# Programmierprojekt FS20
# Bachelor Chasers
# Diary

Adrian Prokopczyk
Johannes Nussbaum
Anna Diack
Meipei Nghiem

**Eintrag 1:** Donnerstag, 20.02.2020
Erster Termin der Einführungsvorlesung. Sehr rasch nachdem wir unsere Gruppe gebildet hatten, einigten wir uns auf Black Jack als Basis für unser Spiel. Um es etwas spannender zu gestalten, möchten wir Begriffe aus dem Studentenleben einbauen: Das Ziel ist es, 180 Punkte zu sammeln ("Bachelor"). Die Karten könnten benannt werden nach Ereignissen im Studentenleben: Kaffee und Lernen gibt Pluspunkte; Partymachen und Schwänzen gibt Minuspunkte.
Wir erstellen eine WhatsApp-Gruppe zur Kommunikation untereinander.

**Eintrag 2:** Mittwoch, 26.02.2020
Vor der ersten Übungsstunde trafen wir uns, um uns Gedanken zu machen über unser Spiel. Wir beschliessen einige Veränderungen und Verfeinerungen, die Anna in einem Word-Dokument festhält. Der Name unseres Spiels wird "Bachelor's Chase" sein, und wir sind die "Chaser".
Als nächstes Treffen (neben den Vorlesungsstunden) wird Do, 5. März, 10 Uhr an der Spiegelgasse festgelegt.
Vorläufig gelten folgende Zuständigkeiten:

- Anna bereitet die PowerPoint für den Meilenstein 1 vor, und den Gantt-Projektplan.
- Meipei macht das Mock-up unseres Spiels.
- Johannes wird während der ganzen Projektdauer das Tagebuch betreuen.
- Adrian fängt mit dem Networking an.

**Eintrag 3:** Freitag, 28.02.2020
Meipei fügt unserem Repository ein gitignore-File hinzu und löscht die überflüssigen Dateien.
Adrian wird einen ersten Entwurf des Netzwerkprotokolls anfertigen und ins Repository stellen.

**Eintrag 4:** Donnerstag, 05.03.2020
Treffen der gesamten Gruppe, um den Meilenstein 1 vorzubereiten. Wir arbeiten intensiv an der Powerpoint-Präsentation. Dabei ist zum Vorschein gekommen, dass wir den Spielablauf und die Regeln präzisieren müssen.

Dazu ist es auch nötig, uns Gedanken zu machen über das Netzwerkprotokoll, und was für Züge und Befehle möglich sein sollen.
Meipei wird weiter am Mockup arbeiten, und Johannes wird das gitignore-File anpassen.
Die Arbeitsaufteilung wird folgendermassen präzisiert:

- Server: Adrian und Meipei
- Client: Johannes und Anna
- Tagebuch: Johannes
- Präsentationen: Anna
- GUI: Meipei und Anna

Nächstes Treffen:

- Fr, 06. März, 14.00 Uhr im Gruppenarbeitsraum Spiegelgasse: Präsentation weitermachen, Netzwerkprotokoll besprechen
- Mo, 09. März, 08.30 Uhr im Gruppenarbeitsraum Spiegelgasse: Netzwerkprotokoll, Code produzieren für Meilenstein 2
- Mi, 11. März, 08.30 Uhr im Gruppenarbeitsraum Spiegelgasse: Präsentation einüben

**Eintrag 5:** Freitag, 06.03.2020
Treffen von Meipei, Adrian und Anna. Johannes ist krank und steht deshalb nur per WhatsApp-Chat zur Verfügung.
Wir legen folgende Aufteilung für die Präsentation fest:

- Anna: Spielidee, Spielbeschreibung/Regeln
- Mei: Karten, Mockup
- Adrian: Anforderungen, Client/Server
- Johannes: Organisation/Software, Fragen/Abschluss

Die Präsentation ist jetzt fertig.
Mei hat das gitignore mithilfe des Mails des Tutors angepasst.
Adrian hat angefangen, Server und Client zu implementieren.
Anna hat das gantt fertiggestellt.

**Eintrag 6:** Montag, 09.03.2020
Treffen von Meipei, Adrian und Anna. Johannes ist krank und schaltet sich per Skype zu.
Wir übten die Präsentation ein und stoppten die Zeit. Für die kommenden Tage gilt die folgende Arbeitsteilung:

- Mei und Adrian beginnen, den Chat zu programmieren.
- Anna beginnt mit einem Client und versucht es an den Chat von Mei und Adrian anzupassen.
- Johannes beginnt, das Ping-Pong zu programmieren.

**Eintrag 7:** Mittwoch, 11.03.2020
Treffen der gesamten Gruppe.
Wir einigen uns darauf, Englisch zu verwenden für alle Variablen- und Methodennamen, Kommentare im Programm, und auch für die Anzeigetexte des Games. Das einzige, was Deutsch bleibt, sind die Commit messages, weil wir dort schon auf Deutsch begonnen haben.

**Eintrag 8:** Donnerstag, 12.03.2020
Wir treffen uns in einem Sitzungsraum, um den Vortrag aufzunehmen. Es klappt alles ausgezeichnet, sodass wir nach kurzer Zeit bereits die fertige Video-Datei haben mit den Teilen von allen Gruppenmitgliedern.

**Eintrag 9:** Montag, 16.03.2020
Telefonkonferenz auf Discord.
Der Handler hat noch kleinere Probleme, Adrian arbeitet weiter daran. Die anderen Klassen müssen irgendwann noch angepasst werden, dass sie den Handler benutzen, statt direkt miteinander zu kommunizieren.
Der Chat funktioniert schon recht gut, ausser dass Umlaute noch nicht funktionieren.
Das Netzwerkprotokoll steht grösstenteils, und wird in den kommenden Tagen von Adrian in seinem Handler implementiert werden.
Mei kümmert sich um die Usernamen.
Anna kümmert sich um Hamachi.
Johannes verschiebt das Login und Logout vom Chat zum Handler.

Wir setzen uns Freitag, 20. März als Deadline für alle Anforderungen von Milestone 2. Sollte dann etwas nicht klappen, haben wir noch das Wochenende als Puffer.

**Eintrag 10:** Mittwoch, 18.03.2020
Übungsstunde per Zoom. Lange Diskussion über den Handler. Leider ist es uns noch unklar, wie wir die Kommunikation der Clients mit dem Server organisieren. Wir verbringen einen Grossteil des Nachmittags beim gemeinsamen Programmieren, d.h. Johannes teilt seinen Bildschirm und schreibt Code, und alle vier diskutieren darüber, wie der Code geschrieben/abgeändert werden soll.
Schliesslich vertagen wir das weitere Vorgehen auf morgen 13 Uhr (Ausweichtermin, weil Johannes eine Terminkollision hat: 16 Uhr)

**Eintrag 11:** Donnerstag, 19.03.2020
Wir beginnen um 13 Uhr ein Zoom-Meeting. Mei hat auf
https://github.com/RuthRainbow/Chat-Server/ ein gutes Beispiel gefunden für das, was wir brauchen. Nun versuchen wir unseren vorhandenen Code diesem Beispiel gemäss anzupassen.
Leider müssen wir unsere Versuche aufgeben, unseren bisherigen Code zu verbessern. Wir beginnen bei Null, und einigen uns auf folgendes Modell:
Sowohl Server als auch Client werden als statische Klassen von der Konsole aus gestartet; sie haben ihren Code jeweils in ihrer main-Methode.
Der Client startet aus der main-Methode heraus ein ClientHandler-Objekt, das die gesamte Netzwerkkommunikation übernimmt. Ebenfalls startet der Client einen ChatClient, welcher für den Chat verantwortlich ist.
Sobald der Server die einkommende Verbindung erhalten hat, erstellt er ein ServerHandler-Objekt, das fortan für die Kommunikation zwischen dem Server und *diesem* Client zuständig ist. Ebenso startet er einen Chatserver.
Die beiden Handler-Klassen haben zwei Aufgaben: Erstens lauschen sie in ihrer run()-Methode auf eingehende Nachrichten, entschlüsseln diese ankommenden Strings gemäss Netzwerkprotokoll, und rufen die entsprechenden Methoden auf, um die Informationen weiterzuleiten. Zweitens haben sie Methoden, die vom dahinterstehenden Server/Client aufgerufen werden können, um eine Nachricht gemäss Netzwerkprotokoll rauszuschicken.