سامانه نقشه عملیاتی همکارانه (Ops Map)

پروژه‌ی نقشه عملیاتی همکارانه (Ops Map) یک نرم‌افزار کاربردی بلادرنگ مبتنی بر معماری کلاینت-سرور است که با زبان جاوا و با استفاده از JavaFX برای رابط کاربری و Socket Programming برای ارتباطات شبکه پیاده‌سازی شده است. هدف اصلی این پروژه، ایجاد بستری برای همکاری هم‌زمان چند کاربر (با نقش‌های فرمانده و اپراتور) روی یک نقشه‌ی مشترک است تا بتوانند:

- **مسیرهای عملیاتی ترسیم کنند.**
- **نشانگرهای تاکتیکی (امن، خطر، پایگاه) درج کنند.**
- **مناطق عملیاتی را به صورت اشکال هندسی (مستطیل، دایره) مشخص کنند.**
- **از دسته‌بندی رنگی برای تیم‌ها استفاده کنند.**
- **اشیاء ایجاد شده توسط خود را حذف کنند (و فرماندهان بتوانند هر شیء را حذف کنند).**
- **مکان‌نمای ماوس یکدیگر را دنبال کنند.**
- **با یکدیگر چت متنی داشته باشند.**
- **وضعیت نقشه را ذخیره و بارگذاری کنند.**
- **تصویر دلخواه به عنوان پس‌زمینه نقشه قرار دهند.**

پروژه با رعایت اصول شیءگرایی، استفاده از الگوهای طراحی (Singleton، Observer، Command) و مدیریت استثناها پیاده‌سازی شده است. در ادامه هر یک از بخش‌های کد به تفکیک پکیج و کلاس تشریح می‌شود.

۲. معماری کلی سیستم

سیستم از سه لایه‌ی اصلی تشکیل شده است:

1. **سرور مرکزی** (Server)

- **مدیریت احراز هویت کاربران.**
- **نگهداری وضعیت جاری نقشه در حافظه.**
- **دریافت پیام‌های کلاینت‌ها، اعمال تغییرات روی وضعیت نقشه و پخش پیام‌ها به سایر کلاینت‌ها.**
- **ذخیره و بازیابی نقشه در/از فایل.**

2. **کلاینت** (Client)

- **رابط کاربری گرافیکی مبتنی بر** JavaFX.
- **اتصال به سرور و ارسال/دریافت پیام‌ها.**
- **نمایش نقشه و ابزارهای تعاملی.**
- **اعتبارسنجی اولیه ورودی.**

3. **اشیاء اشتراکی** (Shared)

- **مدل‌های داده‌ای که بین سرور و کلاینت تبادل می‌شوند.**
- **پروتکل پیام‌ها** (Message **و** MessageType).
- **تمام این کلاس‌ها** Serializable **هستند تا امکان ارسال از طریق شبکه و ذخیره در فایل فراهم شود.**

پروتکل ارتباطی:

- **ارتباط از طریق** TCP Socket **و با استفاده از** Object Streams **انجام می‌شود.**
- **هر پیام یک شیء از نوع** Message **است که شامل** MessageType **و** payload **می‌باشد.**
- **سرور پس از دریافت پیام، بسته به نوع آن واکنش نشان می‌دهد.**

۳. شرح کامل بسته‌ها و کلاس‌ها

۳'۱. بسته server

این بسته شامل تمام کلاس‌های سمت سرور است و به چند زیربسته تقسیم می‌شود.

۳'۱'۱. زیربسته server.storage

کلاس UserStore

- **مسئولیت: ذخیره‌سازی موقت اطلاعات کاربران در حافظه (بدون پایگاه داده).**
- **ویژگی‌ها:**
  - Map<String, UserRecord> users – **از** ConcurrentHashMap **استفاده می‌کند (برای هم‌روندی).**
- **کلاس داخلی** UserRecord**:**
  - **نگهدارنده‌ی سه‌تایی (**username, password, role**).**
- **متدهای اصلی:**
  - userExists(String username)**: بررسی وجود کاربر.**
  - validate(String username, String password)**: تطابق رمز عبور (متن ساده – قابل بهبود).**
  - addUser(String username, String password, String role)**: افزودن کاربر جدید.**
  - getRole(String username)**: دریافت نقش کاربر.**
- **نکته: رمز عبور به صورت هش‌شده ذخیره نمی‌شود (ضعف امنیتی شناخته‌شده).**

کلاس MapStorage

- **مسئولیت: ذخیره و بازیابی شیء** MapState **در/از فایل با استفاده از سریال‌سازی جاوا.**
- **ثابت:** private static final String FILE = "map_state.dat"**.**
- **متدهای ایستا:**

- save(MapState state): **نوشتن شیء در فایل.**
- load(): **خواندن شیء از فایل.**
- **نکته: مسیر فایل ثابت است و امکان تنظیم وجود ندارد.**

۳'۱'۲. زیربسته server.manager

کلاس MapStateManager (Singleton)

- **مسئولیت: نگهداری نمونه‌ی یگانه از وضعیت جاری نقشه.**
- **ویژگی‌ها:**
  - private static MapStateManager instance
  - private MapState currentState
- **متدها:**
  - getInstance(): **دریافت نمونه.**
  - getCurrentState(), setCurrentState(MapState).
  - addRoute(Route), addMarker(Marker), addRegion(RegionShape): **افزودن عناصر به وضعیت جاری.**
  - synchronized boolean removeObject(String id):
    - **حذف شیء با شناسه از لیست مسیرها، نشانگرها یا مناطق.**
    - **از** Iterator **برای پیمایش و حذف ایمن استفاده می‌کند.**
    - **این متد همگام‌سازی شده تا در محیط چندنخی مشکلی ایجاد نشود.**

کلاس ClientManager

- **مسئولیت: مدیریت مجموعه‌ی کلاینت‌های متصل.**

- **ویژگی‌ها:**
- Set<ClientHandler> clients = Collections.synchronizedSet(new HashSet<>())
- **متدها:**
- addClient(ClientHandler), removeClient(ClientHandler) **.**
- broadcast(Message msg, ClientHandler exclude) **:**
- **ارسال پیام به تمام کلاینت‌ها به جز یک کلاینت خاص.**
- **با بلوک** synchronized(clients) **از تداخل جلوگیری می‌کند.**
- getUsers **():**
- **لیستی از اشیاء** User **مربوط به کلاینت‌های آنلاین می‌سازد (بدون رمز عبور).**

۳٬۱٬۳. زیربسته server.auth

کلاس AuthManager (Singleton)

- **مسئولیت: مدیریت فرآیندهای ورود و ثبت‌نام کاربران.**
- **ویژگی‌ها:**
- private UserStore store
- **متدها:**
- login(String username, String password) **:**
- **اعتبارسنجی با** UserStore.validate **.**
- **در صورت موفقیت، یک شیء** User **با نقش و نام کاربری (بدون رمز) برمی‌گرداند.**
- register(String username, String password, String role) **:**
- **اگر کاربر وجود نداشته باشد، او را به** UserStore **اضافه می‌کند.**

- **نکته: در نسخه اصلاح‌شده، نقش باید** Commander **یا** Operator **باشد و سرور آن را اعتبارسنجی می‌کند.**

۳’۱’۴. کلاس MainServer

- **مسئولیت: نقطه‌ی ورود سرور.**
- **ثابت:** public static final int PORT = 5000.
- **متد** main**:**
- **ایجاد** ServerSocket **روی پورت مشخص.**
- **حلقه‌ی بی‌نهایت: پذیرش اتصال جدید، ایجاد یک شیء** ClientHandler **و شروع نخ آن.**
- **نکته: محدودیت تعداد اتصال اعمال نشده است (آسیب‌پذیری** DoS**).**

۳’۱’۵. کلاس ClientHandler (extends Thread)

- **مسئولیت: مدیریت یک کلاینت خاص از لحظه‌ی اتصال تا قطع ارتباط.**
- **ویژگی‌ها:**
  - Socket socket, ObjectInputStream in, ObjectOutputStream out
  - ClientManager clientManager
  - User user **(پس از احراز هویت مقدار می‌گیرد)**
- **متد** run()**:**
- **ایجاد جریان‌های ورودی/خروجی.**
- **دریافت پی‌درپی پیام‌ها با** in.readObject() **و فراخوانی** handleMessage().
- **در صورت بروز استثناء (قطع اتصال)، کاربر از** ClientManager **حذف و پیام‌های** USER_LEFT **و** USER_LIST **پخش می‌شود.**

- **متد** handleMessage(Message msg)**:**
- **بر اساس** MessageType **عملیات مناسب را انجام می‌دهد.**
- **برای تمام پیام‌ها به جز** LOGIN **و** REGISTER**:**
- **ابتدا** if (user == null) **بررسی می‌شود و در صورت عدم احراز هویت، خطای** UNAUTHORIZED **برگردانده می‌شود.**
- **پیام‌های ترسیم (**DRAW_ROUTE, ADD_MARKER, ADD_REGION**):**
- **شیء دریافتی به** MapStateManager **اضافه می‌شود.**
- **پیام به سایر کلاینت‌ها پخش می‌شود.**
- **پیام** REMOVE_OBJECT**:**
- **شناسه از** payload **استخراج می‌شود.**
- **با متد کمکی** findOwnerById **مالک شیء از وضعیت جاری نقشه پیدا می‌شود.**
- **بررسی مجوز:**
- **اگر کاربر فرستنده با مالک یکی نباشد و نقش او** Commander **نباشد ← خطای** FORBIDDEN**.**
- **اگر شیء وجود نداشته باشد ← خطای** NOT_FOUND**.**
- **در صورت تأیید،** MapStateManager.removeObject(id) **فراخوانی و نتیجه بررسی می‌شود.**
- **اگر حذف موفق باشد، پیام به سایر کلاینت‌ها پخش می‌شود.**
- **پیام** MOUSE_MOVE **و** CHAT**:**
- **فقط در صورت لاگین بودن، به سایرین پخش می‌شود.**
- **پیام** SAVE_STATE**:**
- **شیء** MapState **دریافتی با** MapStorage.save() **ذخیره می‌شود.**
- **پیام** LOAD_STATE**:**
- MapStorage.load() **وضعیت ذخیره‌شده را می‌خواند.**

- MapStateManager.setCurrentState() **وضعیت جاری را به‌روز می‌کند.**
- **وضعیت جدید هم به درخواست‌دهنده و هم با** broadcast **به همه‌ی کلاینت‌ها ارسال می‌شود.**
- **متد کمکی** findOwnerById(String id, MapState state)**:**
- **در لیست نشانگرها، مناطق و مسیرها به دنبال شناسه می‌گردد و** owner **را برمی‌گرداند.**
- **متدهای** handleLogin **و** handleRegister**:**
- **اعتبارسنجی ورودی (طول نام کاربری و رمز).**
- **در ثبت‌نام، نقش باید** Commander **یا** Operator **باشد.**
- **در صورت موفقیت، کاربر در** AuthManager **ثبت/احراز می‌شود،** user **تنظیم می‌شود، به** ClientManager **افزوده می‌شود و پیام‌های خوش‌آمدگویی و وضعیت نقشه ارسال می‌گردد.**
- **متدهای** send(Message) **و** sendError(String code, String text)**:**
- **ارسال پیام به کلاینت.**
- **نکات امنیتی اصلاح‌شده:**
- **بررسی احراز هویت برای همه‌ی دستورات.**
- **بررسی مجوز برای حذف اشیاء.**
- **اعتبارسنجی نقش در ثبت‌نام.**

۳٬۲. بسته shared

این بسته شامل کلاس‌های مشترک بین سرور و کلاینت است و به دو زیربسته model **و** message **تقسیم می‌شود.**

۳٬۲٬۱. زیربسته shared.model

کلاس User

- **ویژگی‌ها:** String username, String role, String password
- **کاربرد: انتقال اطلاعات کاربر هنگام ورود/ثبت‌نام.**
- **نکته: رمز عبور فقط برای انتقال استفاده می‌شود و در سمت سرور ذخیره نمی‌شود.**

کلاس Route

- **ویژگی‌ها:**
  - String id – **شناسه‌ی یکتا (تولید شده با** UUID**).**
  - List<Double> xPoints, yPoints – **نقاط مسیر.**
  - String color, double thickness, String owner**.**
- **سازنده‌ها:**
  - Route(String color, double thickness, String owner) – **شناسه به‌طور خودکار ساخته می‌شود.**
  - Route(String id, String color, double thickness, String owner) – **برای بازسازی از فایل.**
- **متد** addPoint(double x, double y)**: افزودن نقطه به مسیر.**
- **نکته: افزودن شناسه، امکان حذف مسیرها را فراهم کرده است.**

کلاس Marker

- **ویژگی‌ها:** double x, y, String type, String color, String owner, String id**.**
- **نکته: نوع نشانگر می‌تواند** SAFE, DANGER, BASE **باشد.**

کلاس RegionShape (abstract)

- **ویژگی‌های پایه:** String id, String color, String owner**.**
- **زیرکلاس‌ها:**

- RectangleRegion: **مختصات گوشه** (x, y, width, height).
- CircleRegion: **مرکز** (centerX, centerY) **و شعاع** (radius).

کلاس MousePosition

- **ویژگی‌ها:** double x, y, String user.
- **کاربرد: ارسال موقعیت موس.**

کلاس ChatMessage

- **ویژگی‌ها:** String user, String text.
- **کاربرد: پیام چت.**

کلاس MapState

- **ویژگی‌ها:**
  - List<Route> routes
  - List<Marker> markers
  - List<RegionShape> regions
- **متدهای افزودن و دریافت.**
- **کاربرد: ذخیره‌سازی وضعیت کامل نقشه و انتقال آن بین کلاینت و سرور.**

۳٬۲٬۲. زیربسته shared.message

enum MessageType

- **تمام ثابت‌های مورد نیاز برای انواع پیام‌ها:**

- LOGIN, REGISTER, LOGIN_SUCCESS, ERROR
- DRAW_ROUTE, ADD_MARKER, ADD_REGION, REMOVE_OBJECT
- MOUSE_MOVE, CHAT
- USER_JOINED, USER_LEFT, USER_LIST
- SAVE_STATE, LOAD_STATE, MAP_STATE

کلاس Message

- **ویژگی‌ها:** MessageType type, Object payload.
- **کاربرد: لفافه‌ی تمام پیام‌های شبکه.**

کلاس ErrorPayload

- **ویژگی‌ها:** String code, String message.
- **کاربرد: ساختار استاندارد خطا برای ارسال از سرور به کلاینت.**

۳٬۳. بسته client

این بسته شامل کلاس‌های سمت کلاینت است.

۳٬۳٬۱. زیربسته client.network

اینترفیس MessageListener

- **متد:** void onMessage(Message message).
- **کاربرد: اعلان دریافت پیام به کنترلرها.**

کلاس ClientConnection (Singleton)

- **مسئولیت: مدیریت اتصال** TCP **به سرور.**
- **ویژگی‌ها:** Socket socket, ObjectOutputStream out, ObjectInputStream in**.**
- **متدها:**
  - connect(String host, int port)**: برقراری اتصال و ایجاد جریان‌ها.**
  - send(Message msg)**: ارسال پیام.**
  - receive()**: دریافت یک پیام (مسدودکننده).**
  - startListening(MessageListener listener)**: ایجاد نخ شنونده که به‌طور پیوسته پیام‌ها را دریافت و به** listener **تحویل می‌دهد.**

۳٬۳٬۲. زیربسته client.controller

کلاس LoginController

- **مسئولیت: کنترل صفحه‌ی ورود/ثبت‌نام.**
- **ویژگی‌های** FX**:**
  - TextField usernameField
  - PasswordField passwordField
  - ComboBox<String> roleBox (Commander, Operator)
  - Label errorLabel
- **متد** initialize**: مقداردهی** roleBox**.**
- **متد** authenticate(MessageType type)**:**
  - **اعتبارسنجی سمت کلاینت (طول نام کاربری، رمز عبور).**
  - **اتصال به سرور از طریق** ClientConnection**.**

- **ارسال پیام** LOGIN **یا** REGISTER**.**
- **دریافت پاسخ:**
- **اگر** LOGIN_SUCCESS **باشد، کاربر جاری در** MainClient **ذخیره و صفحه‌ی اصلی باز می‌شود.**
- **اگر** ERROR **باشد، پیام خطا در** errorLabel **نمایش داده می‌شود.**

کلاس MainController (implements MessageListener)

- **مسئولیت: کنترل صفحه‌ی اصلی نقشه.**
- **ویژگی‌های** FX**:**
  - Pane mapPane – **بستر رسم.**
  - **ابزارها:** ToggleButton routeToolBtn, markerToolBtn, regionToolBtn
  - **تنظیمات:** ChoiceBox markerTypeBox, ColorPicker colorPicker, Slider thicknessSlider, ChoiceBox shapeTypeBox
  - **دکمه‌ها:** Button deleteSelectedBtn, Button loadBgBtn
  - **چت و کاربران:** ListView chatList, TextField chatInput, ListView userList
- **ویژگی‌های داخلی (مدل محلی):**
  - Polyline currentLine, Route currentRoute – **برای رسم مسیر.**
  - double startX, startY, Shape tempShape – **برای رسم منطقه.**
  - Map<String, Circle> cursors – **مکان‌نمای سایر کاربران.**
  - Map<String, Marker> markers, Map<String, RegionShape> regions – **نگاشت شناسه به شیء.**
  - List<Route> routes – **لیست مسیرها (برای سازگاری).**
  - Map<String, Route> routesMap – **نگاشت شناسه به مسیر (برای حذف).**
  - Map<String, Polyline> routeNodes – **نگاشت شناسه به گره گرافیکی مسیر (برای حذف).**

- User currentUser, String selectedObjectId.

- **متد** initialize**:**

- **مقداردهی اولیه، گروه‌بندی ابزارها، تنظیم رویدادها و شروع شنود شبکه.**

- **متدهای راه‌اندازی رویداد:**

- setupRouteDrawing**(): رسم مسیر با کلیک و درگ – در پایان، مسیر با شناسه به سرور ارسال و در نقشه‌های محلی ذخیره می‌شود.**

- setupMarker**(): کلیک چپ روی نقشه → ساخت** Marker **با شناسه‌ی جدید → ارسال به سرور و رسم.**

- setupRegion**(): کلیک و درگ برای رسم مستطیل/دایره → ساخت** RegionShape **با شناسه‌ی جدید → ارسال به سرور و رسم.**

- setupMouseTracking**(): ارسال** MousePosition **با هر حرکت موس.**

- setupObjectSelection**(): انتخاب شیء با کلیک و ذخیره‌ی** selectedObjectId.

- **متد** deleteSelected**():**

- **اگر شیئی انتخاب نشده باشد، خطا نشان می‌دهد.**

- **مالک شیء از نقشه‌های محلی استخراج می‌شود.**

- canModify(owner) **بررسی می‌شود (نقش** Commander **یا مالک بودن).**

- **در صورت مجاز بودن،** removeObject(selectedObjectId) **فراخوانی می‌شود.**

- **متد** removeObject(String id)**:**

- **پیام** REMOVE_OBJECT **با شناسه به سرور ارسال می‌شود.**

- **همچنین** removeById(id) **برای حذف محلی فراخوانی می‌شود.**

- **متد** removeById(String id)**:**

- **شیء از نقشه‌های محلی (**markers, regions, routesMap, routeNodes**) حذف می‌شود.**

- **گره گرافیکی متناظر از** mapPane **حذف می‌شود.**
- **متد** onMessage(Message msg)**:**
- **بروزرسانی رابط کاربری در** Platform.runLater**.**
- ADD_MARKER, ADD_REGION **: رسم شیء جدید.**
- REMOVE_OBJECT **: فراخوانی** removeById**.**
- DRAW_ROUTE **: رسم مسیر دریافتی (**drawRemoteRoute**).**
- MOUSE_MOVE **: نمایش/جابجایی مکان‌نما.**
- CHAT **: افزودن پیام به** chatList**.**
- MAP_STATE **: پاک کردن نقشه و بارگذاری مجدد (**loadMap**).**
- USER_LIST **: به‌روزرسانی لیست کاربران.**
- ERROR **: نمایش خطا.**
- **متد** drawRemoteRoute(Route route)**:**
- **ایجاد** Polyline **از نقاط مسیر، ذخیره در** routesMap **و** routeNodes**.**
- **تنظیم رویداد کلیک راست برای حذف.**
- **متد** loadMap(MapState state)**:**
- **پاک کردن تمام ساختارهای محلی و بازسازی نقشه از روی** state**.**
- **متد** canModify(String owner)**:**
- **بررسی مجوز:** Commander **یا همان کاربر.**
- **متد** showError(String)**: نمایش پیام خطا در** Alert**.**

۳٬۳٬۳. کلاس MainClient (extends Application)

- **مسئولیت: راه‌انداز اصلی** JavaFX**.**

- **ویژگی‌های ایستا:** Stage primaryStage, User currentUser.
- **متد** start(Stage)**: نمایش صفحه‌ی لاگین.**
- **متد** showLogin() **و** showMain()**: بارگذاری** FXML **و تغییر صحنه.**
- **متدهای** setCurrentUser **و** getCurrentUser.

۴. جریان کاری (Workflow) سیستم

۴'۱. راه‌اندازی سرور

1. **اجرای** MainServer.
2. **سرور روی پورت ۵۰۰۰ منتظر اتصال می‌ماند.**

۴'۲. ورود یا ثبت‌نام کاربر

1. **کاربر در کلاینت لاگین یا ثبت‌نام می‌کند.**
2. **اعتبارسنجی اولیه در** LoginController.
3. **اتصال به سرور با** ClientConnection.connect().
4. **ارسال پیام** LOGIN **یا** REGISTER.
5. **سرور در** ClientHandler.handleLogin/handleRegister **اعتبارسنجی نهایی را انجام می‌دهد.**
6. **در صورت موفقیت:**
   - **کاربر در** AuthManager **ثبت/احراز می‌شود.**
   - ClientHandler.user **تنظیم می‌شود.**
   - **کلاینت به** ClientManager **افزوده می‌شود.**
   - **پیام** LOGIN_SUCCESS **به همراه وضعیت نقشه و لیست کاربران ارسال می‌شود.**

- **پیام** USER_JOINED **و** USER_LIST **به سایر کاربران پخش می‌شود.**

۴٬۳. رسم مسیر

1. **کاربر ابزار** Route **را انتخاب می‌کند.**
2. **با کلیک و درگ روی** mapPane**، نقاط به** currentLine **و** currentRoute **اضافه می‌شود.**
3. **پس از رها کردن موس:**
   - **مسیر محلی ذخیره می‌شود (**routesMap, routeNodes**).**
   - **پیام** DRAW_ROUTE **با شیء** Route **به سرور ارسال می‌شود.**
4. **سرور:**
   - **مسیر را به** MapStateManager **اضافه می‌کند.**
   - **پیام را به سایر کلاینت‌ها پخش می‌کند.**
5. **کلاینت‌های دیگر:**
   - drawRemoteRoute **مسیر را روی نقشه رسم می‌کنند و در ساختارهای محلی ذخیره می‌کنند.**

۴٬۴. افزودن نشانگر

1. **کاربر ابزار** Marker **را انتخاب می‌کند.**
2. **کلیک روی نقشه:**
   - **ساخت** Marker **با شناسه‌ی جدید.**
   - **رسم محلی و ارسال پیام** ADD_MARKER **به سرور.**
3. **سرور: نشانگر را به** MapStateManager **اضافه و پیام را پخش می‌کند.**

۴٬۵. افزودن منطقه

1. **کاربر ابزار** Region **را انتخاب می‌کند.**

2. **کلیک و درگ: رسم موقت مستطیل/دایره.**

3. **پس از رها کردن:**

- **ساخت** RectangleRegion **یا** CircleRegion **با شناسه‌ی جدید.**
- **رسم محلی و ارسال پیام** ADD_REGION**.**

۴٬۶. حذف اشیاء

1. **کاربر روی یک شیء کلیک راست می‌کند یا آن را انتخاب کرده و دکمه‌ی** Delete Selected **را می‌زند.**

2. **کلاینت:**

- **بررسی** canModify(owner)**.**
- **ارسال پیام** REMOVE_OBJECT **با شناسه به سرور.**
- **حذف محلی** (removeById)**.**

3. **سرور:**

- **یافتن مالک با** findOwnerById**.**
- **بررسی مجوز (مالک یا** Commander**).**
- **فراخوانی** MapStateManager.removeObject(id)**.**
- **پخش پیام به سایر کلاینت‌ها.**

4. **سایر کلاینت‌ها: با دریافت پیام،** removeById **را اجرا می‌کنند.**

۴٬۷. ذخیره و بارگذاری نقشه

1. **کاربر دکمه‌ی** Save Map **را می‌زند:**

- collectMapState() **وضعیت محلی را جمع‌آوری می‌کند.**

- **پیام** SAVE_STATE **با** MapState **به سرور ارسال می‌شود.**
- **سرور** MapStorage.save() **را فراخوانی می‌کند.**

2. **کاربر دکمه‌ی** Load Map **را می‌زند:**

- **پیام** LOAD_STATE **ارسال می‌شود.**
- **سرور** MapStorage.load() **را اجرا می‌کند.**
- **وضعیت جدید در** MapStateManager **تنظیم می‌شود.**
- **سرور** MAP_STATE **را به درخواست‌دهنده و سپس با** broadcast **به همه ارسال می‌کند.**
- **کلاینت‌ها** loadMap **را اجرا کرده و نقشه‌ی جدید را نمایش می‌دهند.**

۴'۸. چت و مکان‌نما

- **چت: کاربر در** chatInput **تایپ کرده و** Enter **می‌زند ← پیام** CHAT **ارسال می‌شود ← سرور پخش می‌کند ← کلاینت‌ها پیام را در** chatList **نمایش می‌دهند.**
- **مکان‌نما: با هر حرکت موس، پیام** MOUSE_MOVE **ارسال می‌شود ← سرور پخش می‌کند ← کلاینت‌ها موقعیت موس سایر کاربران را با دایره‌های نارنجی نشان می‌دهند.**