Chat System

1) شرح مسئله

سیستم به صورت Client/Server پیاده‌سازی می‌شود:

- Server :
  - مدیریت اتصال کاربران، روم‌ها، پیام‌ها، فایل‌های آپلود شده
  - ارسال پیام‌ها به اعضای روم (Broadcast)
  - ذخیره پیام‌ها برای مشاهده تاریخچه و Export به کلاینت
  - ایجاد روم پیش‌فرض به نام lobby هنگام اجرا
- Client :
  - دریافت ورودی کاربر از CLI
  - ارسال درخواست به سرور
  - دریافت و نمایش پیام‌ها/رویدادها
  - دانلود فایل‌ها و ذخیره خروجی JSON در مسیر دلخواه روی کلاینت

الزامات ورود:

- کاربر باید ابتدا Register و سپس Login کند.
- نام کاربری تکراری پذیرفته نمی‌شود.

نکته مهم:

- خروجی JSON باید روی سیستم کلاینت ذخیره شود (نه روی سرور).
- کاربر پس از ورود موفق:
  - یا خودکار وارد lobby شود (پیشنهادی)
  - یا با دستور join وارد روم شود.

2) دستورات مورد نیاز (CLI)

ورودی کلاینت دو نوع است:

- `Command` : با «/» شروع می‌شود.
- `Message` : بدون «/» و پیام چت محسوب می‌شود.

2.1 دستورات عمومی

- `/help` نمایش راهنمای دستورات
- `/exit` خروج از برنامه کلاینت
- `/register <username>` ایجاد حساب کاربری جدید
- `/login <username>` ورود به سیستم با نام کاربری (الزام قبل از ارسال پیام یا پیوستن به روم)

2.2 دستورات مدیریت روم

- `/create <roomName>` ایجاد روم جدید
- `/join <roomName>` ورود به روم
- `/leave` خروج از روم فعلی
- `/rooms` نمایش لیست روم‌ها
- `/users` نمایش کاربران آنلاین داخل روم فعلی

2.3 پیام‌رسانی

- `<text>` ارسال پیام متنی به روم فعلی

2.4 فایل (آپلود/دانلود فایل متنی `.txt`)

- `/upload <localPath>` آپلود فایل `txt` از مسیر محلی کلاینت

- `/download <fileId> <savePath>` > دانلود فایل با شناسه و ذخیره در مسیر دلخواه روی کلاینت

2.5 خروجی گرفتن از چت (Export)

- `/export last <N> <savePath>` خروجی گرفتن از آخرین `N` پیام روم فعلی به صورت `JSON` و ذخیره روی کلاینت
  - قیود پیشنهادی: `1 <= N <= 200`
  - اگر `N` خارج از بازه یا نامعتبر بود، خطای مناسب نمایش داده شود.

3) نیازمندی‌های طراحی

3.1 معماری و شبکه

- ارتباط کلاینت و سرور از طریق `TCP Socket`
- پشتیبانی همزمان چند کلاینت در سرور:
  - `Thread-per-client` یا `ThreadPool`
- در کلاینت حداقل دو `Thread` :
  - `Thread` دریافت پیام‌ها از سرور و چاپ
  - `Thread` دریافت ورودی کاربر و ارسال

3.2 Serialization

- انتقال داده بین کلاینت و سرور به‌صورت ارسال/دریافت آبجکت
  - پیشنهاد: `ObjectInputStream` و `ObjectOutputStream`
- ترتیب ساخت `Stream` ها باید صحیح باشد تا `Deadlock` ایجاد نشود.

3.3 Generics

الزام استفاده از ساختارهای جنریک:

- `Packet<T>` جنریک برای ارتباط شبکه (`Request/Response`)
- حداقل یک مورد دیگر مانند: `Repository<T>` یا `Serializer<T>` یا `Message<Tcontent>` با `TContent extends Content`

نمونه ساده `Packet` :

```
public class Packet<T> implements java.io.Serializable {
  private PacketType type;
  private T payload;
  private long timestamp;
  private String requestId; // اختیاری

  // getters, setters, constructors
}
```

3.4 Concurrency + Critical Section

بخش‌های بحرانی را شناسایی و محافظت کنید:

- اضافه/حذف کاربر از روم
- `Broadcast` پیام به اعضای روم
- ثبت پیام در `history` روم
- مدیریت `Map` مربوط به `fileId -> FileMetadata`

روش‌های محافظت:

- `synchronized` ، `ReentrantLock` ، یا ساختارهای همزمان مانند `ConcurrentHashMap`

در گزارش کوتاه مشخص کنید `Critical Section` ها کجا هستند و چرا.

3.5 کلاس‌ها/ماژول‌های پیشنهادی

(دقیقاً لازم نیست نام کلاس‌ها همین باشد، اما باید نقش‌ها پوشش داده شود)

- شبکه و پروتکل:
  - `Packet<T>`
    - `type: PacketType`
    - `payload: T`
    - `timestamp: long`
    - `requestId: String` \(اختیاری) برای ردیابی درخواست‌ها
  - `enum PacketType`
    - for example: (LOGIN_REQ, LOGIN_RES, CHAT_MSG, ROOM_EVENT, COMMAND_REQ, COMMAND_RES, FILE_UPLOAD_REQ, FILE_DOWNLOAD_REQ, EXPORT_REQ, EXPORT_RES, ERROR)
- مدل داده:
  - `UserSession` :
    - username
    - loggedIn (true/false): وضعیت ورود
    - currentRoom
    - socket/streams
  - `Room` :
    - name
    - members
    - <history: List<Message>
  - `Message` :
    - id
    - sender

- timestamp
- type: TEXT | SYSTEM | FILE
- content
  - `FileMetadata` :
    - fileId
    - originalName
    - uploader
    - uploadTime
    - size
    - serverPath
- سرویس‌ها:
  - `RoomService` (create/join/leave/list/users)
  - `MessageService` (append/history/last N)
  - `FileStorageService` (save/load/validate)
  - `ExportService` (Server: تهیه لیست پیام‌ها؛ Client: ساخت `JSON` و نوشتن روی دیسک)
    - Server-side: تولید لیست پیام‌ها برای export
    - Client-side: ساخت JSON و نوشتن در مسیر مورد نظر
- `Command Pattern` :
  - `Command interface` : برای پردازش دستورات سمت کلاینت یا سمت سرور
  - `CommandContext` : حاوی سرویس‌ها و وضعیت جاری
  - کلاس مستقل برای هر دستور مانند `JoinCommand` , `UploadCommand` , `ExportCommand`

3.6 مشخصات فایل و ذخیره‌سازی

- فقط `.txt` مجاز
- اندازه فایل پیشنهادی: حداکثر `200KB`
- سرور فایل را در پوشه‌ای مانند `server_storage/` ذخیره کند (در صورت نبود، بسازد)
- `fileId` یکتا (مثلاً `UUID`)

3.7 قالب JSON برای Export

حداقل شامل:

- نام روم
- زمان `export`
- لیست پیام‌ها (شامل فراداده)

نمونه ساختار:

```
{
  "room": "lobby",
  "exportedAt": "2025-01-01T12:34:56Z",
  "messages": [
    {
      "id": "uuid",
      "sender": "alice",
      "timestamp": 1704100000,
      "type": "TEXT",
      "content": "Hello"
    }
  ]
}
```

3.8 مدیریت خطا (Error Handling)

موارد زیر باید هندل شود (برنامه نباید کرش کند):

- خطاهای ورودی و منطقی:
  - نام کاربری تکراری/نامعتبر یا خالی
  - `join` به روم ناموجود
  - `leave` وقتی کاربر در هیچ رومی نیست
  - ارسال پیام خارج از روم (در صورت تعریف چنین حالت)
  - `N` نامعتبر در `export`
- خطاهای فایل:
  - مسیر آپلود نامعتبر / نبودن فایل
  - فایل غیر `txt`
  - `fileId` نامعتبر برای دانلود
  - عدم دسترسی به `savePath` روی کلاینت
  - خطا در ذخیره روی سرور
- خطاهای شبکه:
  - قطع ناگهانی اتصال (`SocketException` , `EOFException`)
  - خطاهای I/O در ارسال/دریافت

3.9 Exception Management Platform in Java

- چند `Custom Exception` مانند:
  - `RoomNotFoundException`
  - `DuplicateUsernameException`
  - `FileTransferException`
  - `InvalidCommandException`
- ماژول مرکزی مانند `ExceptionManager` یا `ErrorHandler` که:
  - خطاها را `Log` کند
  - به پاسخ استاندارد تبدیل نماید (`PacketType.ERROR` با `ErrorPayload`)

- حداقل یک استفاده صریح از `finally` برای بستن منابع/پاکسازی

4) تحویل‌دادنی‌ها

۱. کد کامل پروژه (`Client + Server`)

۲. فایل `README` شامل:

- نحوه اجرا (پورت، دستور run سرور/کلاینت)
- لیست دستورات و مثال کوتاه
- توضیح ساختار پروژه

۳. گزارش کوتاه (۲ تا ۵ صفحه) شامل:

- اینکه هر سرفصل `HW3` کجا در پروژه پیاده‌سازی شده
- `Pros & Cons` روش قدیمی vs `Exception`
- `Critical Section` ها و روش `Synchronization`
- اشاره به `Runtime Exception` و `finally`