1. Class Overview

Package: entityClasses
Class: Post
Version: 1.01 (2025-10-16)
Author: Suurya Ganti

Purpose

The Post class represents a student post or question in the Student Discussion System.
It stores all key attributes such as author, title, content, thread category, and timestamps and supports full **CRUD (Create, Read, Update, Delete)** operations with validation.
It serves as the foundational data entity for the Students User Stories in TP2 and anticipates later staff moderation and analytics features for TP3.

Design Summary

- Implements **Create**, **Read**, **Update**, and **Delete** methods.
- Ensures **input validation** on all fields (title, content, author, thread).
- Supports **soft deletion** for moderation and data integrity.
- Generates **unique short IDs** for UI friendliness.
- Maintains **timestamps** for analytics and sorting.
- Prepares for **reply tracking**, moderation, and participation evaluation.

---

2. Attributes and Rationale

Attribute	Type	Description	Rationale / Source
postId	String	Unique identifier prefixed with P- and 5-character code.	Provides concise, human-readable identifiers similar to Ed Discussion post IDs.

authorUsername	String	Username of the student who created the post.	Required for authorship, grading, and participation tracking.
title	String	Title or short subject line of the post.	Makes threads scannable; validates readability and context.
content	String	The main body of the post.	Core data for discussion; validation ensures sufficient substance.
thread	String	Category (e.g., General, Homework).	Allows filtering and organization by topic.
createdAt	LocalDateTime	Timestamp of creation.	Used for chronological sorting and analytics.
updatedAt	LocalDateTime	Timestamp of last edit.	Tracks changes for version control and edit history.
isDeleted	boolean	Marks soft deletion status.	Maintains moderation capability without permanent data loss.
replyCount	int	Number of replies to the post.	Used for engagement and participation metrics.

Constants for Validation
MIN_TITLE_LENGTH, MAX_TITLE_LENGTH, MIN_CONTENT_LENGTH, MAX_CONTENT_LENGTH define input boundaries.
DEFAULT_THREAD ensures consistent fallback when no thread is provided.

---

3. CRUD and Validation Summary

Operation	Method(s)	Description	Requirements Satisfied
Create	Constructors Post() and Post(String, String, String, String)	Instantiates a validated post with auto-generated ID and timestamps.	REQ-POST-CRUD-Create, REQ-VALID-Author, REQ-VALID-Title

Read	Getters and getSummary()	Retrieves post data and formatted output for UI lists.	REQ-POST-CRUD-Read
Update	setTitle(), setContent(), setThread()	Allows editing of post fields while maintaining timestamps.	REQ-POST-CRUD-Update, REQ-VALID-Length
Delete	markAsDeleted(), restore()	Implements soft deletion / restoration of posts.	REQ-POST-CRUD-Delete
Auxiliary	incrementReplyCount(), decrementReplyCount()	Tracks engagement count.	REQ-POST-REPLYCOUNT

Each CRUD operation performs runtime validation and preserves integrity of the data model.

---

4. Input Validation

Title Validation

- Minimum: 3 characters
- Maximum: 200 characters
- Throws IllegalArgumentException for invalid input

Content Validation

- Minimum: 10 characters
- Maximum: 5000 characters
- Rejects null or empty text

Thread Validation

- Defaults to “General” if null or empty

Author Validation

- Must be non-null and non-empty

These checks ensure high-quality posts consistent with Ed Discussion's standards.

---

5. Method Summaries (Javadoc Extracts)

Constructors

```
/**
 * Default constructor.
 * Generates a unique 5-character ID prefixed with 'P-'.
 * Initializes timestamps and default thread.
 */
public Post()

/**
 * Parameterized constructor for creating a validated Post object.
 * @param authorUsername Username of the creator
 * @param title Title of the post (validated for length)
 * @param content Main body text (validated)
 * @param thread Discussion category
 */
public Post(String authorUsername, String title, String content, String thread)
```

Update Operations

```
/**
 * Updates the post title.
 * Validates length and updates timestamp.
 * @throws IllegalArgumentException if title length invalid.
 */
public void setTitle(String title)

/**
 * Updates post content with validation and refreshes updatedAt.
 */
public void setContent(String content)

/**
```

```
 * Assigns a thread category or defaults to 'General' if blank.
 */
public void setThread(String thread)
```

Delete/Restore

```
/** Soft deletes the post; retains data for moderation. */
public void markAsDeleted()

/** Restores a previously deleted post. */
public void restore()
```

Utility Methods

```
/** Formats timestamps for UI display consistency. */
public String getFormattedCreatedAt()
public String getFormattedUpdatedAt()

/** Returns readable summary of post metadata for list views. */
public String getSummary()
```

---

6. Alignment with Student User Stories

User Story	Implementation Detail
US-1: Students can create new posts	Constructors handle creation, ID generation, and validation.
US-2: Students can edit posts	setTitle() and setContent() allow controlled editing.
US-3: Students can delete or restore posts	markAsDeleted() and restore() manage visibility.
US-4: Posts display title, author, and timestamps	getSummary() formats readable info for GUIs.
US-5: Students can categorize posts by topic	setThread() assigns threads, defaulting to General.
US-6: Replies are counted for engagement	replyCount and increment/decrement methods track metrics.

---

7. Anticipation of Staff Epics (TP3 Readiness)

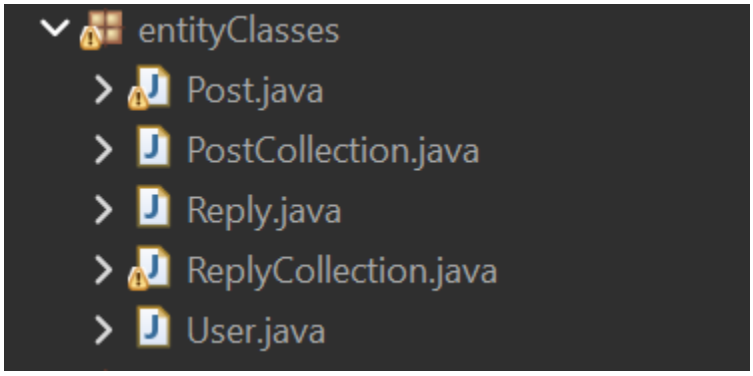
- isDeleted and timestamps enable moderation workflows and audit trails.
- replyCount supports participation analytics for instructors.
- Strict validation maintains clean datasets for staff reports.
- SecureRandom ID generation ensures traceable but concise identifiers.

---

8. Internal Comments and Screenshot Captions

Screenshot 1 – Package Placement:

Figure 3.1: entityClasses → Post.java visible in IDE confirming correct placement



Screenshot 2 – CRUD and Validation:

Figure 3.2: Highlight of setTitle() showing validation and timestamp update

```
public void setTitle(String title) {
    if (title == null || title.trim().isEmpty())
        throw new IllegalArgumentException("Post title cannot be null or empty.");
    String trimmed = title.trim();
    if (trimmed.length() < MIN_TITLE_LENGTH)
        throw new IllegalArgumentException("Post title must be at least " + MIN_TITLE_LENGTH + " characters.");
    if (trimmed.length() > MAX_TITLE_LENGTH)
        throw new IllegalArgumentException("Post title cannot exceed " + MAX_TITLE_LENGTH + " characters.");
    this.title = trimmed;
    this.updatedAt = LocalDateTime.now();
}
```

Screenshot 3 – ID Generation:

Figure 3.3: gen5() method using SecureRandom to ensure unique, non-guessable IDs.

```java
/** Generate a short 5-character ID (upper-case alphanumeric). */
private static String gen5() {
    char[] buf = new char[5];
    for (int i = 0; i < buf.length; i++) buf[i] = ALPHANUM[RNG.nextInt(ALPHANUM.length)];
    return new String(buf);
}
```

Screenshot 4 – Soft Delete:

Figure 3.4: markAsDeleted() demonstrates soft deletion pattern for moderation instead of physical removal.

```java
public void markAsDeleted() {
    this.isDeleted = true;
    this.updatedAt = LocalDateTime.now();
}
```

Screenshot 5 – Summary Output:

Figure 3.5: getSummary() truncates content to 100 characters for readability

```java
public String getSummary() {
    String status = isDeleted ? "[DELETED] " : "";
    String preview = content.length() > 100 ? content.substring(0, 97) + "..." : content;
    return String.format("%s%s\nBy: %s | Thread: %s | Replies: %d | %s",
            status, title, authorUsername, thread, replyCount, getFormattedCreatedAt());
}
```

---

9. Requirements Traceability Table

REQ ID	Description	Evidence
REQ-POST-CRUD-Create	Create new post	Constructor Javadoc + code screenshot
REQ-POST-CRUD-Read	Display post info	getSummary() method
REQ-POST-CRUD-Update	Edit post data	setTitle(), setContent()

REQ-POST-CRUD-Delete	Soft delete / restore	markAsDeleted(), restore()
REQ-VALID-Length	Input constraints enforced	Exception logic in setters
REQ-POST-THREAD	Categorization	setThread() default behavior
REQ-POST-REPLYCOUNT	Reply tracking	incrementReplyCount(), decrementReplyCount()
REQ-POST-TIMESTAMP	Time tracking	getFormattedCreatedAt(), getFormattedUpdatedAt()

---

10. Conclusion

The Post class fulfills all TP2 Task 3 deliverables:

- Located correctly in entityClasses
- Fully implements CRUD functionality with validation
- Attributes cover all Student User Stories and anticipated staff needs
- Includes clear Javadoc documentation and rationale for design choices
- Internal comments clarify purpose and requirement coverage