DANH SÁCH 20 ĐỀ TÀI BÀI TẬP LỚN

Một số đề tài bên dưới có thể mang tính chất kỹ thuật nhằm mục tiêu đánh giá khai triển kỹ năng lập trình OOP hơn là tĩnh thực tiễn (không có tính khả thi ứng dụng khi triển khai thực tế do giới hạn của dạng ứng dụng Console App)

Nhóm 1 – Quản lý Thư viện s Giáo dục

1. Quản lý mượn/trả sách thư viện

Đề tài 1 – Quản lý mượn/trả sách thư viện

1. Mục tiêu

Xây dựng **Java OOP Console Application** giúp thư viện quản lý sách, thành viên, và phiếu mượn/trả. Dữ liệu lưu bằng **tệp CSV hoặc Serializable**.

2. Chức năng chính

- Quản lý sách: thêm/sửa/xóa/tìm kiếm.
- Quản lý thành viên: đăng ký, cập nhật thông tin.
- Quản lý phiếu mượn/trả: tạo phiếu, tính tiền phạt.
- Thống kê: top 5 sách mượn nhiều nhất, tổng tiền phạt.
- Lưu/đọc dữ liệu từ tệp khi khởi động/thoát.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Item (abstract) → Book, Magazine
  - Person (abstract) → Student, Lecturer
- **Interface**: Persistable (load/save), Searchable<T>.
- **Ngoại lệ tùy biến**: ItemNotFoundException, OutOfCopiesException.

4. Lưu trữ dữ liệu

- books.csv | data.bin
- members.csv | data.bin
- loans.csv | data.bin

5. Chia việc cho 3 thành viên

- **Thành viên A**: Thiết kế domain (Item, Book, Magazine, Person…), xử lý kế thừa.
- **Thành viên B**: Tầng lưu trữ (CSV/Serializable), xử lý đọc/ghi, validate dữ liệu.
- **Thành viên C**: Xây dựng CLI menu, nghiệp vụ mượn/trả, báo cáo thống kê.

6. Bàn giao

- **Dataset mẫu**: ≥ 10 sách, ≥ 5 thành viên, ≥ 5 phiếu mượn.
- **Sơ đồ lớp** (PlantUML/Draw.io).
- **10 test case**: input → output kỳ vọng.
- **Hướng dẫn chạy** C ảnh minh họa CLI.

GỢI Ý MỘT SỐ NỘI DUNG

Mục tiêu: Quản lý tài liệu, thành viên, mượn/trả, phạt trễ.

Chức năng: CRUD sách/thành viên; tạo phiếu mượn; trả C tính phạt; thống kê top

sách/người mượn; quá hạn.

Kế thừa:

- Item (abstract) → Book, Magazine (override getLateFeePerDay()),
- Person (abstract) → Student, Lecturer (override getBorrowLimit()).

Lưu tệp: data/items.csv, data/people.csv, data/loans.csv (hoặc data/library.bin).

Chia việc: A (domain + exceptions), B (CSV/Serializable + repo), C (CLI + báo cáo).

Mẫu skeleton chung (áp dụng cho mọi đề tài — chỉnh tên lớp/thuộc tính theo từng bài):

2. Đăng ký học phần C tính học phí

Đề tài 2 – Đăng ký học phần s tính học phí

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý đăng ký học phần và tính học phí cho sinh viên.

Ứng dụng cho phép quản lý **học phần**, **lớp học phần**, **giảng viên**, **sinh viên** và **phiếu đăng ký**.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý học phần**: thêm, sửa, xóa, tìm kiếm theo mã hoặc tên học phần.
- **Quản lý lớp học phần**: tạo, sửa, xóa lớp; gán giảng viên, phòng học, ca học, giới hạn số lượng sinh viên.
- **Quản lý sinh viên và giảng viên**: thêm mới, chỉnh sửa thông tin cơ bản.
- **Đăng ký học phần**:
  - Sinh viên tìm kiếm học phần/lớp học phần.
  - Đăng ký lớp học phần nếu thỏa mãn các điều kiện.
  - Tự động kiểm tra: tiên quyết, trùng lịch, giới hạn số tín chỉ, số chỗ còn trống.
- **Hủy đăng ký**: sinh viên có thể hủy trước hạn, hệ thống tính toán hoàn/khấu trừ học phí theo quy định.
- **Tính học phí**: học phí theo số tín chỉ, có phụ phí cho lớp thực hành/lab.
- **Thống kê s báo cáo**:
  - Tổng học phí mỗi sinh viên trong học kỳ.
  - Danh sách sinh viên đăng ký theo từng lớp học phần.
  - Doanh thu học phí theo học kỳ.
- **Lưu trữ dữ liệu bền vững**: ghi/đọc tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Course *(abstract)* → TheoryCourse, LabCourse
    - getCreditFee() được **override** để tính phí tín chỉ lý thuyết/lab.
  - Person *(abstract)* → Student, Lecturer
- **Interface**:
  - Persistable (load/save dữ liệu).
  - Searchable<T> (tìm kiếm theo từ khóa).
- **Ngoại lệ tùy biến**:
  - PrerequisiteNotMetException
  - ScheduleConflictException

- CourseFullException
- CreditLimitExceededException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - courses.csv — thông tin học phần.
  - class_sections.csv — thông tin lớp học phần.
  - students.csv — thông tin sinh viên.
  - lecturers.csv — thông tin giảng viên.
  - enrollments.csv — thông tin phiếu đăng ký.
  - prerequisites.csv — quan hệ tiên quyết giữa các học phần.
  - passed_courses.csv — danh sách học phần sinh viên đã hoàn thành.
- **Yêu cầu lưu trữ**:
  - Mã định danh tự động hoặc UUID.
  - Định dạng ngày theo chuẩn yyyy-MM-dd.
  - Đảm bảo đọc/ghi dữ liệu chính xác, tránh mất mát khi thoát ứng dụng.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 10 học phần (trong đó ≥ 2 học phần lab).
  - ≥ 5 giảng viên.
  - ≥ 6 sinh viên.
  - ≥ 8 lớp học phần.
  - ≥ 15 phiếu đăng ký.
- **Báo cáo mẫu**:
  - Danh sách học phần đã đăng ký của một sinh viên cụ thể.
  - Tổng số tín chỉ và học phí cho từng sinh viên.
  - Doanh thu học phí theo học kỳ.
- **Test case bắt buộc** (≥ 10):
  - Đăng ký thành công.
  - Đăng ký trùng lịch → báo lỗi.
  - Đăng ký vượt số tín chỉ → báo lỗi.
  - Đăng ký thiếu học phần tiên quyết → báo lỗi.
  - Đăng ký khi lớp đầy → báo lỗi.
  - Tính học phí chính xác khi có lớp lý thuyết và lab.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (Course, TheoryCourse, LabCourse, Person, Student, Lecturer), xử lý logic kiểm tra tiên quyết, trùng lịch, giới hạn tín chỉ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho học phần, lớp, sinh viên, giảng viên, phiếu đăng ký. Đảm bảo import/export và validate dữ liệu.
C	Thiết kế CLI menu, xử lý nghiệp vụ đăng ký/hủy, tính học phí, xuất báo cáo thống kê.

Yêu cầu nâng cao: Mỗi thành viên triển khai ≥ 1 tính năng bổ sung, ví dụ:

- Tìm kiếm học phần/lớp bằng từ khóa mờ (fuzzy search).
- Xuất báo cáo học phí ra CSV.
- Sắp xếp danh sách lớp học phần theo nhiều tiêu chí.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

3. Quản lý điểm và bảng xếp hạng sinh viên

Đề tài 3 – Quản lý điểm và bảng xếp hạng sinh viên

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ quản lý **điểm học phần** của sinh viên và tạo **bảng xếp hạng** theo lớp, khoa hoặc toàn trường.

Ứng dụng cho phép nhập điểm, tính điểm trung bình, xếp loại học lực và xuất báo cáo.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý học phần**: thêm, sửa, xóa, tìm kiếm theo mã hoặc tên học phần.
- **Quản lý sinh viên và giảng viên**: nhập, chỉnh sửa thông tin cơ bản.
- **Nhập điểm học phần**:
  - Giảng viên nhập điểm **quiz**, **giữa kỳ**, **cuối kỳ**, **bài tập lớn** (tuỳ môn).
  - Cho phép sửa điểm nếu nhập sai.
- **Tính điểm tổng kết s GPA**:
  - Công thức tính điểm cuối kỳ = (quiz * w1 + giữa kỳ * w2 + cuối kỳ * w3 + bài tập * w4).
  - GPA = trung bình cộng trọng số điểm học phần đã học.
- **Xếp loại học lực**:
  - Xuất sắc, Giỏi, Khá, Trung bình, Yếu theo thang điểm.
- **Tạo bảng xếp hạng**:
  - Theo lớp.
  - Theo khoa.
  - Toàn trường.
- **Thống kê s báo cáo**:
  - Danh sách sinh viên có GPA cao nhất.
  - Tỷ lệ sinh viên đạt từng mức xếp loại.
- **Lưu trữ dữ liệu**: tự động ghi/đọc tệp khi khởi động/thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Assessment *(abstract)* → Quiz, Exam, Project
    - weight() được **override** để xác định trọng số từng loại điểm.
  - Person *(abstract)* → Student, Lecturer.
- **Interface**:
  - Persistable (load/save dữ liệu).
  - Rankable (tính GPA, xếp loại).
- **Ngoại lệ tùy biến**:
  - InvalidScoreException
  - StudentNotFoundException
  - CourseNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - courses.csv — thông tin học phần.

- students.csv — thông tin sinh viên.
- lecturers.csv — thông tin giảng viên.
- grades.csv — điểm thành phần và điểm tổng kết.
- class_groups.csv — thông tin lớp/khoa của sinh viên.

- **Yêu cầu lưu trữ**:
  - Mã định danh tự động hoặc UUID.
  - Chuẩn định dạng điểm thập phân (#.##).
  - Đảm bảo đồng bộ dữ liệu giữa các tệp.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 8 học phần.
  - ≥ 20 sinh viên.
  - ≥ 3 khoa, 4 lớp.
  - ≥ 50 bản ghi điểm.
- **Báo cáo mẫu**:
  - Danh sách sinh viên kèm GPA.
  - Bảng xếp hạng top 5 sinh viên theo khoa.
  - Tỷ lệ xếp loại học lực.
- **Test case bắt buộc** (≥ 10):

1. Nhập điểm thành phần hợp lệ → tính điểm tổng kết chính xác.
2. Nhập điểm vượt giới hạn (âm hoặc >10) → báo lỗi.
3. Tính GPA cho sinh viên đã hoàn thành ≥ 5 học phần.
4. Tạo bảng xếp hạng theo lớp.
5. Tạo bảng xếp hạng theo khoa.
6. Thống kê tỷ lệ sinh viên đạt Giỏi.
7. Tìm kiếm sinh viên theo tên.
8. Tìm kiếm học phần theo mã.
9. Báo cáo danh sách sinh viên học lại học phần.
10. Xuất báo cáo bảng xếp hạng ra CSV.

6. Chia việc cho 3 thành viên (cân bằng)

Thành chính viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Assessment, Quiz, Exam, Project, Person, Student, Lecturer), xử lý logic tính điểm thành phần, điểm tổng kết, GPA, xếp loại.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho học phần, sinh viên, giảng viên, điểm số; import/export dữ liệu; validate định dạng điểm.
C	Thiết kế CLI menu, nhập/sửa điểm, xem GPA, tạo bảng xếp hạng, xuất báo cáo thống kê.

Yêu cầu nâng cao:

- Vẽ biểu đồ phân bố điểm (xuất ra CSV hoặc text-based histogram).
- Tích hợp tính năng lọc bảng xếp hạng theo GPA tối thiểu.
- Cho phép tìm kiếm mờ theo tên sinh viên/học phần.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

4. Quản lý khóa học trực tuyến C chứng chỉ

Đề tài 4 – Quản lý khóa học trực tuyến s chứng chỉ

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý **khóa học trực tuyến**, **tiến độ học tập** của học viên và **cấp chứng chỉ** khi hoàn thành.

Ứng dụng cho phép quản lý khóa học, mô-đun, giảng viên, học viên, theo dõi tiến độ, tính điểm và xuất chứng chỉ.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

---

2. Chức năng chính

- **Quản lý khóa học**: thêm, sửa, xóa, tìm kiếm khóa học.
- **Quản lý mô-đun học tập**:
  - Mỗi khóa học có nhiều mô-đun: video, quiz, bài tập lớn.
  - Cho phép bật/tắt mô-đun hoặc cập nhật nội dung.
- **Quản lý người dùng**:
  - **Học viên**: đăng ký khóa học, xem tiến độ học tập.
  - **Giảng viên**: tạo nội dung, quản lý bài kiểm tra.
- **Theo dõi tiến độ học tập**:
  - Học viên hoàn thành video, làm quiz, nộp bài tập.
  - Hệ thống cập nhật % tiến độ cho từng mô-đun và khóa học.
- **Chấm điểm s xếp loại**:
  - Quiz/bài tập được chấm tự động.
  - Tính điểm trung bình khóa học dựa trên trọng số từng mô-đun.
- **Cấp chứng chỉ**:
  - Học viên hoàn thành 100% khóa học và đạt điểm tổng kết ≥ mức yêu cầu sẽ được cấp chứng chỉ.
  - Chứng chỉ có mã duy nhất, lưu trữ lâu dài.
- **Thống kê s báo cáo**:
  - Danh sách học viên đã hoàn thành khóa học.
  - Tỷ lệ hoàn thành theo khóa học.
  - Top học viên điểm cao nhất trong từng khóa.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

---

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - User *(abstract)* → Learner, Instructor.
  - Module *(abstract)* → VideoModule, QuizModule, ProjectModule.
    - getDuration() và isCompleted() **override** tùy loại mô-đun.
- **Interface**:
  - Graded (mô-đun có tính điểm, ví dụ quiz, project).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - ModuleNotFoundException
  - CourseNotFoundException
  - InvalidProgressException
  - CertificateAlreadyIssuedException

---

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - courses.csv — thông tin khóa học.
  - modules.csv — thông tin mô-đun của từng khóa học.

- users.csv — thông tin học viên/giảng viên.
- progress.csv — trạng thái hoàn thành mô-đun theo học viên.
- grades.csv — điểm quiz, project.
- certificates.csv — chứng chỉ được cấp.

- **Yêu cầu lưu trữ**:
  - Mã khóa học, mã mô-đun, mã chứng chỉ sinh tự động hoặc UUID.
  - Trạng thái tiến độ lưu dạng phần trăm.
  - Ngày cấp chứng chỉ định dạng yyyy-MM-dd.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 5 khóa học.
  - ≥ 15 mô-đun (kết hợp video, quiz, project).
  - ≥ 10 học viên.
  - ≥ 3 giảng viên.
  - ≥ 20 chứng chỉ.
- **Báo cáo mẫu**:
  - Danh sách học viên hoàn thành khóa học.
  - Tỷ lệ hoàn thành của từng khóa.
  - Top 3 học viên điểm cao nhất.
- **Test case bắt buộc** (≥ 10):

1. Đăng ký khóa học thành công.
2. Cập nhật tiến độ video → tiến độ khóa học tăng đúng %.
3. Làm quiz, tính điểm tự động → lưu đúng file grades.csv.
4. Hoàn thành tất cả mô-đun → cấp chứng chỉ.
5. Cấp chứng chỉ lần 2 cho cùng khóa học → báo lỗi.
6. Báo cáo tỷ lệ hoàn thành chính xác.
7. Xem chứng chỉ học viên theo ID.
8. Tìm kiếm khóa học theo từ khóa.
9. Tìm kiếm học viên theo tên/email.
10. Xuất danh sách chứng chỉ ra CSV.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (User, Learner, Instructor, Module, VideoModule, QuizModule, ProjectModule), xử lý tiến độ, tính điểm, xếp loại.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho khóa học, mô- đun, học viên, tiến độ, chứng chỉ; import/export dữ liệu.
C	Thiết kế CLI menu, đăng ký khóa học, cập nhật tiến độ, chấm quiz, cấp chứng chỉ, thống kê, xuất báo cáo.

Yêu cầu nâng cao:

- Xuất chứng chỉ PDF tự động (tùy chọn).
- Tìm kiếm khóa học nâng cao (lọc theo giảng viên, loại mô-đun, thời lượng).
- Xuất báo cáo tiến độ học tập ra CSV.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

5. Quản lý lịch thi C in phiếu báo danh

Đề tài 5 – Quản lý lịch thi s in phiếu báo danh

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ quản lý **lịch thi**, **phòng thi**, **giám thị**, **sinh viên** và **phiếu báo danh**.

Ứng dụng cho phép tạo lịch thi, phân bổ phòng, in phiếu báo danh và xuất các báo cáo liên quan.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý học phần**: thêm, sửa, xóa, tìm kiếm theo mã hoặc tên học phần.
- **Quản lý kỳ thi**:
  - Tạo các kỳ thi (giữa kỳ, cuối kỳ, bổ sung).
  - Quản lý thông tin về ngày thi, ca thi, thời gian bắt đầu và kết thúc.
- **Phân bổ phòng thi**:
  - Quản lý phòng học, sức chứa, trang thiết bị.
  - Tự động gán sinh viên vào phòng thi theo sức chứa tối đa.
- **Quản lý giám thị**:
  - Gán giám thị cho từng phòng thi.
  - Báo cáo phân công giám thị.
- **In phiếu báo danh**:
  - Tạo phiếu báo danh cho từng sinh viên gồm: thông tin cá nhân, học phần, thời gian, phòng thi.
  - Lưu phiếu báo danh ra tệp CSV.
- **Kiểm tra trùng lịch**:
  - Cảnh báo khi sinh viên bị xếp 2 môn thi cùng thời điểm.
- **Thống kê s báo cáo**:
  - Danh sách phòng thi, số sinh viên, giám thị.
  - Danh sách sinh viên dự thi theo phòng.
  - Thống kê số phòng, số ca thi, số học phần.
- **Lưu trữ dữ liệu**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Exam *(abstract)* → MidtermExam, FinalExam, MakeupExam
    - getDuration() **override** tùy loại kỳ thi.
  - Room *(abstract)* → TheoryRoom, LabRoom
    - getCapacity() **override** theo từng loại phòng.
  - Person *(abstract)* → Student, Invigilator (giám thị).
- **Interface**:
  - Persistable (load/save dữ liệu).
  - Printable (in phiếu báo danh).
- **Ngoại lệ tùy biến**:
  - RoomFullException
  - ScheduleConflictException
  - ExamNotFoundException
  - StudentNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:

- courses.csv — thông tin học phần.
- exams.csv — thông tin kỳ thi.
- rooms.csv — danh sách phòng thi.
- students.csv — thông tin sinh viên.
- invigilators.csv — thông tin giám thị.
- assignments.csv — phân công phòng thi, sinh viên, giám thị.
- admission_tickets.csv — phiếu báo danh sinh viên.

- **Yêu cầu lưu trữ**:
  - Mã phòng, mã kỳ thi, mã phiếu báo danh sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Đảm bảo đồng bộ dữ liệu giữa các tệp.

---

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 6 học phần.
  - ≥ 3 kỳ thi (giữa kỳ, cuối kỳ, bổ sung).
  - ≥ 5 phòng thi.
  - ≥ 10 giám thị.
  - ≥ 30 sinh viên.
- **Báo cáo mẫu**:
  - Danh sách sinh viên trong từng phòng thi.
  - Danh sách phòng thi, sức chứa, số lượng sinh viên dự thi.
  - Danh sách giám thị phân công.
- **Test case bắt buộc** (≥ 10):

1. Tạo kỳ thi mới thành công.
2. Thêm phòng thi mới và kiểm tra sức chứa.
3. Phân bổ sinh viên vào phòng thi → không vượt sức chứa.
4. Cảnh báo khi sinh viên trùng lịch thi.
5. Gán giám thị cho phòng thi → không vượt số giám thị tối đa.
6. In phiếu báo danh chính xác.
7. Xuất danh sách sinh viên dự thi ra CSV.
8. Xuất báo cáo phân công giám thị.
9. Tìm kiếm sinh viên theo tên hoặc mã sinh viên.
10. Thống kê số phòng thi và số sinh viên dự thi.

---

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Exam, MidtermExam, FinalExam, Room, TheoryRoom, LabRoom, Person, Student, Invigilator), xử lý logic phân bổ sinh viên, kiểm tra trùng lịch.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho kỳ thi, phòng, sinh viên, giám thị, phiếu báo danh; import/export dữ liệu.
C	Thiết kế CLI menu, tạo lịch thi, phân bổ phòng, in phiếu báo danh, thống kê và xuất báo cáo.

Yêu cầu nâng cao:

- Xuất phiếu báo danh ra PDF (tùy chọn).
- Sắp xếp sinh viên tự động vào phòng tối ưu theo sức chứa.
- Xuất báo cáo phân công phòng thi ra CSV.

---

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).

- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

Nhóm 2 – Kinh doanh s Dịch vụ

6. Quản lý thuê xe (ô tô, xe máy, xe tải)

Đề tài 6 – Quản lý thuê xe (ô tô, xe máy, xe tải)

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý **hoạt động cho thuê xe** bao gồm ô tô, xe máy, xe tải.

Ứng dụng cho phép quản lý danh mục xe, khách hàng, hợp đồng thuê, trả xe, tính phí thuê, phí trễ hạn và xuất báo cáo.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

---

2. Chức năng chính

- **Quản lý xe**:
  - Thêm, sửa, xóa thông tin xe (biển số, loại xe, giá thuê, tình trạng).
  - Tìm kiếm xe theo loại, biển số, tình trạng.
- **Quản lý khách hàng**:
  - Thêm, sửa, xóa thông tin khách hàng.
  - Tìm kiếm khách hàng theo tên, số điện thoại hoặc số CMND/CCCD.
- **Quản lý hợp đồng thuê xe**:
  - Tạo hợp đồng thuê mới, ghi nhận thông tin khách hàng, loại xe, ngày bắt đầu và dự kiến trả.
  - Kiểm tra tình trạng xe: chỉ cho thuê xe **đang rảnh**.
  - Cập nhật trạng thái xe khi xe được thuê/trả.
- **Trả xe s tính phí**:
  - Tính tổng chi phí thuê dựa trên số ngày thuê, loại xe.
  - Tính phí phạt trả trễ (nếu có).
- **Báo cáo s thống kê**:
  - Danh sách xe đang cho thuê, xe còn trống.
  - Tổng doanh thu theo tháng/quý.
  - Top khách hàng thuê xe nhiều nhất.
  - Thống kê số lượng xe theo loại (ô tô, xe máy, xe tải).
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

---

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Vehicle *(abstract)* → Car, Motorbike, Truck
    - getDailyPrice() và getLateFeePerHour() **override** cho từng loại xe.
  - Person *(abstract)* → Customer.
- **Interface**:
  - Rentable (thuê/trả xe).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - VehicleNotAvailableException
  - ContractNotFoundException
  - InvalidReturnDateException
  - CustomerNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - vehicles.csv — thông tin xe (biển số, loại xe, giá thuê, tình trạng).
  - customers.csv — thông tin khách hàng.
  - contracts.csv — hợp đồng thuê xe.
  - returns.csv — thông tin trả xe và phí trễ (nếu có).
  - revenue.csv — doanh thu từng tháng/quý.
- **Yêu cầu lưu trữ**:
  - Mã xe, mã hợp đồng sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd.
  - Trạng thái xe: AVAILABLE, RENTED, MAINTENANCE.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 15 xe (ô tô, xe máy, xe tải).
  - ≥ 10 khách hàng.
  - ≥ 10 hợp đồng thuê xe.
  - ≥ 5 lượt trả xe trễ hạn.
- **Báo cáo mẫu**:
  - Danh sách xe đang được thuê.
  - Tổng doanh thu trong tháng hiện tại.
  - Top 3 khách hàng thuê nhiều xe nhất.
- **Test case bắt buộc** (≥ 10):

1. Tạo hợp đồng thuê xe thành công.
2. Cho thuê xe đang rảnh → cập nhật trạng thái RENTED.
3. Cho thuê xe đã được thuê → báo lỗi.
4. Tính phí thuê đúng cho loại xe ô tô.
5. Tính phí thuê đúng cho loại xe tải.
6. Trả xe đúng hạn → không có phí trễ.
7. Trả xe trễ hạn → tính phí phạt chính xác.
8. Thống kê doanh thu theo tháng.
9. Báo cáo top khách hàng thuê xe nhiều nhất.
10. Tìm kiếm xe theo biển số.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên chính	Nhiệm vụ
A	Thiết kế domain C kế thừa (Vehicle, Car, Motorbike, Truck, Person, Customer), xử lý logic thuê/trả, tính phí, phí trễ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho xe, khách hàng, hợp đồng, trả xe; import/export dữ liệu.
C	Thiết kế CLI menu, tạo hợp đồng thuê, trả xe, tính phí, xuất báo cáo thống kê.

Yêu cầu nâng cao:

- Tìm kiếm xe nâng cao: lọc theo loại xe, giá thuê, tình trạng.
- Xuất hợp đồng thuê ra file PDF/CSV.
- Thống kê doanh thu theo quý và vẽ biểu đồ ASCII trong console.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.

- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

7. Bán vé rạp phim C đặt chỗ ghế

Đề tài 7 – Quản lý bán vé rạp phim s đặt chỗ ghế

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý rạp phim**, bao gồm **lịch chiếu**, **đặt vé**, **chọn ghế** và **thanh toán**.

Ứng dụng cho phép quản lý phim, suất chiếu, phòng chiếu, ghế ngồi, khách hàng và hóa đơn thanh toán.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý phim**:
  - Thêm, sửa, xóa phim.
  - Thông tin phim: tên, thể loại, thời lượng, độ tuổi, giá vé cơ bản.
- **Quản lý phòng chiếu s ghế**:
  - Thêm, sửa, xóa phòng chiếu.
  - Tạo sơ đồ ghế ngồi cho từng phòng.
  - Quản lý ghế theo loại: thường, VIP.
- **Quản lý suất chiếu**:
  - Tạo lịch chiếu phim theo ngày, giờ, phòng.
  - Tránh trùng lịch chiếu cùng phòng.
- **Đặt vé s chọn ghế**:
  - Khách hàng chọn phim, suất chiếu, ghế.
  - Tính tổng giá vé dựa trên loại ghế và giảm giá nếu có.
- **Thanh toán s hóa đơn**:
  - Xuất hóa đơn khi đặt vé thành công.
  - Lưu thông tin giao dịch để phục vụ thống kê.
- **Báo cáo s thống kê**:
  - Danh sách vé bán ra theo phim/suất chiếu.
  - Doanh thu theo ngày/tháng.
  - Top 3 phim có doanh thu cao nhất.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Seat *(abstract)* → StandardSeat, VipSeat
    - getSurcharge() **override** để tính phụ phí ghế VIP.
  - Person *(abstract)* → Customer.
- **Interface**:
  - DiscountPolicy (giảm giá cho thành viên, sinh viên, khuyến mãi).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - SeatAlreadyBookedException
  - ShowtimeConflictException
  - InvalidSeatException
  - CustomerNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:

- movies.csv — thông tin phim.
- rooms.csv — thông tin phòng chiếu.
- seats.csv — thông tin ghế từng phòng.
- showtimes.csv — lịch chiếu phim.
- customers.csv — thông tin khách hàng.
- tickets.csv — vé đã bán.
- invoices.csv — hóa đơn thanh toán.

- **Yêu cầu lưu trữ**:
  - Mã vé, mã hóa đơn, mã suất chiếu sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái ghế: AVAILABLE, BOOKED.

---

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 6 phim.
  - ≥ 3 phòng chiếu.
  - ≥ 60 ghế/phòng (ghế thường + ghế VIP).
  - ≥ 8 suất chiếu/ngày.
  - ≥ 30 khách hàng.
- **Báo cáo mẫu**:
  - Danh sách ghế đã đặt cho một suất chiếu cụ thể.
  - Tổng số vé bán ra theo từng phim.
  - Doanh thu theo tháng.
  - Top 3 phim có doanh thu cao nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm phim mới thành công.
2. Tạo suất chiếu hợp lệ → tránh trùng lịch phòng chiếu.
3. Đặt vé thành công khi ghế còn trống.
4. Đặt vé thất bại khi ghế đã được đặt → báo lỗi.
5. Tính giá vé chính xác khi chọn ghế VIP.
6. Tính tổng hóa đơn cho nhiều vé cùng suất chiếu.
7. Xuất danh sách vé đã đặt ra CSV.
8. Báo cáo doanh thu theo ngày.
9. Báo cáo top 3 phim doanh thu cao nhất.
10. Tìm kiếm suất chiếu theo tên phim.

---

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên chính	Nhiệm vụ
A	Thiết kế domain C kế thừa (Seat, StandardSeat, VipSeat, Person, Customer), xử lý logic đặt vé, tính giá vé, giảm giá.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho phim, suất chiếu, ghế, vé, hóa đơn; import/export dữ liệu.
C	Thiết kế CLI menu, đặt vé, thanh toán, thống kê, xuất báo cáo doanh thu và vé đã đặt.

Yêu cầu nâng cao:

- Giảm giá thành viên (loyalty), khuyến mãi theo ngày lễ.
- Xuất vé xem phim ra file PDF/CSV.
- Gợi ý suất chiếu còn ghế trống khi suất chiếu mong muốn đã hết vé.

---

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).

- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

8. Quản lý nhà hàng C đặt bàn trực tuyến

Đề tài 8 – Quản lý nhà hàng s đặt bàn trực tuyến

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý **hoạt động nhà hàng**, bao gồm **bàn ăn**, **thực đơn**, **đặt bàn trực tuyến**, **thanh toán** và **báo cáo doanh thu**.
Ứng dụng giúp quản lý thông tin khách hàng, bàn ăn, món ăn, đặt bàn, hóa đơn thanh toán và thống kê hoạt động nhà hàng.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý bàn ăn**:
  - Thêm, sửa, xóa bàn ăn.
  - Quản lý trạng thái bàn: trống, đã đặt, đang phục vụ.
- **Quản lý thực đơn**:
  - Thêm, sửa, xóa món ăn.
  - Quản lý thông tin món: tên, loại, giá, khuyến mãi.
  - Tìm kiếm món theo tên, loại.
- **Đặt bàn trực tuyến**:
  - Khách hàng chọn bàn, nhập thông tin đặt bàn (ngày, giờ, số lượng người).
  - Kiểm tra tình trạng bàn và tránh trùng lịch đặt.
- **Quản lý hóa đơn**:
  - Tạo hóa đơn khi khách dùng xong bữa.
  - Tính tổng chi phí, giảm giá nếu có.
  - Xuất hóa đơn ra tệp CSV.
- **Báo cáo s thống kê**:
  - Doanh thu theo ngày, tháng.
  - Top 5 món bán chạy nhất.
  - Số lượng bàn đặt trước mỗi ngày.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Table *(abstract)* → StandardTable, VipTable
    - getSurcharge() **override** tính phụ phí cho bàn VIP.
  - MenuItem *(abstract)* → Food, Drink.
  - Person *(abstract)* → Customer.
- **Interface**:
  - DiscountPolicy (áp dụng khuyến mãi cho món ăn hoặc hóa đơn).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - TableAlreadyBookedException
  - TableNotFoundException
  - MenuItemNotFoundException
  - CustomerNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - tables.csv — thông tin bàn ăn.
  - menu.csv — thông tin món ăn, đồ uống.
  - customers.csv — thông tin khách hàng.
  - bookings.csv — thông tin đặt bàn.
  - invoices.csv — hóa đơn thanh toán.
  - revenue.csv — doanh thu theo ngày/tháng.
- **Yêu cầu lưu trữ**:
  - Mã bàn, mã món ăn, mã hóa đơn, mã đặt bàn sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái bàn: AVAILABLE, BOOKED, OCCUPIED.

---

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 15 bàn ăn (bao gồm bàn VIP và bàn thường).
  - ≥ 20 món ăn, đồ uống.
  - ≥ 30 khách hàng.
  - ≥ 20 lượt đặt bàn.
  - ≥ 15 hóa đơn.
- **Báo cáo mẫu**:
  - Danh sách bàn đã đặt trong một ngày cụ thể.
  - Tổng doanh thu theo tháng.
  - Top 5 món ăn bán chạy nhất.
- **Test case bắt buộc** (≥ 10):

1. Đặt bàn thành công khi bàn trống.
2. Đặt bàn thất bại khi bàn đã có lịch đặt → báo lỗi.
3. Hủy đặt bàn thành công → trạng thái bàn cập nhật về AVAILABLE.
4. Thêm món ăn mới vào thực đơn thành công.
5. Tìm kiếm món ăn theo tên.
6. Tính tổng hóa đơn chính xác khi có giảm giá.
7. Xuất hóa đơn ra file CSV.
8. Báo cáo doanh thu theo ngày.
9. Báo cáo top 5 món bán chạy nhất.
10. Thống kê số bàn đã đặt trong một ngày cụ thể.

---

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Table, StandardTable, VipTable, MenuItem, Food, Drink, Person, Customer), xử lý logic đặt bàn, tính phụ phí, tính hóa đơn.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho bàn, món ăn, khách hàng, hóa đơn; import/export dữ liệu.
C	Thiết kế CLI menu, đặt bàn, thanh toán, báo cáo doanh thu, xuất hóa đơn, thống kê món bán chạy.

Yêu cầu nâng cao:

- Gợi ý bàn trống dựa trên ngày/giờ mong muốn.
- Xuất hóa đơn ra PDF/CSV.
- Quản lý chương trình khuyến mãi riêng cho khách VIP.

---

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.

- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

9. Bán vé máy bay nội địa

Đề tài 9 – Quản lý bán vé máy bay nội địa

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý đặt vé máy bay nội địa**, bao gồm **chuyến bay**, **máy bay**, **ghế**, **khách hàng**, **vé**, **hóa đơn** và **báo cáo doanh thu**. Ứng dụng cho phép tra cứu chuyến bay, đặt vé, chọn ghế, hủy vé, thanh toán và xuất báo cáo.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý máy bay**:
  - Thêm, sửa, xóa thông tin máy bay.
  - Thông tin: mã máy bay, số ghế, phân loại ghế (phổ thông, thương gia, hạng nhất).
- **Quản lý chuyến bay**:
  - Tạo, sửa, xóa chuyến bay.
  - Quản lý thông tin: mã chuyến bay, điểm đi, điểm đến, giờ khởi hành, giờ hạ cánh, giá cơ bản.
  - Kiểm tra trùng lịch giữa các chuyến sử dụng cùng một máy bay.
- **Quản lý ghế**:
  - Quản lý trạng thái ghế: trống, đã đặt.
  - Phân biệt loại ghế để tính phụ phí.
- **Đặt vé s chọn ghế**:
  - Khách hàng chọn chuyến bay, chọn ghế, nhập thông tin.
  - Tính tổng giá vé = giá cơ bản + phụ phí ghế (nếu có).
- **Quản lý khách hàng**:
  - Thêm, sửa, xóa thông tin khách hàng.
  - Tìm kiếm khách hàng theo tên, số điện thoại, email.
- **Hủy vé s hoàn tiền**:
  - Hủy vé trước giờ bay ≥ 48h: hoàn 90%.
  - Hủy vé trước giờ bay < 48h: hoàn 50%.
  - Sau khi cất cánh: không hoàn.
- **Báo cáo s thống kê**:
  - Doanh thu theo ngày, tháng.
  - Tỷ lệ ghế trống/đã đặt theo từng chuyến.
  - Top 3 đường bay doanh thu cao nhất.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Seat *(abstract)* → EconomySeat, BusinessSeat, FirstClassSeat
    - getSurcharge() **override** cho từng loại ghế.
  - Flight *(abstract)* → DomesticFlight.
  - Person *(abstract)* → Customer.
- **Interface**:
  - RefundPolicy (tính số tiền hoàn trả khi hủy vé).
  - Persistable (load/save dữ liệu).

- **Ngoại lệ tùy biến**:
  - SeatAlreadyBookedException
  - FlightNotFoundException
  - InvalidCancellationException
  - CustomerNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - planes.csv — thông tin máy bay.
  - flights.csv — thông tin chuyến bay.
  - seats.csv — thông tin ghế trên từng chuyến.
  - customers.csv — thông tin khách hàng.
  - tickets.csv — vé đã bán.
  - invoices.csv — hóa đơn thanh toán.
  - revenue.csv — doanh thu theo ngày/tháng.
- **Yêu cầu lưu trữ**:
  - Mã vé, mã hóa đơn, mã chuyến bay, mã máy bay sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái ghế: AVAILABLE, BOOKED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 5 máy bay.
  - ≥ 10 chuyến bay.
  - ≥ 150 ghế (phân loại Economy, Business, First Class).
  - ≥ 30 khách hàng.
  - ≥ 20 vé đã bán.
- **Báo cáo mẫu**:
  - Danh sách ghế trống trên một chuyến bay.
  - Tổng doanh thu theo tháng.
  - Top 3 đường bay doanh thu cao nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm máy bay mới thành công.
2. Tạo chuyến bay thành công, không trùng lịch cùng máy bay.
3. Đặt vé thành công khi ghế còn trống.
4. Đặt vé thất bại khi ghế đã được đặt → báo lỗi.
5. Tính tổng giá vé chính xác với ghế hạng thương gia.
6. Hủy vé thành công và tính tiền hoàn đúng chính sách.
7. Xuất danh sách vé đã đặt ra CSV.
8. Báo cáo tỷ lệ ghế trống/đã đặt cho chuyến bay cụ thể.
9. Báo cáo doanh thu theo tháng.
10. Thống kê top 3 đường bay doanh thu cao nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Seat, EconomySeat, BusinessSeat, FirstClassSeat, Flight, DomesticFlight, Person, Customer), xử lý logic đặt vé, tính giá vé, hoàn tiền.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho máy bay, chuyến bay, ghế, vé, hóa đơn; import/export dữ liệu.

C Thiết kế CLI menu, đặt vé, chọn ghế, hủy vé, thống kê doanh thu, xuất báo cáo vé và chuyến bay.

Yêu cầu nâng cao:

- Gợi ý chuyến bay còn ghế trống theo ngày/điểm đến.
- Xuất vé máy bay ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh với các quý trước.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

10. Quản lý cửa hàng điện thoại C bảo hành

Đề tài 10 – Quản lý cửa hàng điện thoại s bảo hành

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý **cửa hàng bán điện thoại** bao gồm **sản phẩm**, **khách hàng**, **hóa đơn**, **phiếu bảo hành**, **trung tâm bảo hành** và **thống kê doanh thu**.

Ứng dụng cho phép quản lý bán hàng, bảo hành sản phẩm, lập hóa đơn, xuất báo cáo doanh thu và bảo hành.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý sản phẩm**:
  - Thêm, sửa, xóa thông tin điện thoại.
  - Quản lý tồn kho, số lượng nhập/xuất.
  - Tìm kiếm sản phẩm theo mã, tên, thương hiệu.
- **Quản lý khách hàng**:
  - Thêm, sửa, xóa thông tin khách hàng.
  - Tìm kiếm khách hàng theo tên, số điện thoại, email.
- **Bán hàng s hóa đơn**:
  - Tạo hóa đơn khi khách mua điện thoại.
  - Tính tổng giá trị hóa đơn, áp dụng khuyến mãi nếu có.
  - Xuất hóa đơn ra CSV.
- **Quản lý bảo hành**:
  - Cấp phiếu bảo hành cho sản phẩm.
  - Quản lý yêu cầu bảo hành: ghi nhận lỗi, thời gian tiếp nhận, trả bảo hành.
  - Kiểm tra thời hạn bảo hành.
- **Báo cáo s thống kê**:
  - Doanh thu theo ngày, tháng, quý.
  - Top 5 mẫu điện thoại bán chạy nhất.
  - Danh sách sản phẩm đang trong bảo hành.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Product *(abstract)* → Smartphone, FeaturePhone

        - getWarrantyPeriod() **override** để tính thời gian bảo hành theo loại sản phẩm.
    - Person *(abstract)* → Customer.
- **Interface**:
    - DiscountPolicy (áp dụng giảm giá theo chương trình khuyến mãi).
    - WarrantyService (quản lý thông tin bảo hành).
    - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
    - OutOfStockException
    - ProductNotFoundException
    - WarrantyExpiredException
    - CustomerNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
    - products.csv — thông tin sản phẩm.
    - customers.csv — thông tin khách hàng.
    - invoices.csv — hóa đơn bán hàng.
    - warranties.csv — thông tin phiếu bảo hành.
    - repairs.csv — thông tin sửa chữa bảo hành.
    - revenue.csv — doanh thu theo ngày/tháng/quý.
- **Yêu cầu lưu trữ**:
    - Mã sản phẩm, mã hóa đơn, mã phiếu bảo hành sinh tự động hoặc UUID.
    - Định dạng ngày yyyy-MM-dd.
    - Trạng thái bảo hành: ACTIVE, EXPIRED, IN_PROCESS.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
    - ≥ 20 mẫu điện thoại (smartphone + feature phone).
    - ≥ 30 khách hàng.
    - ≥ 15 hóa đơn.
    - ≥ 10 phiếu bảo hành.
- **Báo cáo mẫu**:
    - Danh sách sản phẩm bán ra trong tháng.
    - Top 5 điện thoại bán chạy nhất.
    - Danh sách sản phẩm đang bảo hành.
- **Test case bắt buộc** (≥ 10):

1. Thêm sản phẩm mới thành công.
2. Bán sản phẩm thành công khi còn tồn kho.
3. Bán sản phẩm thất bại khi hết hàng → báo lỗi.
4. Tạo hóa đơn bán hàng chính xác.
5. Cấp phiếu bảo hành cho sản phẩm vừa bán.
6. Kiểm tra tình trạng bảo hành: còn hạn → hợp lệ.
7. Kiểm tra tình trạng bảo hành: hết hạn → báo lỗi.
8. Xuất hóa đơn ra CSV.
9. Báo cáo doanh thu theo tháng.
10. Thống kê top 5 sản phẩm bán chạy nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (Product, Smartphone, FeaturePhone, Person, Customer), xử lý logic bán hàng, tính khuyến mãi, quản lý bảo hành.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho sản phẩm, khách hàng, hóa đơn, bảo hành; import/export dữ liệu.

C Thiết kế CLI menu, nhập hàng, bán hàng, tạo hóa đơn, tra cứu bảo hành, thống kê doanh thu.

Yêu cầu nâng cao:

- Tìm kiếm nâng cao: lọc sản phẩm theo thương hiệu, khoảng giá, thời gian bảo hành.
- Xuất hóa đơn ra PDF/CSV.
- Thống kê doanh thu theo quý và vẽ biểu đồ ASCII trong console.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

Nhóm 3 – Quản lý Đời sống

11. Quản lý hồ sơ bệnh nhân C lịch khám

Đề tài 11 – Quản lý hồ sơ bệnh nhân s lịch khám

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý hồ sơ bệnh nhân**, **lịch khám**, **bác sĩ**, **khoa khám bệnh** và **hóa đơn dịch vụ y tế**.
Ứng dụng cho phép đặt lịch khám, quản lý hồ sơ bệnh án, thanh toán dịch vụ, thống kê doanh thu và tình trạng bệnh nhân.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý hồ sơ bệnh nhân**:
  - Thêm, sửa, xóa hồ sơ bệnh nhân.
  - Lưu thông tin cá nhân, lịch sử khám, bệnh án.
  - Tìm kiếm bệnh nhân theo tên, số điện thoại, mã hồ sơ.
- **Quản lý bác sĩ s khoa khám**:
  - Quản lý thông tin bác sĩ: chuyên khoa, lịch trực, thông tin liên hệ.
  - Quản lý khoa khám bệnh: nội, ngoại, sản, nhi, tim mạch, v.v.
- **Đặt lịch khám**:
  - Bệnh nhân chọn ngày, giờ, bác sĩ hoặc chuyên khoa.
  - Kiểm tra trùng lịch giữa các cuộc hẹn.
- **Khám bệnh s lưu bệnh án**:
  - Ghi nhận chẩn đoán, toa thuốc, chỉ định xét nghiệm.
  - Cập nhật hồ sơ bệnh nhân tự động.
- **Quản lý dịch vụ s hóa đơn**:
  - Thêm, sửa, xóa dịch vụ y tế: khám bệnh, xét nghiệm, phẫu thuật.
  - Tính chi phí khám chữa bệnh, in hóa đơn thanh toán.
- **Báo cáo s thống kê**:
  - Doanh thu theo ngày, tháng, khoa.
  - Số lượt khám theo bác sĩ.
  - Danh sách bệnh nhân khám định kỳ hoặc điều trị dài hạn.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Person *(abstract)* → Patient, Doctor.
  - MedicalService *(abstract)* → ExaminationService, SurgeryService, TestService
    - getCost() **override** để tính chi phí từng loại dịch vụ.
- **Interface**:
  - Schedulable (quản lý lịch hẹn).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - AppointmentConflictException
  - DoctorNotFoundException
  - PatientNotFoundException
  - ServiceNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - patients.csv — thông tin bệnh nhân.
  - doctors.csv — thông tin bác sĩ.
  - departments.csv — danh sách khoa khám bệnh.
  - appointments.csv — lịch khám bệnh.
  - medical_records.csv — hồ sơ bệnh án.
  - services.csv — danh sách dịch vụ y tế.
  - invoices.csv — hóa đơn thanh toán.
  - revenue.csv — doanh thu theo ngày/tháng/khoa.
- **Yêu cầu lưu trữ**:
  - Mã hồ sơ, mã bác sĩ, mã dịch vụ, mã hóa đơn sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái cuộc hẹn: SCHEDULED, COMPLETED, CANCELLED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 20 bệnh nhân.
  - ≥ 10 bác sĩ.
  - ≥ 5 khoa khám.
  - ≥ 15 dịch vụ y tế.
  - ≥ 30 lịch khám bệnh.
- **Báo cáo mẫu**:
  - Danh sách bệnh nhân khám trong một ngày cụ thể.
  - Doanh thu của từng khoa trong tháng.
  - Top 3 bác sĩ có nhiều lịch khám nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm bệnh nhân mới thành công.
2. Đặt lịch khám thành công khi bác sĩ còn trống lịch.
3. Đặt lịch khám thất bại khi bác sĩ đã kín lịch → báo lỗi.
4. Ghi nhận bệnh án sau khi khám xong.
5. Thêm dịch vụ xét nghiệm mới thành công.
6. Tính chi phí khám và xét nghiệm chính xác.
7. Xuất hóa đơn khám bệnh ra CSV.
8. Báo cáo doanh thu theo ngày.
9. Thống kê số lượt khám theo bác sĩ.
10. Danh sách bệnh nhân điều trị dài hạn.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên chính	Nhiệm vụ
A	Thiết kế domain C kế thừa (Person, Patient, Doctor, MedicalService, ExaminationService, SurgeryService, TestService), xử lý logic đặt lịch khám, tính chi phí dịch vụ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho bệnh nhân, bác sĩ, dịch vụ, lịch khám, hồ sơ bệnh án, hóa đơn; import/export dữ liệu.
C	Thiết kế CLI menu, đặt lịch khám, ghi bệnh án, tạo hóa đơn, thống kê doanh thu, xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý bác sĩ trống lịch theo chuyên khoa.
- Xuất toa thuốc hoặc bệnh án ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh với kỳ trước.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

12. Quản lý phòng gym C gói tập

Đề tài 12 – Quản lý phòng gym s gói tập

1. Mục tiêu

Xây dựng một **Java Console Application** quản lý **phòng gym**, bao gồm **gói tập**, **hội viên**, **huấn luyện viên**, **điểm danh**, **lịch tập**, **thanh toán** và **thống kê doanh thu**. Ứng dụng cho phép hội viên đăng ký gói tập, đặt lịch tập với huấn luyện viên, ghi nhận điểm danh và quản lý thông tin thanh toán.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý hội viên**:
  - Thêm, sửa, xóa hội viên.
  - Tìm kiếm hội viên theo tên, số điện thoại, email.
- **Quản lý huấn luyện viên (PT)**:
  - Thêm, sửa, xóa huấn luyện viên.
  - Quản lý thông tin PT: tên, trình độ, lịch rảnh.
- **Quản lý gói tập**:
  - Tạo các gói tập: gói tháng, gói 10 buổi, gói không giới hạn.
  - Thông tin gói: tên, giá, thời hạn, số buổi tối đa.
- **Đăng ký gói tập s thanh toán**:
  - Hội viên đăng ký gói tập.
  - Tính toán chi phí và ghi nhận thanh toán.
- **Quản lý lịch tập s điểm danh**:
  - Đặt lịch tập với PT.
  - Ghi nhận điểm danh tự động theo lịch hoặc thủ công.
  - Cảnh báo khi hội viên hết buổi hoặc hết hạn gói tập.
- **Báo cáo s thống kê**:

- Doanh thu theo tháng/quý.
- Danh sách hội viên sắp hết hạn hoặc hết buổi tập.
- Top PT có nhiều lịch tập nhất.

- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Membership *(abstract)* → MonthlyPass, SessionPack, UnlimitedPass
    - isActive() và getRemainingSessions() **override** cho từng loại gói tập.
  - Person *(abstract)* → Member, Trainer.
- **Interface**:
  - Schedulable (quản lý lịch tập).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - PackageExpiredException
  - OutOfSessionsException
  - TrainerNotFoundException
  - MemberNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - members.csv — thông tin hội viên.
  - trainers.csv — thông tin huấn luyện viên.
  - packages.csv — thông tin gói tập.
  - registrations.csv — thông tin đăng ký gói tập.
  - schedules.csv — lịch tập cá nhân hoặc với PT.
  - checkins.csv — thông tin điểm danh.
  - payments.csv — thanh toán gói tập.
  - revenue.csv — doanh thu theo tháng/quý.
- **Yêu cầu lưu trữ**:
  - Mã hội viên, mã gói tập, mã thanh toán, mã lịch tập sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái gói tập: ACTIVE, EXPIRED, FULL.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 15 hội viên.
  - ≥ 5 huấn luyện viên.
  - ≥ 4 gói tập.
  - ≥ 20 lịch tập.
  - ≥ 10 thanh toán gói tập.
- **Báo cáo mẫu**:
  - Danh sách hội viên còn ≤ 3 buổi tập.
  - Danh sách hội viên sắp hết hạn gói tập.
  - Doanh thu theo tháng.
  - Top 3 PT có nhiều lịch tập nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm hội viên mới thành công.
2. Đăng ký gói tập thành công khi còn chỗ.
3. Đăng ký gói tập thất bại khi đã hết hạn → báo lỗi.
4. Điểm danh buổi tập thành công khi còn buổi.

5. Điểm danh thất bại khi hết buổi → báo lỗi.
6. Đặt lịch tập với PT thành công.
7. Đặt lịch tập thất bại khi PT bận → báo lỗi.
8. Báo cáo doanh thu theo tháng.
9. Báo cáo danh sách hội viên sắp hết hạn.
10. Thống kê top 3 PT có nhiều lịch tập nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Membership, MonthlyPass, SessionPack, UnlimitedPass, Person, Member, Trainer), xử lý logic gói tập, điểm danh, lịch tập.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho hội viên, PT, gói tập, lịch tập, điểm danh; import/export dữ liệu.
C	Thiết kế CLI menu, đăng ký gói tập, điểm danh, quản lý lịch tập, thống kê doanh thu, xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý gói tập phù hợp dựa trên lịch sử đăng ký.
- Xuất hóa đơn thanh toán ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh với các kỳ trước.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

13. Quản lý sự kiện C đăng ký tham dự

Đề tài 13 – Quản lý sự kiện s đăng ký tham dự

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý sự kiện** bao gồm **tạo sự kiện**, **đăng ký tham dự**, **quản lý vé**, **khách mời**, **thanh toán** và **thống kê**.
Ứng dụng giúp đơn vị tổ chức sự kiện quản lý thông tin chi tiết sự kiện, khách mời, vé, doanh thu và báo cáo thống kê.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý sự kiện**:
  - Tạo, sửa, xóa thông tin sự kiện.
  - Thông tin: tên sự kiện, địa điểm, thời gian, mô tả, số lượng vé tối đa.
- **Quản lý vé**:
  - Phân loại vé: thường, VIP, ưu đãi.
  - Quản lý giá vé, số lượng còn lại.
  - Kiểm tra vé còn trống trước khi đặt.
- **Quản lý khách tham dự**:
  - Thêm, sửa, xóa thông tin khách tham dự.
  - Tìm kiếm khách theo tên, email, số điện thoại.
- **Đăng ký tham dự s thanh toán**:

- Khách hàng chọn sự kiện, chọn loại vé.
- Thanh toán và xuất vé.

- **Hủy đăng ký s hoàn tiền**:
  - Hủy vé trước 7 ngày: hoàn 90%.
  - Hủy vé từ 3 đến 7 ngày: hoàn 50%.
  - Hủy vé trong vòng 3 ngày: không hoàn.
- **Báo cáo s thống kê**:
  - Số vé đã bán theo sự kiện.
  - Doanh thu theo sự kiện/tháng.
  - Top sự kiện có doanh thu cao nhất.
- **Lưu trữ dữ liệu bền vững**: tự động đọc/ghi tệp khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Ticket *(abstract)* → StandardTicket, VipTicket, DiscountTicket
    - getPrice() **override** để tính giá tùy loại vé.
  - Person *(abstract)* → Attendee.
  - Event *(abstract)* → MusicEvent, ConferenceEvent, WorkshopEvent.
- **Interface**:
  - RefundPolicy (xử lý hoàn tiền khi hủy vé).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - TicketSoldOutException
  - EventNotFoundException
  - AttendeeNotFoundException
  - InvalidCancellationException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - events.csv — thông tin sự kiện.
  - tickets.csv — thông tin vé.
  - attendees.csv — thông tin khách tham dự.
  - bookings.csv — thông tin đăng ký tham dự.
  - payments.csv — thông tin thanh toán.
  - revenue.csv — doanh thu theo sự kiện/tháng.
- **Yêu cầu lưu trữ**:
  - Mã sự kiện, mã vé, mã đặt chỗ sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái vé: AVAILABLE, BOOKED, CANCELLED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 5 sự kiện.
  - ≥ 3 loại vé cho mỗi sự kiện.
  - ≥ 30 khách tham dự.
  - ≥ 20 vé đã bán.
- **Báo cáo mẫu**:
  - Danh sách khách tham dự một sự kiện cụ thể.
  - Số vé đã bán của từng sự kiện.
  - Top 3 sự kiện có doanh thu cao nhất.
- **Test case bắt buộc** (≥ 10):

1. Tạo sự kiện mới thành công.
2. Thêm loại vé mới thành công.

3. Đặt vé thành công khi còn vé.
4. Đặt vé thất bại khi hết vé → báo lỗi.
5. Hủy vé hợp lệ và tính tiền hoàn chính xác.
6. Hủy vé muộn → không hoàn tiền.
7. Xuất vé ra file CSV.
8. Báo cáo số vé đã bán của sự kiện cụ thể.
9. Thống kê doanh thu theo tháng.
10. Thống kê top 3 sự kiện doanh thu cao nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Ticket, StandardTicket, VipTicket, DiscountTicket, Person, Attendee, Event), xử lý logic đặt vé, tính giá vé, hoàn tiền.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho sự kiện, vé, khách tham dự, đăng ký, thanh toán; import/export dữ liệu.
C	Thiết kế CLI menu, đăng ký tham dự, thanh toán, hủy vé, thống kê doanh thu, xuất báo cáo vé và sự kiện.

Yêu cầu nâng cao:

- Gợi ý sự kiện sắp diễn ra dựa trên ngày hiện tại.
- Xuất vé điện tử ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh với các kỳ trước.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

14. Quản lý chi tiêu cá nhân C nhóm

Đề tài 14 – Quản lý chi tiêu cá nhân s nhóm

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý chi tiêu cá nhân** và **chi tiêu nhóm**.

Ứng dụng cho phép người dùng ghi nhận các khoản thu – chi, phân loại chi tiêu, chia tiền nhóm, báo cáo thống kê tài chính và xuất dữ liệu ra file.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý tài khoản**:
  - Thêm, sửa, xóa tài khoản cá nhân hoặc nhóm.
  - Mỗi tài khoản có số dư ban đầu và số dư hiện tại.
- **Quản lý khoản thu – chi**:
  - Ghi nhận các khoản **thu**: lương, thưởng, chuyển tiền.
  - Ghi nhận các khoản **chi**: ăn uống, đi lại, giải trí, học tập, đầu tư.
  - Hỗ trợ gắn nhãn và phân loại khoản thu – chi.
- **Quản lý nhóm chi tiêu**:
  - Tạo nhóm chi tiêu, thêm thành viên.

- Ghi nhận chi tiêu chung và tính toán số tiền từng thành viên phải trả hoặc được nhận.
- **Báo cáo s thống kê**:
  - Báo cáo chi tiêu theo **ngày / tháng / năm**.
  - Thống kê tỷ lệ chi tiêu theo danh mục.
  - Xuất báo cáo tổng hợp số dư, khoản thu – chi.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Transaction *(abstract)* → Income, Expense
    - getAmount() **override** để xử lý giá trị thu hoặc chi.
  - Account *(abstract)* → PersonalAccount, GroupAccount.
  - Person *(abstract)* → User.
- **Interface**:
  - Reportable (tạo báo cáo chi tiêu).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - InsufficientBalanceException
  - AccountNotFoundException
  - TransactionNotFoundException
  - GroupMemberNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - accounts.csv — thông tin tài khoản cá nhân và nhóm.
  - transactions.csv — thông tin thu – chi.
  - groups.csv — thông tin nhóm chi tiêu.
  - members.csv — thông tin thành viên nhóm.
  - balances.csv — số dư tài khoản.
  - reports.csv — dữ liệu báo cáo.
- **Yêu cầu lưu trữ**:
  - Mã giao dịch, mã tài khoản, mã nhóm sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd.
  - Phân loại giao dịch: INCOME, EXPENSE.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 5 tài khoản cá nhân.
  - ≥ 2 nhóm chi tiêu.
  - ≥ 30 giao dịch thu – chi.
  - ≥ 10 thành viên nhóm.
- **Báo cáo mẫu**:
  - Báo cáo chi tiêu theo tháng.
  - Báo cáo tỷ lệ chi tiêu theo danh mục.
  - Số dư hiện tại của từng tài khoản.
- **Test case bắt buộc** (≥ 10):

1. Tạo tài khoản cá nhân mới thành công.
2. Ghi nhận khoản thu → số dư tăng chính xác.
3. Ghi nhận khoản chi → số dư giảm chính xác.
4. Thêm thành viên vào nhóm chi tiêu thành công.
5. Ghi nhận chi tiêu nhóm thành công.
6. Tính toán số tiền từng thành viên cần trả hoặc nhận chính xác.

7. Báo cáo tổng thu, tổng chi trong tháng.
8. Báo cáo tỷ lệ chi tiêu theo danh mục.
9. Xuất báo cáo chi tiêu ra CSV.
10. Tìm kiếm giao dịch theo từ khóa hoặc khoảng ngày.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Transaction, Income, Expense, Account, PersonalAccount, GroupAccount, Person, User), xử lý logic ghi nhận thu – chi, tính toán chia tiền nhóm.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho tài khoản, giao dịch, nhóm chi tiêu, báo cáo; import/export dữ liệu.
C	Thiết kế CLI menu, quản lý tài khoản, thu – chi, nhóm, xuất báo cáo, thống kê số dư.

Yêu cầu nâng cao:

- Gợi ý cách tiết kiệm dựa trên lịch sử chi tiêu.
- Xuất báo cáo chi tiêu ra PDF/CSV.
- Tích hợp phân tích chi tiêu bằng biểu đồ ASCII trong console.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

15. Quản lý khách sạn C đặt phòng

Đề tài 15 – Quản lý khách sạn s đặt phòng

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý khách sạn** bao gồm **phòng**, **khách hàng**, **đặt phòng**, **thanh toán**, **dịch vụ bổ sung** và **thống kê doanh thu**.
Ứng dụng cho phép khách hàng đặt phòng, sử dụng dịch vụ, thanh toán, quản lý tình trạng phòng và xuất báo cáo.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý phòng**:
  - Thêm, sửa, xóa phòng.
  - Quản lý thông tin: loại phòng, giá, sức chứa, trạng thái.
  - Phân loại phòng: thường, VIP, suite.
- **Quản lý khách hàng**:
  - Thêm, sửa, xóa thông tin khách hàng.
  - Tìm kiếm khách hàng theo tên, số điện thoại, email.
- **Đặt phòng s hủy phòng**:
  - Đặt phòng theo ngày nhận và trả.
  - Tính chi phí thuê dựa trên loại phòng và số ngày.
  - Hủy phòng theo chính sách hủy đặt.
- **Quản lý dịch vụ bổ sung**:
  - Đăng ký dịch vụ: ăn sáng, spa, thuê xe, giặt là.

- Tính phí dịch vụ cộng thêm vào hóa đơn.
- **Thanh toán s hóa đơn**:
  - Tạo hóa đơn chi tiết cho khách hàng.
  - Xuất hóa đơn ra CSV.
- **Báo cáo s thống kê**:
  - Danh sách phòng còn trống theo ngày.
  - Doanh thu theo ngày/tháng.
  - Top 3 khách hàng sử dụng dịch vụ nhiều nhất.
- **Lưu trữ dữ liệu bền vững**: tự động đọc/ghi tệp khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Room *(abstract)* → StandardRoom, VipRoom, SuiteRoom
    - getPrice() **override** để tính giá theo từng loại phòng.
  - Service *(abstract)* → BreakfastService, SpaService, CarRentalService, LaundryService.
  - Person *(abstract)* → Customer.
- **Interface**:
  - Billable (tính tổng chi phí phòng + dịch vụ).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - RoomNotAvailableException
  - BookingNotFoundException
  - CustomerNotFoundException
  - InvalidCancellationException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - rooms.csv — thông tin phòng.
  - customers.csv — thông tin khách hàng.
  - bookings.csv — thông tin đặt phòng.
  - services.csv — thông tin dịch vụ bổ sung.
  - payments.csv — thông tin thanh toán.
  - invoices.csv — hóa đơn chi tiết.
  - revenue.csv — doanh thu theo ngày/tháng.
- **Yêu cầu lưu trữ**:
  - Mã phòng, mã đặt phòng, mã hóa đơn, mã dịch vụ sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd.
  - Trạng thái phòng: AVAILABLE, BOOKED, OCCUPIED, MAINTENANCE.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 20 phòng (gồm thường, VIP, suite).
  - ≥ 15 khách hàng.
  - ≥ 10 dịch vụ bổ sung.
  - ≥ 20 lượt đặt phòng.
- **Báo cáo mẫu**:
  - Danh sách phòng còn trống vào một ngày cụ thể.
  - Doanh thu theo tháng.
  - Top 3 khách hàng dùng nhiều dịch vụ nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm phòng mới thành công.

2. Đặt phòng thành công khi còn phòng trống.
3. Đặt phòng thất bại khi phòng đã được đặt → báo lỗi.
4. Đăng ký dịch vụ bổ sung thành công.
5. Tính tổng chi phí phòng + dịch vụ chính xác.
6. Hủy phòng hợp lệ trước hạn → hoàn tiền một phần.
7. Hủy phòng muộn → không hoàn tiền.
8. Xuất hóa đơn ra CSV.
9. Báo cáo doanh thu theo tháng.
10. Thống kê top 3 khách hàng sử dụng nhiều dịch vụ nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Room, StandardRoom, VipRoom, SuiteRoom, Service, Person, Customer), xử lý logic đặt phòng, hủy phòng, tính phí dịch vụ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho phòng, khách hàng, dịch vụ, đặt phòng, hóa đơn; import/export dữ liệu.
C	Thiết kế CLI menu, đặt phòng, đăng ký dịch vụ, thanh toán, thống kê doanh thu, xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý phòng trống theo ngày và số lượng khách.
- Xuất hóa đơn ra file PDF/CSV.
- Thống kê doanh thu theo quý, so sánh các kỳ.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

Nhóm 4 – Ứng dụng Giải trí s Hệ thống

16. Quản lý câu lạc bộ thể thao

Đề tài 16 – Quản lý câu lạc bộ thể thao

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý hoạt động câu lạc bộ thể thao** bao gồm **thành viên**, **huấn luyện viên**, **môn thể thao**, **lịch tập**, **giải đấu** và **thống kê thành tích**.

Ứng dụng cho phép người dùng đăng ký thành viên, quản lý lịch tập, phân công huấn luyện viên, ghi nhận kết quả giải đấu và xuất báo cáo thống kê.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý thành viên**:
  - Thêm, sửa, xóa thành viên.
  - Tìm kiếm thành viên theo tên, mã thành viên, số điện thoại.
- **Quản lý huấn luyện viên**:
  - Thêm, sửa, xóa huấn luyện viên.
  - Quản lý chuyên môn và phân công môn thể thao.

- **Quản lý môn thể thao**:
  - Thêm, sửa, xóa môn thể thao.
  - Ghi nhận thông tin: tên môn, mô tả, huấn luyện viên phụ trách.
- **Quản lý lịch tập**:
  - Đặt lịch tập cho từng môn, nhóm hoặc cá nhân.
  - Tránh trùng lịch giữa các thành viên và huấn luyện viên.
- **Quản lý giải đấu**:
  - Tạo giải đấu mới, thêm thành viên tham gia.
  - Cập nhật kết quả, xếp hạng và thành tích.
- **Báo cáo s thống kê**:
  - Thống kê số thành viên tham gia từng môn thể thao.
  - Báo cáo thành tích giải đấu.
  - Top 5 thành viên đạt nhiều thành tích nhất.
- **Lưu trữ dữ liệu bền vững**: tự động đọc/ghi tệp khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Person *(abstract)* → Member, Coach.
  - Sport *(abstract)* → Football, Basketball, Tennis, Badminton…
    - getTrainingSchedule() **override** để trả về lịch tập từng môn.
  - Competition *(abstract)* → League, Tournament.
- **Interface**:
  - Schedulable (quản lý lịch tập và giải đấu).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - ScheduleConflictException
  - MemberNotFoundException
  - CoachNotFoundException
  - SportNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - members.csv — thông tin thành viên.
  - coaches.csv — thông tin huấn luyện viên.
  - sports.csv — thông tin môn thể thao.
  - schedules.csv — lịch tập.
  - competitions.csv — thông tin giải đấu.
  - results.csv — kết quả thi đấu.
  - achievements.csv — thành tích của thành viên.
- **Yêu cầu lưu trữ**:
  - Mã thành viên, mã huấn luyện viên, mã môn thể thao, mã giải đấu sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái lịch tập: SCHEDULED, COMPLETED, CANCELLED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 20 thành viên.
  - ≥ 5 huấn luyện viên.
  - ≥ 4 môn thể thao.
  - ≥ 10 lịch tập.
  - ≥ 3 giải đấu.
- **Báo cáo mẫu**:

- Danh sách thành viên tham gia từng môn thể thao.
- Kết quả và xếp hạng của giải đấu gần nhất.
- Top 5 thành viên đạt nhiều thành tích nhất.

- **Test case bắt buộc** (≥ 10):

1. Thêm thành viên mới thành công.
2. Thêm huấn luyện viên thành công.
3. Phân công huấn luyện viên cho môn thể thao.
4. Đặt lịch tập thành công khi không trùng lịch.
5. Đặt lịch tập thất bại khi huấn luyện viên bận → báo lỗi.
6. Tạo giải đấu mới thành công.
7. Thêm thành viên vào giải đấu thành công.
8. Cập nhật kết quả thi đấu chính xác.
9. Báo cáo thành tích giải đấu.
10. Thống kê top 5 thành viên đạt nhiều thành tích nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Person, Member, Coach, Sport, Competition), xử lý logic lịch tập, kết quả giải đấu, thống kê thành tích.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho thành viên, huấn luyện viên, môn thể thao, lịch tập, giải đấu; import/export dữ liệu.
C	Thiết kế CLI menu, quản lý thành viên, huấn luyện viên, đăng ký lịch tập, ghi nhận kết quả, thống kê thành tích.

Yêu cầu nâng cao:

- Gợi ý môn thể thao phù hợp cho thành viên dựa trên lịch sử tham gia.
- Xuất bảng thành tích giải đấu ra file PDF/CSV.
- Thống kê lịch sử tham gia giải đấu của từng thành viên.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

17. Quản lý trung tâm ngoại ngữ

Đề tài 17 – Quản lý trung tâm ngoại ngữ

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý trung tâm ngoại ngữ** bao gồm **khóa học**, **học viên**, **giáo viên**, **lịch học**, **kết quả thi** và **thanh toán học phí**.
Ứng dụng cho phép quản lý toàn bộ hoạt động của trung tâm từ đăng ký khóa học, xếp lớp, theo dõi điểm số cho đến thống kê doanh thu.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý học viên**:
  - Thêm, sửa, xóa học viên.
  - Tìm kiếm học viên theo tên, mã học viên, số điện thoại hoặc email.
- **Quản lý giáo viên**:

- Thêm, sửa, xóa thông tin giáo viên.
- Quản lý trình độ, ngôn ngữ giảng dạy và lịch rảnh.

- **Quản lý khóa học**:
  - Tạo, sửa, xóa khóa học.
  - Thông tin khóa học: tên, trình độ, ngôn ngữ, học phí, giáo viên phụ trách.
- **Quản lý lớp học s lịch học**:
  - Xếp học viên vào lớp theo trình độ.
  - Tạo lịch học cho từng lớp.
  - Tránh trùng lịch giữa các lớp, học viên và giáo viên.
- **Quản lý điểm thi s chứng chỉ**:
  - Nhập điểm thi định kỳ và cuối khóa.
  - Tính điểm tổng kết và xếp loại.
  - Cấp chứng chỉ cho học viên hoàn thành khóa học đạt yêu cầu.
- **Quản lý học phí**:
  - Ghi nhận thanh toán học phí.
  - Cảnh báo học viên chưa đóng học phí.
- **Báo cáo s thống kê**:
  - Thống kê doanh thu theo tháng/quý.
  - Danh sách học viên đạt chứng chỉ.
  - Top 3 giáo viên phụ trách nhiều khóa học nhất.
- **Lưu trữ dữ liệu bền vững**: tự động đọc/ghi tệp khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Person *(abstract)* → Student, Teacher.
  - Course *(abstract)* → EnglishCourse, JapaneseCourse, ChineseCourse, KoreanCourse…
    - calculateTuition() **override** để tính học phí từng khóa.
  - Certificate *(abstract)* → CompletionCertificate, ExcellenceCertificate.
- **Interface**:
  - Schedulable (quản lý lịch học).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - ScheduleConflictException
  - StudentNotFoundException
  - TeacherNotFoundException
  - CourseNotFoundException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - students.csv — thông tin học viên.
  - teachers.csv — thông tin giáo viên.
  - courses.csv — thông tin khóa học.
  - classes.csv — thông tin lớp học.
  - schedules.csv — lịch học.
  - scores.csv — điểm thi.
  - certificates.csv — chứng chỉ học viên.
  - payments.csv — thông tin học phí.
  - revenue.csv — doanh thu theo tháng/quý.
- **Yêu cầu lưu trữ**:
  - Mã học viên, mã khóa học, mã chứng chỉ, mã thanh toán sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.

- Trạng thái học phí: PAID, UNPAID.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 30 học viên.
  - ≥ 10 giáo viên.
  - ≥ 5 khóa học.
  - ≥ 10 lớp học.
  - ≥ 20 lịch học.
- **Báo cáo mẫu**:
  - Danh sách học viên hoàn thành khóa học.
  - Doanh thu theo từng tháng/quý.
  - Top 3 giáo viên phụ trách nhiều lớp nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm học viên mới thành công.
2. Thêm giáo viên mới thành công.
3. Tạo khóa học mới và phân công giáo viên.
4. Xếp học viên vào lớp phù hợp.
5. Đặt lịch học hợp lệ khi giáo viên còn trống.
6. Đặt lịch học thất bại khi giáo viên trùng lịch → báo lỗi.
7. Nhập điểm thi cuối khóa chính xác.
8. Cấp chứng chỉ cho học viên đủ điều kiện.
9. Báo cáo danh sách học viên chưa đóng học phí.
10. Thống kê doanh thu theo quý.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (Person, Student, Teacher, Course, Certificate), xử lý logic đăng ký khóa học, lịch học, tính học phí và cấp chứng chỉ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho học viên, giáo viên, khóa học, lịch học, chứng chỉ, học phí; import/export dữ liệu.
C	Thiết kế CLI menu, quản lý học viên, khóa học, thanh toán học phí, nhập điểm thi, thống kê doanh thu và xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý khóa học phù hợp dựa trên trình độ học viên.
- Xuất chứng chỉ hoàn thành ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh với các kỳ trước.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

18. Quản lý rạp chiếu phim C đặt vé xem phim

Đề tài 18 – Quản lý rạp chiếu phim s đặt vé xem phim

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý rạp chiếu phim**, bao gồm **phim**, **suất chiếu**, **phòng chiếu**, **ghế ngồi**, **đặt vé**, **hóa đơn** và **thống kê doanh thu**.

Ứng dụng cho phép khách đặt vé xem phim, chọn ghế, thanh toán và quản lý báo cáo doanh thu.
Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý phim**:
  - Thêm, sửa, xóa thông tin phim.
  - Lưu thông tin: tên phim, thể loại, thời lượng, độ tuổi giới hạn, mô tả.
- **Quản lý phòng chiếu**:
  - Quản lý thông tin: mã phòng, số ghế, sơ đồ ghế.
  - Phân loại phòng: thường, VIP, IMAX.
- **Quản lý suất chiếu**:
  - Tạo, sửa, xóa suất chiếu.
  - Liên kết phim, phòng chiếu, thời gian chiếu.
  - Kiểm tra trùng lịch chiếu.
- **Quản lý ghế**:
  - Trạng thái ghế: trống, đã đặt.
  - Phân loại ghế: thường, VIP, đôi.
- **Đặt vé xem phim**:
  - Chọn phim, suất chiếu, ghế ngồi.
  - Tự động tính tổng tiền vé.
- **Thanh toán s hóa đơn**:
  - Xuất hóa đơn vé xem phim.
  - Lưu thông tin thanh toán.
- **Báo cáo s thống kê**:
  - Doanh thu theo phim, suất chiếu, ngày/tháng.
  - Tỷ lệ lấp đầy phòng chiếu.
  - Top 3 phim doanh thu cao nhất.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Seat *(abstract)* → StandardSeat, VipSeat, CoupleSeat
    - getPrice() **override** theo loại ghế.
  - Movie *(abstract)* → FeatureFilm, Documentary, Animation.
- **Interface**:
  - Billable (tính giá vé và tạo hóa đơn).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - SeatAlreadyBookedException
  - MovieNotFoundException
  - ShowtimeConflictException
  - PaymentFailedException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - movies.csv — thông tin phim.
  - rooms.csv — thông tin phòng chiếu.
  - showtimes.csv — thông tin suất chiếu.
  - seats.csv — sơ đồ ghế từng phòng.
  - tickets.csv — thông tin vé.
  - payments.csv — thông tin thanh toán.

- revenue.csv — doanh thu theo phim, suất chiếu, ngày/tháng.
- **Yêu cầu lưu trữ**:
  - Mã phim, mã suất chiếu, mã vé, mã hóa đơn sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái ghế: AVAILABLE, BOOKED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 10 phim đang chiếu.
  - ≥ 5 phòng chiếu.
  - ≥ 20 suất chiếu.
  - ≥ 100 ghế (gồm ghế thường, VIP, đôi).
- **Báo cáo mẫu**:
  - Danh sách vé đã bán cho một suất chiếu cụ thể.
  - Doanh thu theo phim/tháng.
  - Top 3 phim doanh thu cao nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm phim mới thành công.
2. Tạo suất chiếu thành công.
3. Đặt vé thành công khi ghế còn trống.
4. Đặt vé thất bại khi ghế đã đặt → báo lỗi.
5. Tính giá vé chính xác theo loại ghế.
6. Xuất hóa đơn vé xem phim.
7. Báo cáo doanh thu theo tháng.
8. Báo cáo tỷ lệ ghế trống/đã đặt cho suất chiếu.
9. Thống kê doanh thu theo phim.
10. Thống kê top 3 phim doanh thu cao nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ
A	Thiết kế domain C kế thừa (Seat, StandardSeat, VipSeat, CoupleSeat, Movie, FeatureFilm, Documentary, Animation), xử lý logic đặt vé, tính giá vé, quản lý suất chiếu.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho phim, phòng chiếu, suất chiếu, vé, thanh toán; import/export dữ liệu.
C	Thiết kế CLI menu, đặt vé, chọn ghế, thanh toán, thống kê doanh thu, xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý phim sắp chiếu trong tuần.
- Xuất vé xem phim điện tử ra file PDF/CSV.
- Thống kê doanh thu theo quý và so sánh giữa các kỳ.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

19. Quản lý bãi giữ xe thông minh

Đề tài 19 – Quản lý bãi giữ xe thông minh

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý bãi giữ xe thông minh**, bao gồm **quản lý phương tiện**, **thẻ giữ xe**, **bãi đỗ xe**, **tính phí tự động**, **thanh toán** và **thống kê doanh thu**.

Ứng dụng cho phép khách gửi xe, quản lý thẻ, tìm kiếm vị trí xe nhanh chóng, đồng thời thống kê doanh thu và tình trạng bãi giữ xe.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý phương tiện**:
  - Thêm, sửa, xóa thông tin xe: biển số, loại xe (xe máy, ô tô, xe tải…).
  - Tra cứu xe theo biển số hoặc loại xe.
- **Quản lý bãi đỗ s chỗ đậu xe**:
  - Thêm, sửa, xóa chỗ đậu xe.
  - Phân loại chỗ: chuẩn, VIP, cho xe điện (kèm trạm sạc).
  - Theo dõi trạng thái chỗ đậu: trống, đã đặt, đang sử dụng.
- **Quản lý thẻ giữ xe**:
  - Cấp thẻ gửi xe cho khách hàng.
  - Phân loại thẻ: lượt, tháng, VIP.
- **Quản lý vào/ra xe**:
  - Quản lý thông tin vào bãi: biển số, loại xe, thời gian vào.
  - Quản lý thông tin ra bãi: thời gian ra, tự động tính phí.
- **Tính phí giữ xe tự động**:
  - Tính phí theo loại xe, loại thẻ, thời gian gửi.
  - Giảm giá cho khách VIP.
- **Thanh toán s hóa đơn**:
  - Xuất hóa đơn thanh toán cho mỗi lần gửi xe.
  - Ghi nhận lịch sử thanh toán.
- **Báo cáo s thống kê**:
  - Số lượng xe đang đậu theo từng thời điểm.
  - Doanh thu theo ngày/tháng/quý.
  - Top khách hàng gửi xe nhiều nhất.
- **Lưu trữ dữ liệu bền vững**: đọc/ghi tệp tự động khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Vehicle *(abstract)* → Motorbike, Car, Truck
    - calculateFee() **override** để tính phí gửi xe tùy loại phương tiện.
  - ParkingCard *(abstract)* → OneTimeCard, MonthlyCard, VipCard.
  - ParkingSpot *(abstract)* → StandardSpot, VipSpot, EVSpot.
- **Interface**:
  - Billable (tính phí giữ xe và tạo hóa đơn).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - ParkingFullException
  - VehicleNotFoundException
  - InvalidCardException
  - PaymentFailedException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:

- vehicles.csv — thông tin phương tiện.
- spots.csv — thông tin chỗ đậu xe.
- cards.csv — thông tin thẻ giữ xe.
- parking_logs.csv — nhật ký vào/ra bãi.
- invoices.csv — hóa đơn thanh toán.
- payments.csv — thông tin thanh toán.
- revenue.csv — doanh thu theo ngày/tháng/quý.

- **Yêu cầu lưu trữ**:
  - Mã xe, mã chỗ đậu, mã thẻ, mã hóa đơn sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd và giờ HH:mm.
  - Trạng thái chỗ đậu: AVAILABLE, BOOKED, OCCUPIED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 20 chỗ đậu xe (chuẩn, VIP, EV).
  - ≥ 30 phương tiện.
  - ≥ 10 khách hàng VIP.
  - ≥ 20 hóa đơn thanh toán.
- **Báo cáo mẫu**:
  - Danh sách xe đang đậu trong bãi.
  - Doanh thu theo ngày/tháng.
  - Top 3 khách hàng gửi xe nhiều nhất.
- **Test case bắt buộc** (≥ 10):

1. Thêm xe mới thành công.
2. Cấp thẻ gửi xe lượt/tháng/VIP thành công.
3. Vào bãi thành công khi còn chỗ.
4. Vào bãi thất bại khi bãi đầy → báo lỗi.
5. Ra bãi thành công, tính phí chính xác.
6. Thanh toán hóa đơn thành công.
7. Xuất hóa đơn ra CSV.
8. Báo cáo doanh thu theo tháng.
9. Thống kê số lượng xe đang đậu tại một thời điểm.
10. Thống kê top 3 khách hàng gửi xe nhiều nhất.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (Vehicle, Motorbike, Car, Truck, ParkingCard, ParkingSpot), xử lý logic vào/ra xe, tính phí giữ xe, quản lý thẻ.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho phương tiện, chỗ đậu, thẻ, nhật ký gửi xe, hóa đơn; import/export dữ liệu.
C	Thiết kế CLI menu, xử lý vào/ra xe, thanh toán, thống kê doanh thu, xuất báo cáo trạng thái bãi xe.

Yêu cầu nâng cao:

- Gợi ý chỗ đậu phù hợp cho xe theo loại và trạng thái.
- Hỗ trợ đăng ký gửi xe dài hạn (gói tháng, quý).
- Xuất hóa đơn thanh toán ra file PDF/CSV.
- Thống kê lượng xe EV để quản lý trạm sạc.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.

- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.

20. Quản lý kho C xuất nhập hàng

Đề tài 20 – Quản lý kho s xuất nhập hàng

1. Mục tiêu

Xây dựng một **Java Console Application** hỗ trợ **quản lý kho hàng**, bao gồm **quản lý sản phẩm**, **phiếu nhập kho**, **phiếu xuất kho**, **tồn kho**, **đơn đặt hàng** và **thống kê doanh thu**.

Ứng dụng giúp người dùng theo dõi chính xác số lượng sản phẩm trong kho, xử lý xuất nhập hàng, kiểm soát hàng tồn và xuất báo cáo.

Không sử dụng cơ sở dữ liệu — toàn bộ dữ liệu được **lưu trữ bền vững bằng tệp** (CSV hoặc nhị phân Serializable).

2. Chức năng chính

- **Quản lý sản phẩm**:
  - Thêm, sửa, xóa thông tin sản phẩm.
  - Thông tin sản phẩm: mã sản phẩm, tên, loại, giá nhập, giá bán, tồn kho.
  - Tìm kiếm sản phẩm theo tên, mã hoặc loại sản phẩm.
- **Quản lý phiếu nhập kho**:
  - Tạo phiếu nhập khi bổ sung hàng vào kho.
  - Ghi nhận thông tin: nhà cung cấp, số lượng nhập, ngày nhập.
  - Tự động cập nhật tồn kho.
- **Quản lý phiếu xuất kho**:
  - Tạo phiếu xuất khi giao hàng.
  - Ghi nhận thông tin khách hàng, số lượng xuất, ngày xuất.
  - Kiểm tra tồn kho trước khi xuất.
- **Quản lý đơn đặt hàng**:
  - Ghi nhận yêu cầu đặt hàng của khách.
  - Liên kết với phiếu xuất kho khi giao hàng.
- **Kiểm kê kho**:
  - Thống kê số lượng tồn thực tế.
  - So sánh giữa tồn thực tế và hệ thống.
- **Báo cáo s thống kê**:
  - Báo cáo nhập – xuất – tồn theo ngày/tháng.
  - Thống kê sản phẩm bán chạy nhất.
  - Báo cáo doanh thu theo kỳ.
- **Lưu trữ dữ liệu bền vững**: tự động đọc/ghi tệp khi khởi động và thoát ứng dụng.

3. Thiết kế OOP

- **Kế thừa/đa hình**:
  - Product *(abstract)* → Electronics, Clothing, Food, Furniture
    - calculateProfit() **override** để tính lợi nhuận theo từng loại sản phẩm.
  - Order *(abstract)* → ImportOrder, ExportOrder.
- **Interface**:
  - Reportable (tạo báo cáo nhập – xuất – tồn).
  - Persistable (load/save dữ liệu).
- **Ngoại lệ tùy biến**:
  - OutOfStockException
  - ProductNotFoundException

- OrderNotFoundException
- InvalidQuantityException

4. Lưu trữ tệp (không dùng CSDL)

- **Tệp CSV/Serializable**:
  - products.csv — thông tin sản phẩm.
  - import_orders.csv — phiếu nhập kho.
  - export_orders.csv — phiếu xuất kho.
  - customers.csv — thông tin khách hàng.
  - suppliers.csv — thông tin nhà cung cấp.
  - inventory.csv — tồn kho.
  - revenue.csv — doanh thu theo ngày/tháng.
- **Yêu cầu lưu trữ**:
  - Mã sản phẩm, mã đơn hàng, mã phiếu nhập/xuất sinh tự động hoặc UUID.
  - Định dạng ngày yyyy-MM-dd.
  - Trạng thái đơn hàng: PENDING, COMPLETED, CANCELLED.

5. Kiểm thử s đầu ra

- **Dataset mẫu**:
  - ≥ 20 sản phẩm thuộc nhiều loại.
  - ≥ 10 nhà cung cấp.
  - ≥ 15 khách hàng.
  - ≥ 20 phiếu nhập kho.
  - ≥ 15 phiếu xuất kho.
- **Báo cáo mẫu**:
  - Báo cáo nhập – xuất – tồn kho theo tháng.
  - Top 5 sản phẩm bán chạy nhất.
  - Doanh thu theo quý.
- **Test case bắt buộc** (≥ 10):

1. Thêm sản phẩm mới thành công.
2. Tạo phiếu nhập kho hợp lệ → tồn kho tăng chính xác.
3. Tạo phiếu xuất kho hợp lệ → tồn kho giảm chính xác.
4. Xuất kho thất bại khi tồn kho không đủ → báo lỗi.
5. Ghi nhận đơn đặt hàng thành công.
6. Cập nhật tồn kho tự động sau khi xuất kho.
7. Báo cáo số lượng tồn kho từng sản phẩm.
8. Báo cáo doanh thu theo tháng.
9. Thống kê sản phẩm bán chạy nhất.
10. Xuất báo cáo nhập – xuất – tồn ra CSV.

6. Chia việc cho 3 thành viên (cân bằng)

Thành viên	Nhiệm vụ chính
A	Thiết kế domain C kế thừa (Product, Electronics, Clothing, Food, Furniture, Order, ImportOrder, ExportOrder), xử lý logic nhập/xuất kho, kiểm kê tồn kho, tính lợi nhuận.
B	Xây dựng tầng lưu trữ tệp (CSV/Serializable), repository cho sản phẩm, phiếu nhập, phiếu xuất, khách hàng, nhà cung cấp, doanh thu; import/export dữ liệu.
C	Thiết kế CLI menu, nhập hàng, xuất hàng, quản lý tồn kho, xử lý đơn đặt hàng, thống kê doanh thu, xuất báo cáo.

Yêu cầu nâng cao:

- Gợi ý nhập thêm sản phẩm sắp hết tồn kho.
- Xuất báo cáo tồn kho và doanh thu ra file PDF/CSV.
- Thống kê xu hướng bán hàng theo mùa.

7. Bàn giao

- **Sơ đồ lớp** (PlantUML/Draw.io).
- **Dataset mẫu** trong thư mục data/.
- **10 test case**: input → output mong đợi.
- **Hướng dẫn chạy ứng dụng** (README.md).
- **Ảnh minh họa CLI** cho các luồng chính.