AI Usage Report

(Group – 11)

Ceylin Oner

Farid Aljanov

Mehmet Alperen Calıs

Md Mashroor Zilan Snigdho

Department of Statistics, Middle East Technical University

STAT295: Object Oriented Programming

Dr. Aysegul Ozkaya Eren

May 20, 2025

Usage 1:

What question did I ask AI Tool?

How can I use Generic like TypeScript?

What responses did I receive?

Using generics in Java like:

```
List<String> names;
```

How did I incorporate or modify the AI Tool's suggestions?

Then I used to intantiate the fields like, `private List<Book> books;`

Usage 2:

What question did I ask AI Tool?

Should I check for null values when adding books or members in Java?

What responses did I receive?

It is recommended to always validate inputs to avoid `NullPointerException`.

How did I incorporate or modify the AI Tool's suggestions?

So, I added null checks in the `addBook`, `removeBook`, `addMember`, and `removeMember` methods.

Usage 3:

What question did I ask AI Tool?

How can I ensure that book and member IDs are unique when adding them to a list?

What responses did I receive?

It suggested checking if a book/member with the same ID already exists in the list before adding.

How did I incorporate or modify AI Tool's suggestions?

I added ID existence checks in `addBook` and `addMember` methods using `getBook(id)` and `getMember(id)`.

Usage 4:

What questions did I ask AI Tool?

Should I throw custom exceptions instead of `IllegalArgumentException`?"

What responses did I receive?

It suggested using custom exceptions for more meaningful error messages in large projects.

How did I incorporate or modify AI Tool's suggestions?

I kept `IllegalArgumentException` but included a human-readable error message.

Usage 5:

What questions did I ask AI Tool?

How can I use `super()`?

What responses did I receive?

`super()` must be the first line in the child class constructor.

How did I incorporate or modify AI Tool's suggestions?

I wrote it as instructed, pass any fields if those are the same as parent class in child class.

Usage 6:

What questions did I ask AI Tool?

What's `@override`? How to use it?

What responses did I receive?

`@override` is an annotation in Java that tells the compiler: “I am overriding a method that comes from a superclass or an interface.” Yes — it is related to polymorphism, specifically runtime polymorphism (method overriding). You put `@Override` just above a method that overrides a method from a parent class or interface.

How did I incorporate or modify AI Tool's suggestions?

I used it to override `toString()` method.

Usage 7:

What questions did I ask AI Tool?

What's an abstract class? Benefit?

What responses did I receive?

Cannot be instantiated directly. It is meant to be extended by other classes that provide specific implementations for the abstract methods.

How did I incorporate or modify AI Tool's suggestions?

I made the `Person` class abstract. Other classes use it.

Usage 8:

What question did I ask AI Tool?

How should I validate email addresses in a constructor?

What responses did I receive?

You should check for `null` and ensure the email contains `@` and `.` characters to avoid invalid formats.

How did I incorporate or modify the AI Tool's suggestions?

I added a condition in the `Member` constructor that throws an `IllegalArgumentException` if the email is null or does not contain `@` and “`.`”.

Usage 9:

What question did I ask AI Tool?

How can I implement borrowing limits and prevent duplicate borrowing?

What responses did I receive?

The tool suggested checking if the book is already in the `borrowedBooks` list and limiting the total borrowed books with a size check.

How did I incorporate or modify the AI Tool's suggestions?

I added checks in the `borrowBook()` method to prevent duplicate borrowing and limit borrowing to 5 books per member.

Usage 10:

What question did I ask AI Tool?

How do I safely return books and ensure valid operations in `returnBook()`?

What responses did I receive?

Add null checks and verify if the book is in the borrowed list before allowing a return.

How did I incorporate or modify the AI Tool's suggestions?

In the `returnBook()` method, I checked if the book is null and confirmed it was actually borrowed before removing it.

Usage 11:

What question did I ask AI Tool?

Should I override `toString()` for custom output in classes?

What responses did I receive?

Yes, overriding `toString()` provides a meaningful string representation and helps with debugging and logging.

How did I incorporate or modify the AI Tool's suggestions?

I overrode the `toString()` method in the `Member` class to return member details like name, email, borrowed book count, and ID.

Usage 12:

What question did I ask AI Tool?

How should I set loan due dates based on the borrowing date?

What responses did I receive?

You can use `LocalDate.plusDays(7)` to automatically calculate the due date as one week after borrowing.

How did I incorporate or modify the AI Tool's suggestions?

In the `Loan` constructor, I set `dueDate` to `borrowDate.plusDays(7)` to enforce a 7-day borrowing policy.

Usage 13:

What question did I ask AI Tool?

How do I validate date logic when working with `LocalDate`?

What responses did I receive?

Ensure that `returnDate` is not before `borrowDate`, and always check for null before doing comparisons.

How did I incorporate or modify the AI Tool's suggestions?

In both the constructor and the `setReturnDate()` method of `Loan`, I added conditions to throw an error if the return date is before the borrow date.

Usage 14:

Question that I ask:
In a library system, how can I construct a base class for both Books and DigitalResources classes.

The response:
ChatGPT suggested that create an abstract class with shared attributes such as id, title and author, and define an abstract method like toString(), and in this way polymorphism would be satisfied.

How did I incorporate or modify the AI Tool's suggestions?
I defined an abstract base class called LibraryResource, that has common fields and an abstract toString() method. Then I extend this class for both Book and DigitalResource classes.

Usage 15:

Question that I ask:
How do I ensure constructor parameters to prevent bad inputs like negative IDs or empty titles?

The response:
ChatGPT suggested that by IllegalArgumentException with appropriate messages, would make sure that invalid objects are not created.

How did I incorporate or modify the AI Tool's suggestions?
I added parameter checks in all constructors, such as making sure IDs are positive and strings are not null or empty. I used IllegalArgumentException with clean messages.

Usage 16:

Question that I ask:
How can I ensure that the number of available copies does not drop below 0 when a book is borrowed?

The response:
ChatGPT suggested that before allowing a book to be borrowed, the value of copiesAvailable should be checked to see if it is greater than zero, and if not, an exception should be thrown.

How did I incorporate or modify the AI Tool's suggestions?
When there are no more copies to borrow, I introduced the borrowBook() method with an IllegalStateException.

Usage 17:

Question that I ask:
How can I view different information for Book and DigitalResources?

The response:
ChatGPT recommended overriding the toString() method for each subclass to display relevant aspects for each class.

How did I incorporate or modify the AI Tool's suggestions?
I optimized the toString() method for both Book and DigitalResource to display related attributes, while still displaying the shared ones such as title, author, and id.

Usage 18:

Question that I ask:
How can I successfully implement that the child class constructor properly uses the parent ones?

The response:
ChatGPT suggested that the super(...) statement should be the first line in the child class.

How did I incorporate or modify the AI Tool's suggestions?
In both Book and DigitalResource classes, I used super(id, title, author).

Usage 19:

Question that I ask:
What is the difference between IllegalArgumentException and IllegalStateException?

The response:
ChatGPT explained that IllegalArgumentException should be used when method arguments are invalid (e.g. null, negative), and IllegalStateException should be used when the object is in a state that is not suitable for the requested operation.

How did I incorporate or modify the AI Tool's suggestions?
I used IllegalArgumentException in the constructors of Book and DigitalResource to validate inputs like title, author, and publication year. I used IllegalStateException in the borrowBook() method to ensure that borrowing is only allowed when copies are available.

Usage 20:

What question did I ask AI Tool?
Should I validate all IDs and inputs in constructors?

What responses did I receive?
Yes, it's good practice to validate IDs (e.g., check for positive integers) and avoid `null` values for required fields to ensure data integrity and prevent runtime errors.

How did I incorporate or modify the AI Tool's suggestions?
In all three classes — `Reservation`, `Fine`, and `Review` — I added validation in constructors to ensure IDs are positive and essential fields like `reservationDate` and `comment` are not `null`.

Usage 21:

What question did I ask AI Tool?
How do I represent a many-to-one association between classes like Book and Member in real-life systems?

What responses did I receive?
Use association classes such as `Reservation` and `Review` that link the `Book` and `Member` entities using their IDs and store relevant data like date, rating, and comments.

How did I incorporate or modify the AI Tool's suggestions?
I implemented `Reservation` and `Review` classes to represent associations between `Book` and `Member`, including attributes like date, rating, and comment to model real-world interactions.

Usage 22:

What question did I ask AI Tool?
What is a clean way to print object details in Java?

What responses did I receive?
Override methods like `toString()` or create custom methods like `reviewDetails()` or `reservationInfo()` to return formatted summaries.

How did I incorporate or modify the AI Tool's suggestions?
I added `reservationInfo()` in the `Reservation` class and `reviewDetails()` in the `Review` class for displaying formatted information. I also overrode `toString()` in the `Fine` class.

Usage 23:

What question did I ask AI Tool?
How should I handle payments for fines to ensure the amount never becomes negative?

What responses did I receive?
Always check that the payment is positive and does not exceed the current fine amount before subtracting it.

How did I incorporate or modify the AI Tool's suggestions?
In the `payFine()` method of the `Fine` class, I added conditions to validate that the payment is positive and does not exceed the fine amount before updating the balance.

Usage 24:

What question did I ask AI Tool?
Should I allow ratings outside the 1–5 range in a review system?

What responses did I receive?
No — it recommended enforcing a rating scale (typically 1 to 5) using input validation to maintain consistency and reliability in review data.

How did I incorporate or modify the AI Tool's suggestions?
I added a condition in the `Review` constructor to throw an `IllegalArgumentException` if the rating is less than 1 or greater than 5, ensuring only valid ratings are accepted.

Usage 25:

What question did I ask AI Tool?
Is it necessary to check for null values in optional string fields like comments?

What responses did I receive?
Yes — it's a good idea to check for `null` to avoid `NullPointerException` and ensure better control over how data is handled, even if the field is optional.

How did I incorporate or modify the AI Tool's suggestions?
In the `Review` constructor, I added a null check for the `comment` field and threw an exception if it was null, preventing runtime issues and enforcing meaningful feedback.