCount;
    return this;
}

public AudioReferenceBuilder withShareCount(int shareCount) {
    this.shareCount = shareCount;
    return this;
}

public void buildInto(AudioReference audioReference) {
    audioReference.editAudio(audioQuality, isDownloadable, title, description,
                           link, accessRights, license, language, rating, viewCount, shareCount);
}
```

}

 GitHub Copilot

Plan:

1. Move AudioReferenceBuilder as inner static class of AudioReference
2. Update field access levels
3. Rename methods to match standard builder pattern
4. Add direct build method to create new instances
5. Update buildInto method to use correct access methods

Implementation:



GitHub Copilot

Plan:

1. Move AudioReferenceBuilder as inner static class of AudioReference
2. Update field access levels
3. Rename methods to match standard builder pattern
4. Add direct build method to create new instances
5. Update buildinto method to use correct access methods

Implementation:

```
public class AudioReference extends Reference {  
    public enum AudioQuality { LOW, MEDIUM, HIGH, VERY_HIGH }  
    private AudioQuality audioQuality;  
  
    public static class Builder {  
        private AudioQuality audioQuality;  
        private boolean isDownloadable;  
        private String title;  
        private String description;  
        private String link;  
        private String accessRights;  
        private String license;  
        private String language;  
        private int rating;  
        private int viewCount;  
        private int shareCount;  
  
        public Builder audioQuality(AudioQuality quality) {  
            this.audioQuality = quality;  
            return this;  
        }  
  
        public Builder downloadable(boolean isDownloadable) {  
            this.isDownloadable = isDownloadable;  
            return this;  
        }  
  
        public Builder title(String title) {  
            this.title = title;  
            return this;  
        }  
  
        public Builder description(String description) {  
            this.description = description;  
            return this;  
        }  
    }  
}
```

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

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

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

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

public Builder rating(int rating) {
    this.rating = rating;
    return this;
}

public Builder viewCount(int viewCount) {
    this.viewCount = viewCount;
    return this;
}

public Builder shareCount(int shareCount) {
    this.shareCount = shareCount;
    return this;
}

public AudioReference build() {
    AudioReference reference = new AudioReference(audioQuality);
    reference.editBasic(title, description, link);
    reference.updateAccessControl(accessRights, license, isDownloadable);
    reference.setStats(rating, language, viewCount, shareCount);
    return reference;
}
```

```
public static class Builder {
    private static final int TITLE_INDEX = 0;
    private static final int DESCRIPTION_INDEX = 1;
    private static final int LINK_INDEX = 2;
    private static final int ACCESS_RIGHTS_INDEX = 3;
    private static final int LICENSE_INDEX = 4;
    private static final int LANGUAGE_INDEX = 5;

    private static final int RATING_INDEX = 0;
    private static final int VIEW_COUNT_INDEX = 1;
    private static final int SHARE_COUNT_INDEX = 2;

    private void setStringProperties(List<String> properties) {
        this.title = properties.get(TITLE_INDEX);
        this.description = properties.get(DESCRIPTION_INDEX);
        this.link = properties.get(LINK_INDEX);
        this.accessRights = properties.get(ACCESS_RIGHTS_INDEX);
        this.license = properties.get(LICENSE_INDEX);
        this.language = properties.get(LANGUAGE_INDEX);
    }

    private void setIntegerProperties(List<Integer> intProperties) {
        this.rating = intProperties.get(RATING_INDEX);
        this.viewCount = intProperties.get(VIEW_COUNT_INDEX);
        this.shareCount = intProperties.get(SHARE_COUNT_INDEX);
    }

    public Builder fromLists(List<String> properties, List<Integer> intProperties,
                           AudioQuality audioQuality, boolean isDownloadable) {
        this.audioQuality = audioQuality;
        this.isDownloadable = isDownloadable;
        setStringProperties(properties);
        setIntegerProperties(intProperties);
        return this;
    }
}
```