Projekt

Objektorientierte Programmierung:

```
private void addEntry(String text) {
        entries.add(text);
        LocalDate date = LocalDate.of(year, month, day);
        String isoDate = date.toString(); // YYYY-MM-DD
        String formattedForServer = isoDate + "|" + text;
        CalenderEntry entryView = new CalenderEntry(text, formattedForServer)
        entryBox.getChildren().add(entryView);
        System.out.println("New Task: " + formattedForServer);
        NetworkManager.sendTask(formattedForServer);
    }
```

RegExp

```
private List<String> searchTasksWithRegex(String regex) {
        String[] everyTask = String.valueOf(NetworkManager.loadUserTasks()).sp
        List<String> matches = new java.util.ArrayList<>();
        for (String task : everyTask) {
            String[] parts = task.split("\\|", 2);
            if (parts.length == 2) {
                String date = parts[0].replace("[", "").replace("]", "").trim();
                String name = parts[1].replace("[", "").replace("]", "").trim();
                if (name.matches(".*" + regex + ".*")) {
                    String[] dateParts = date.split("-");
                    String formattedDate = dateParts[2] + "." + dateParts[1] + "." + dat
                    String output = " " + name + " ist am " + formattedDate + " zu erle
                    matches.add(output);
                }
            }
        }
        if (matches.isEmpty()) {
            matches.add("Keine Treffer gefunden.");
        }
```

```
        return matches;
    }
```

Collections

```
CalenderDayCell:
private final List<String> entries = new ArrayList<>();
entries.add(text);
```

Exception

```
public static boolean connect() {
        try {
            socket = new Socket(SERVER_ADDRESS, PORT);
            out = new PrintWriter(socket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(socket.getInputStrea
            return true;
        } catch (IOException e) {
            System.err.println("Connection error: " + e.getMessage());
            return false;
        }
    }
```

Threads

```
try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("- Client connected: " + socket.getInetAddress().
                new Thread(() -> handleClient(socket)).start();
            }
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
```

Sockets

```
Server:
public static void main(String[] args) {
    loadUsers();
    System.out.println("- Server started on port: " + PORT);
    try (ServerSocket serverSocket = new ServerSocket(PORT)) {
        while (true) {
            Socket socket = serverSocket.accept();
            System.out.println("- Client connected: " + socket.getInetAddress().
            new Thread(() -> handleClient(socket)).start();
        }
    } catch (IOException e) {
        System.err.println("Error: " + e.getMessage());
    }
}

Client:
public static boolean connect() {
    try {
        socket = new Socket(SERVER_ADDRESS, PORT);
        out = new PrintWriter(socket.getOutputStream(), true);
        in = new BufferedReader(new InputStreamReader(socket.getInputStrea
        return true;
    } catch (IOException e) {
        System.err.println("Connection error: " + e.getMessage());
        return false;
    }
}
```

Sonstiges

CSS

```
scene.getStylesheets().add(getClass().getResource("/style.css").toExternalF
        popupStage.setScene(scene);
        popupStage.showAndWait();

        accountBtn.setStyle(
                "-fx-background-color: #d0ddff;" +
                        "-fx-text-fill: black;" +
                        "-fx-font-size: 16px;" +
                        "-fx-font-weight: bold;" +
                        "-fx-padding: 8 20 8 20;" +
                        "-fx-background-radius: 50;"
        );
```

Einfachheit

```
String line;
            while ((line = in.readLine()) != null) {
                String[] parts = line.split("\\|");
                if (parts.length == 0) {
                    out.println("ERROR");
                    continue;
                }
                String command = parts[0];

                switch (command) {
                    case "login" -> handleLogin(parts, out);
                    case "task" -> handleTask(parts, out);
                    case "signup" -> handleSignup(parts, out);
                    case "account" -> handleAccount(parts, out);
                    case "config" -> handleConfig(parts, out);
                    default -> out.println("ERROR");
                }
            }
```