CPT111: Java Programming

Semester 1, 2025-26

Coursework 3: Programming Project – Movie Recommendation & Tracker

Read carefully — no dispensation will be given for lack of awareness of the rules

Assignment type	This is a group programming assignment. You are required to work as group of 4 to 5 students, which will be allocated randomly. (You should be able to find your group members' information from Learning Mall (LM), and should contact the instructors if there is any problem.)
Weighting	Total marks available: **100**, accounting for **30%** of overall module marks.
Release date	**Beginning of Week 8: Monday, 3 November 2025, 11:00 CST**
Due date	**End of Week 13: Sunday, 14 December 2025, 23:59 CST**

Learning outcomes

A. Understand and appreciate the principles of using object-oriented programming techniques for the construction of professional robust, maintainable programs deployed to meet real world business goals;
B. Design, write, compile, test, debug and execute object-oriented programs using an integrated development environment (IDE);
C. Effectively develop object-oriented programs as a member or a leader of a software development team with continuous development strategy supported by AI technology;
D. Implement object-oriented programming to represent, display, and manipulate data while mitigating security risks;
E. Evaluate legal, social, environmental, ethical, diversity, inclusion, and intellectual property issues related to object-oriented program development.
F. Apply knowledge of engineering management principles, commercial context, project and change management in object-oriented program development.

Submission platform	Each group is required to submit an electronic copy of your assignment via LM. You are allowed **ONE** submission only. **It is your responsibility to upload the correct document.**

Late submissions	Penalties will apply for any work submitted after the due date unless you have obtained a formal extension prior to the date for submission. **The penalty applied will be 5% of the available marks for the assignment for each working day or part thereof** that the assignment is late. The penalty will be capped at five working days from the assignment deadline. Work submitted after five working days will receive a grade of **zero**.
Submission confirmation	**We strongly advise you to double check that you have submitted the correct document / final version of your answer.** **Submission of incorrect file:** If you have submitted the incorrect file, you should email the correct file to the instructors **prior to the deadline. Submitting the incorrect file can result in failure.**
Special consideration	Requests for an extension due to illness, misadventure, or other extenuating circumstances beyond your control will only be considered via a formal application for mitigating circumstances (MC) through e-Bridge.
Report format	**ALL** answers **must** be written in English. The report must: • be submitted as a PDF of at most 15 pages (do not submit .doc, .docx, or Apple Pages) • contain headings and subheadings • have 3 cm margins • use a legible font (e.g., Calibri, Arial or Times New Roman) • be presented in 11 point font size with 1.5 line spacing • be paginated
Artificial Intelligence (AI) Permissions	For this assignment, you are permitted to make **non-substantive use** of AI. This means that AI can only be employed for minor tasks such as grammatical corrections, formatting adjustments, or generic suggestions (e.g., "improve clarity"). You should maintain a record of unedited versions of your work prior to processing by AI, which you may be required to provide as part of the verification process. If you are unsure on whether a specific use of AI is allowed or not, please consult the module leader. In the references of your report, you should include a brief citation of the tool name and version, and what it was used for, e.g.: *[1] DeepSeek V3.1, available at https://www.deepseek.com. Used for grammatical proofreading.*

Plagiarism and academic misconduct	It is assumed that you are thoroughly familiar with the policies of XJTLU regarding academic misconduct and plagiarism. Ignorance of the rules is not an acceptable defence against an allegation of academic misconduct. **There are no excuses for engaging in plagiarism.** Assignment answers will be checked for plagiarism. Impermissible similarities between student answers (current and former) can be detected by academic integrity software and by instructor and will be referred to the School's Examination Officer for investigation. Penalties will follow those of the University's Academic Integrity Policy on e-Bridge and can ranged from capped marks to expulsions from the university. The use of AI tools beyond the scope outlined above would constitute a breach of the University Academic Integrity Policy and will result in penalties, which may include an overall mark of **zero** for the assignment and a formal notification of academic misconduct.
Peer Evaluation	**Each group member is required to submit a peer evaluation form, using LM, to describe and rank their contributions and those of their group members.** Each group member's individual mark will depend on this peer evaluation – e.g., group members who contributed substantially less will receive a lower mark relative to the group's mark; those who contributed substantially more will receive a higher mark relative to the group's mark.
Tips	• Read the questions carefully. • Write succinctly and avoid repetition. • Avoid being overly descriptive. • Remember to save/back up your work regularly. XJTLU provides all students free access to XJTLU Box. It may be prudent to save your work to your XJTLU account so that you can access it from multiple devices in case you encounter hardware issues. • You are encouraged to post administrative/procedural questions about the assignment on the LM Q&A Forum. The instructors will answer for the benefit of all students.

Coursework 3: Programming Project – Movie Recommendation & Tracker

The purpose of this assignment is to design and implement a **Movie Recommendation and Tracker System**. The program will allow users to log in, manage watchlists, record viewing history, and receive simple movie recommendations based on their preferences.

The purpose of this coursework is to consolidate your knowledge of **object-oriented programming, file processing, and exception handling**.

1. Background

Movie recommendation systems are widely used by streaming platforms to help users discover new content. These systems typically combine user preferences with a large movie database to provide personalized suggestions.

In this coursework, your task is to implement a **Movie Recommendation and Tracker System**. Your system should allow users to log in, maintain watchlists, record viewing history, and receive simple recommendations based on, for example, their most-watched genres. The project emphasizes **file handling, object-oriented design, and error handling**.

2. Specification

Your implementation must satisfy the following requirements:

1. When not logged in, the user is presented with a main menu on the command line, where they can select from the following options – all these options should be implemented and functional:
   - *Login:* Log in with a username and password
   - *Exit:* Exit the program

2. When logged in, the main menu presents the user with the following options – all these options should be implemented and functional:
   - *Browse movies:* List all the movies stored in the file.
   - *Add movie to watchlist:* Add a movie to the user's personal watchlist.
   - *Remove movie from watchlist:* Remove a movie from the user's personal watchlist.
   - *View watchlist:* List all the movies in the user's personal watchlist.
   - *Mark movie as watched:* Add a movie to the user's personal viewing history. Additionally, if the movie is currently in the user's personal watchlist, it should be removed from the watchlist.
   - *View history:* List all the movies in the user's personal viewing history.
   - *Get recommendations:* List the **top-N** recommended movies for the user, based on the user's watchlist and/or viewing history.
   - *Logout:* Log out of the user's account.

3. Each user has a personal watchlist and viewing history.

4. User data (for all users) is stored in a single **CSV file** with fields: *Username*, *Password*, *Watchlist*, and *History*
   - **A pre-filled CSV file with existing user data is already provided to you.**

5. Movies are stored in a **CSV file** with fields: *ID*, *Title*, *Genre*, *Year*, and *Rating* (from 0.0 to 10.0).
   - **A pre-filled CSV file with a library of movies is already provided to you.**

6. The recommendation algorithm (what runs when the user selects *'get recommendations'*) should be simple, and should display the **top-N recommended**

movies (where *'N'* is an argument passed in to the recommendation function). For example, it can recommend movies based on the user's most watched genres.

7. **Technical Requirements:**
   - Use **object-oriented principles**. At minimum, classes such as: `Movie`, `User`, `Watchlist`, `History`, and `RecommendationEngine` should be implemented.
   - Use **ArrayList** and/or **HashMap** for storing movies and user data.
   - Use **file I/O** for loading movie data and loading/saving user data.
   - Implement **exception handling** to prevent crashes.
   - **Constraint**: You must use only **Java libraries covered in this course** (e.g., `ArrayList`, `HashMap`, `File`, `Scanner`, `BufferedReader`). Use of other libraries will result in **zero marks**.

8. **Advanced Features:** You also need to implement at least one advanced feature in the system (you can implement more than one advanced feature for bonus marks). Specifically, the implementation of these features may involve **adding or updating columns in the data file and UI system**. As examples, you may choose one or more from the following:
   - Functionality for creating a new user account.
   - Functionality for changing a user's password.
   - A recommendation engine that supports **multiple strategies** (e.g., recommend by genre, year, rating, etc.), switchable at runtime.
   - A Graphical User Interface (GUI), using JavaFX, for browsing movies, managing watchlists, and viewing recommendations (replacing the command-line menu).
   - Subclassing `User` into `BasicUser`, `PremiumUser`, etc., each with different privileges.
   - Subclassing `Movie` into `FeatureFilm`, `ShortFilm`, `Documentary`, etc.
   - Functionality for hashing (encrypting) users' passwords before saving them to the CSV file.
   - Other functionalities you can think of.

3. Deliverables

Submit to Learning Mall the following:

1. **Java source code** (zipped).
2. **Written report** (PDF of at most 15 pages; do not submit .doc, .docx, or Apple Pages), which must include:
   - An explanation of the system design and class structure.
   - A description of the recommendation algorithm.
   - A description of the testing methods and results.
   - A description of the mechanisms in place for handling exceptions, invalid inputs, etc.
   - Example screenshots of program execution.
   - A description of how you (as a group) are applying software engineering team management practices to complete this project.

    - An ethical assessment of the system, including (for example) potential privacy issues, a reflection of how the ratings or recommendation system could be potentially abused, etc. Describe potential solutions to these issues.
3. **Peer evaluation form** (each group member is required to complete this on LM).

Only one submission per group is required.

4. Marking

Criteria	Excellent (Full Marks)	Good	Satisfactory	Poor / Missing	Marks
Functionality (35%)	All required features (login, browse, watchlist, history, recommendations, etc.) fully implemented and working correctly; recommendation engine produces appropriate results.	Most features implemented; minor errors in some functionality.	Some features missing or partially working; significant limitations in usability.	Very limited functionality or fails to run.	35
Object-Oriented Design (25%)	Clear, well-structured classes with meaningful names; correct use of encapsulation; modular and extensible design; strong application of OOP principles.	Mostly well-structured; some minor issues with encapsulation or design clarity.	Some OOP principles applied, but design is weak, inconsistent, or poorly organized.	Minimal or no evidence of OOP design; mostly procedural code.	25
Robustness (10%)	Handles invalid input gracefully in all cases; program never crashes under normal use.	Mostly robust; occasional crashes or unhandled cases.	Some error handling implemented; crashes possible in common situations.	Little to no error handling; frequent crashes.	10
Commenting & Naming Convention (10%)	Code is thoroughly commented; variables, methods, & classes have meaningful and appropriate names; consistent style throughout.	Mostly well-commented and named; some minor inconsistencies.	Some comments/naming missing or inconsistent; readability affected.	Very poor commenting/naming; unclear or confusing code.	10
Report (15%)	Clear, well-structured, & complete; explains design, class structure, & recommendation logic; includes detailed testing evidence, description of exception handling, screenshots; describes SE team management practices; thorough ethical assessment.	Generally clear and complete; some gaps in explanation or missing details in testing.	Basic report with limited explanation; lacks depth or supporting evidence.	Very poor, incomplete, or missing report.	15
Advanced Features (Mandatory) (5%)	At least one advanced feature fully implemented.	One advanced feature implemented but with minor flaws.	Attempted an advanced feature but incomplete or buggy.	No advanced feature implemented (automatic failure in this category).	5
Bonus Features (+10)	More than 1 advanced feature implemented, all working well.	More than 1 attempted, most working.	Attempted more than 1 but with major issues.	Only 1 or none attempted.	up to +10
Total					**Max: 100 (incl. bonus)**