Programmierprojekt (Gruppe 15) - Diary

Skip Bo

28. Februar:

Entscheid für das Spiel: Skip-Bo. Lesen der Regeln, erste und sehr abstrakte Überlegungen dazu, was zu implementieren ist.

3. März:

Erstes Treffen, um Zeitplan festzulegen und Aufgaben zu verteilen. Probleme mit git über Intellij. Erarbeiten eines Grundverständnisses für Server-Client Architekturen.

Konkretere Überlegungen zu benötigten Klassen und Reihenfolge der Implementierung. Grundriss von GanttProject erstellt.

3. März - 10. März:

Arbeiten an der Präsentation. Jedes Gruppenmitglied macht seinen Teil der PowerPoint. Erstes individuelles Experimentieren, um selbst einen einfachen Server und Client mit Java zu programmieren.

Erster Versuch, einen simplen Chat zu programmieren.

10. März:

Erstellen eines genauen Zeitplans inklusive Verteilung der Aufgaben. Genaue Überlegungen zu Softwarearchitektur, inklusive Erstellen einer Grafik dazu.

Beenden von Mock-up und diversen anderen Vorbereitungen für die Präsentation am Donnerstag. Grosse Probleme mit Git: Synchronisation klappt bei einem Mitglied nicht.

Schreiben von erstem nützlichen Programmteil, einem Chat, welcher die Nachrichten auf dem Server ausgibt.

11. März:

Chat läuft. Überlegungen dazu, wie er in unser Endprogramm eingebaut werden wird. Feinschliff der Präsentation.

12. März:

Aufnahme der Meilenstein 1 Präsentation.

17. März:

Skype-Konferenz. Bestimmen der Struktur des Netzwerkprotokolls, da es den Grundbaustein für die Kommunikation zwischen Server und Client darstellt.

Überlegungen zu nötigen Befehlen für Chat und Spiel. Klären gewisse Fragen und Unsicherheiten innerhalb der Gruppe. Besprechen des weiteren Vorgehens.

17. - 22. März:

Intensive Code-Arbeit. Server und Client werden geschrieben inklusive jeweiligen Listeners.

Viele Erkenntnisse zu den Anforderungen der jeweiligen Klassen und der Verteilung der Aufgaben zwischen Client und Server: das Login mit dem Starten der Listeners ist komplizierter als gedacht, weil der Listener Thread gewisse Infos braucht, welche auf ersten Blick nur durch einen bereits bestehenden Listener gesammelt werden könnte. Erschwerte Kommunikation wegen Quarantäne.

20.März:

Bei den Spieloperationen sind einige Sachen erstellt worden. Player, Game und Card Klassen sind fast fertig geschrieben. Player Klasse ist besonders wichtig für die Server Implementation, und wir setzten hier Priorität. Kartenverteilung in der Klasse Game ist auch bereit. In einer Methode 'setupGame ()' werden zuerst alle Karten erstellt und dann weiter mit einer Random Variable an verschiedenen Spielern verteilt.

Die Stapelbildung geschieht in der Klasse Pile, die Handkarten der Spieler werden auch wie ein Pile behandelt, aber mit viel wenigen Einträgen. Alle Piles sind in verschiedenen ArrayLists gespeichert, da man die Grösse immer verändern kann!

21. März:

Server ist bereit und mehrere Clients können sich problemlos verbinden. Von der Client Seite ist es möglich, einen Username einzugeben. Server kann sich den Namen auch merken und alle Chat Messages der Clients aufnehmen. Leider ist das Zurückschicken der Nachrichten an alle Clients ein Problem. Die Clients können im Moment Nachrichten schicken, aber sehen die Nachrichten der anderen Clients nicht.

Nach einem langen Debug Session ist es klar, dass der Fehler bei Deklaration vom PrintWriter lag. Wir hatten den zweiten Parameter (autoFlush: true) vergessen; deswegen konnten die Nachrichten und Protokollbefehle nicht an alle Clients verschickt werden. Client-Client Kommunikation läuft zu diesem Zeitpunkt flüssig. Die Gültigkeit der Namen wird auch richtig überprüft vom Server, aber /change name Befehl gibt etwas Falsches aus. Einige Code Abschnitte müssen noch genauer kommentiert/beschrieben werden und das Netzwerkprotokoll muss noch besser dokumentiert werden.

22. März:

Weiterer Testing/Debug Session. /change name Befehlt funktioniert ohne weitere Probleme. Der Fehler lag bei einer falschen Deklaration. Bei genauem Nachlesen der Meilensteinpunkte, wurde uns klar, dass der System Username vom Client vorgeschlagen werden muss, also haben wir einige Sachen ändern müssen – bei uns passierte dies auf Seite Server.

Tagebuch besser und anders formattiert. Netzwerkprotokoll wird auch noch fertig geschrieben und schöner formattiert. Die Code Abschnitte, die noch nicht perfekt dokumentiert waren, wurden ergänzt, damit jeder Teil des Codes für jeden verständlich ist. Ob der Client seine eigene Nachrichten ein zweites Mal sehen soll, wurde zu einem Diskussionsthema. Auf dem Terminal sah dies etwas komisch aus, also entschieden wir uns, dass nur die Chatnachrichten anderer Clients auf Terminal sichtbar sein werden, und auf GUI alle Nachrichten im Chat Box zu sehen sind.

Erster Test mit Himachi erfolgreich, obwohl wir zu Beginn nicht ganz verstanden haben, wie Himachi funktioniert. Chat zwischen Gruppenmitglieder funktioniert einwandfrei, aber wir haben manchmal bei Rohan immer wieder eine Fehleranmeldung bei /change name Befehl bekomment. Im Moment nicht ganz klar wieso, da diese beim Rest des Teams nie vorkam!

Letzte Tests und kleine Verbesserungen werden noch im Laufe des Sonntags durchgeführt, aber die Mehrheit der Meilensteinpunkte sind unsererseits erfüllt!

Ende MS2

27. März:

Erstes Treffen nach der MS2 Präsentation. Unsere MS2 Bewertung ist in der Zwischenzeit angekommen, und wir freuen uns über die Verbesserung von MS1. Diskussionsthema für das Treffen ist das weitere Aufteilen der MS3 Meilensteinpunkte. Zu Beginn werden alle Punkte sorgfältig durchgelesen und verinnerlicht. Die weiteren Aufgaben kann man in 4 Teile zerlegen: Chat, Netzwerkprotokoll in Bezug zum Spiel, Spielimplementation und Manual/Präsentation. Da Guillaume und Manuela den grossen Teil der Server/Client Aufgaben übernommen hatten, werden sie weiter die Seite des Spieles erweitern, während Janni und Rohan am Spiel weiterarbeiten. Da ein Chat GUI schon als Experiment während MS2 entstanden ist, kann dies weiterhin benutzt werden.

Es wird auch klar, dass wir viel weniger Zeit haben als ursprünglich gedacht, da der Abgagetermin der 6. April ist, und nicht der 16.! Viel Codearbeit muss in einer Woche durchgeführt werden, aber an diesem Termin wurde das weiter Vorgehen gut aufgeteit.

29. März

Der Grossteil der Spielregelimplementation ist in der Zwischenzeit zu Stande gekommen. Viel Diskussion zwischen Rohan und Guillaume bei Implementierung des Server/Game Zwischenspiels, wie die Information vom Client über den Server zur Klasse Game ankommt. Langsam wird es klar, dass ‘int id’ für die Klassifikation von Spielern gar nicht so wichtig ist wie gedacht.

31. März

Meeting vor der letzten Übungsstunde, um mögliche Fragen zu besprechen. Chat GUI läuft problemlos und Chat Nachrichten werden korrekt geschickt. Leider ist das Ausloggen und das Schliessen des Chat Fensters noch ein Problem. Bei Logout wird zwar der Client ausgeloggt, aber das Fenster bleibt noch offen und akzeptiert keine Inputs mehr. Diskussion über Instruction Manual und das Design Layout wird gehalten. Eine grobe

Über Build Scripts und Java Libraries muss noch Einiges gelernt werden, und wird bei nächstem Treffen das grosse Thema sein.

1. April

Intensive Code Arbeit an der Klasse Game. Die /play Commands müssen noch implementiert werden, bevor das eigentliche Testen des Spiels beginnen kann. In der Übungsstunde werden External Libraries besprochen und es wird entschieden, den Logger zu implementieren. Die LogOut Probleme sind alle repariert worden, und das Chat Fenster funktioniert perfekt.

3. April

/play Command wurde richtig implementiert und das Testen kann beginnen! Schon zu Beginn wird es klar, dass ArrayList.toString überhaupt nicht so funktiniert wie gedacht, also müssen schon print methoden für die einzelnen Piles implementiert werden. Weiterhin sind noch einige NPEs zu beheben. Oft wurde im Code nach leeren ArrayLists Zugriff gefragt, und alle die Fehler müssen angeschaut werden. Einige Stunden später ist das Spiel spielbereit, ohne Fehler. Einiges der playToMiddle und playToDiscard Methoden mussten korrigiert werden, die Fallunterscheidung funktionierten nicht wie gedacht. Aber im Grossen und Ganzen, keine “unmögliche” Fehler, sondern relativ simple Fehler waren vorhanden. Das Format des Manuals steht auch bereit, aber die einzelnen Abschnitte müssen gefüllt werden.

5. April

Manual ist fast bereit. Text is fertig, aber Word Formattierung scheint eine Herausforderung zu sein. Logger ist auch bereit und funktioniert ohne Mühe. Das .jar File und java doc zu generieren ist ein weiteres Problem, und wir konzentrieren uns auf diese zwei Sachen. Nach mehreren Fehlversuchen und viel Recherche, gelingt es Manuela, java doc und jar zu generieren und den Server und Client dadurch auszuführen. Leider klaptt das Gleiche nicht bei allen Computern des Teams. Wir vermuten, dass es ein Classpath Fehler bei dem Rest sei, aber bei Manuela funktioniert das Ganze problemlos. Ein weiteres Problem ist noch die Methode getGamesList(), die alle Games zum gewissen Zeitpunkt anzeigen soll. Der Grossteil der Meilensteinpunkte sind abgedeckt für den Check-In am 6. April, aber nicht ganz alles. Leider sind einige Sachen, die wir als klein ansahen, ein grösseres Problem als vermutet!

10. April

Erste Diskussionene über die Präsentation am 16. April werden gehalten. Eine Drag and Drop Klasse wurde erstellt um Methoden zu testen, um Karten zu spielen. Es wurde aber bald klar, dass dieses Drag and Drop nicht so funktioniert, wie wir uns das vorgestellt haben: Man konnte die Karte nur etwa jedes fünfte mal nehmen. Deshalb implementieren wir jetzt, wie ursprünglich geplant, ein GUI mit Knöpfen. Es wurden erste Entwürfe von Karten erstellt um das GUI zu testen.

12. April

Die Knöpfe funktionieren einwandfrei und geben die richtigen Befehle an den Server weiter. Man kann aber noch jeden Stapel anklicken und somit falsche Züge machen, die beim Server zu Problemen führen können. Es ist geplant, dass später der Client falsche Züge gar nicht erst zulässt und die Spieler damit auch besser sehen könne, wohin sie die Karte bewegen können. Unsere Prioritäten sind aber momentan, dass der Client die Karten auf dem Spielfeld richtig updated. Dies scheint komplizierter als wir angenommen haben, da der Client nicht weiss, welcher Spieler welche Karten hat und bis jetzt fast nur mit Indizes von Stapeln gearbeitet wurde. Wir müssen nochmals überdenken, welche Informationen der Server dem Client schicken muss, damit der Client alle Karten richtig darstellen kann.

14. April

Meeting zwei Tage vor der Präsentation. Ziel ist es, die Karten des Spielfeldes richtig darzustellen um das Spiel bei der Präsentation im GUI zu spielen und die Präsentation zu üben. Der Server übergibt dem Client jetzt auch die benötigten Informationen über Karten, damit sie vom Client dargestellt werden können. Ein neuer Protokoll Befehl CHECK wurde hinzugefügt, um nach jeder Änderung der Handkarten den Client wieder aktualisieren. Durch die zusätzlichen Informationen über Karten, die jetzt geschickt werden, gab es viele Probleme mit Indizes und es mussten ein paar Methoden nochmals genau angeschaut werden, um die richtigen Indizes zu benutzten. Zudem haben wir festgestellt, dass der Server dem Client Informationen in der falschen Reihenfolge schickt und somit die Handkarten zu früh aktualisiert wurden. All die Probleme haben viel mehr Zeit beansprucht als wir dachten, deshalb konnten wir die Präsentation nicht mehr wirklich üben. Das Spiel ist jetzt aber bereit für die Präsentation und wir werden die Präsentation morgen üben.

17. April

Bis jetzt wurden die Komponenten vom Spiel auf die Lobby Elemente addiert. Wir mussten feststellen, dass es so sehr mühsam ist, beim Spielende alle Komponenten vom Spiel wieder zu entfernen. Deshalb hat das Spiel jetzt seine eigene layeredPane. Durch die layeredPane können nun auch mehrere Knöpfe übereinander dargestellt werden.

18. April

Im Chat kann jetzt in Farbe geschrieben werden. Allerdings gibts jetzt teilweise Exceptions von AWT. Wir wissen nicht woher sie kommen und suchen nach Lösungen. Viele Ideen von Stack Overflow wurden ausprobiert, doch der Fehler ist immernoch da.

19.-21. April

Erstes Einlesen zu JUnit und der Funktionsweise der Unit-Test sowie Gespräch innerhalb der Gruppe, welche Komponente des Spiels getestet werden soll. Die Wahl liegt zuerst bei der Spiellogik selbst, wird aber schnell zur ProtocolExecutor Klasse gewechselt: Tests über diese zentrale Klasse können auch in der Zukunft sehr nützlich sein – zum Beispiel um sicherzustellen, dass Änderungen im Code nicht zu Fehlern an anderen Stellen geführt haben. Diese Klasse zu testen führt unserer Meinung nach zum höchstmöglichen Code coverage (ein Begriff, den wir erst in der folgenden Übungsstunde richtig verstanden haben).

Erste Entwürfe der Tests werfen schnell Fragen auf: schon nur die Tatsache, dass für die Unit-Tests ein Server und Clients gestartet werden müssen und die Methoden nicht unabhängig davon getestet werden können, ist uns zu Beginn nicht klar. Danach kommen Schwierigkeiten auf, da uns nicht bewusst ist, dass Unit-Tests parallel und nicht sequenziell durchgeführt werden. Deshalb muss für jede Testmethode ein neues „Szenario“ bzw. Framework aufgesetzt werden, in dem dann zu Testen ist.

22. April

Dieses Problem wird mithilfe des Tipps unseres Tutors, eine @Before Methode zu implementieren, rasch gelöst. Allerdings müssen dann parallel mehrere Server gestartet werden, was ein umschreiben unseres bisher noch static gebliebenen Servers verlangt.
Daraufhin verläuft das Schreiben der Tests ohne weitere unerwartete Hürden und verbleibt beim stundenlangen Suchen und Finden von Fehlern, sowohl im Programm als auch in den Tests selbst, und schliesslich laufen alle 17 Methoden der ProtocolExecutorTest Klasse. Nur die Methode, welche den Netwerkprotokollbefehl DISPL testet, schlägt noch fehl. Der Grund dafür ist uns zu diesem Zeitpunkt noch nicht klar, wird jedoch in den nächsten Tagen untersucht.

Mehrere Karten auf dem Discard Stapel sind nun sichtbar. Die Anzahl der sichtbaren Karten wurde aber auf 5 beschränkt, da sie sonst zu viel Platz einnehmen. Da die grunsätzlichen Dinge vom Client und vom Server um das Spiel zu spielen funktionieren, wird das GUI noch verschönert und einzelne kleinere Fehler behoben. Wir haben erfolglos versucht, den AWT Fehler durch die invokeLater Methode zu beheben. Auch nach Diskussionen mit unserem Tutor wird eine Lösung nicht klar.

26. April

Tests sind alle fast fertig geschrieben, und wir spielen das Spiel sorgfältig durch und testen jegliche Edge Cases, oder mögliche Fehler und schreiben alle Verbesserungsmöglichkeiten auf. Ideen für Soundimplementation ins Spiel werden entwickelt. Um Whisper Chat in ein Button einzupacken wirft noch einige Probleme auf, das Formattieren der Namen ist nicht ganz wie erwartet. Nach mehreren Fehlversuchen, gelingt es uns, dies auch noch zu schaffen und so für den MS4 bereit zu sein. Das Spiel ist problemlos über GUI spielbar, die Gewinner werden auch richtig ausgegeben.

Obwohl High Score Implementation etwas Zeit brauchte, ist diese auch bereit und wird in ein Text File eingetragen! Das Repo wird ein letztes Mal angeschaut, damit wir keine unnötigen Files darauf haben, und alle benötigten Dokumente hochgeladen sind!

Ende MS4

6. Mai

Das Tutorial ist fertig. Am Anfang gab es ein paar Probleme um ein Tutorial Objekt zu erstellen, da es schwierig zu kontrollieren ist, wann genau Swing Objekte erstellt und die Tutorial Klasse danach für paar Sekunden schlafen sollte. Dieses Problem wurde mit dem Timer gelöst. Der Timer stellte sich im Nachhinein auch in anderen Klassen als sehr nützlich heraus um Tasks später auszuführen. Videos können leider immernoch nicht abgespielt werden. JavaFx hätte wäre wahrschienlich eine gute Library um mp4 Dateien abzuspielen, jedoch gab es Probleme, weil wir Swing benutzten. Die Java Media Library konnte auch unsere .avi Datei nicht lesen. Ansonsten fanden wir die Libraries sehr kompliziert und haben die Dokumentation nicht wirklich verstanden. Deswegen mussten wir dieses Projekt leider aufgeben, da es nur Bonuspunkte sind und wir damit zu viel Zeit verlieren würden.

8. Mai

Wie in der Übungsstunde besprochen wurden noch einige zusätzliche Tests geschrieben, da die bisherigen eher Integrations- als Unit-Tests waren. Dafür wurde das Interface NPWListener erstellt, sowie eine neue Klasse testingSBL. Diese besitzt dieselbe Logik wie die normale SBListener Klasse, wurde jedoch ausschliesslich mit Testen im Sinn geschrieben. Zum Beispiel werden anstatt Methoden des ProtocolExecutors aufzurufen ein String zum Namen dieser Methode gesetzt und der Input kommt statt über ein Socket durch einen PipedInputstream. Somit kann man die SBListener Klasse wirklich einzeln testen, ohne die Tests in laufende Server und Clients einbetten zu müssen.

09. - 10. Mai

Die letzten Features werden dem Spiel zugefügt: eine Hintergrundmusik, Töne zu den Button- und Card Clicks sowie Cheat codes und der Bestrafung derer. Die MP3SPI Library zum Abspielen vom mp3-Dateien bereitet zuerst einige Schwierigkeiten, welche jedoch mit der Hilfe von Forumsbeiträgen erfolgreich überwunden werden können. Ausserdem wird der Fall behandelt, wenn ein Spieler ein Game verlässt, ohne jedoch die App zu verlassen, da bisher ein komplettes Logout die einzige Möglichkeit zum Verlassen eines Games war. Auch auf GUI-Seite müssen noch gewisse Änderungen angebracht werden, zum Beispiel zum Muten der Musik und Soundeffekte. Ausserdem wird der Entwurf für den Trailer gemacht.

11. - 13. Mai

Auch wenn ab jetzt nur noch „der letzte Schliff“ zu tun ist (und die Präsentation), braucht dies mehr Zeit als gedacht. Wir spielen das Spiel mehrere Male, und immer wieder tauchen Fehler auf - speziell beim Verlassen eines Games tauchen immer wieder Fälle auf, welche wir nicht beachtet hatten. Da beim Verlassen des Games das ArrayList mit den verbleibenden Spielern verschoben wird, braucht man je nach dem eine Anpassung der Variable, welche angibt, wer am Zug ist. Ausserdem muss natürlich der Zug beendet werden, falls der Spieler, der das Game verlässt, selbst gerade am Zug war. Speziell durch das ursprüngliche Design der Methoden, welche das Verlassen des Games verarbeiten – welche wir auf zwei Klassen aufgeteilt hatten, weil das uns zu Beginn am Logischsten schien – werden Anpassungen erschwert. Schlussendlich schaffen wir es, alle Operationen in der richtigen Reihenfolge und mit Berücksichtigung aller Spezialfälle durchzuführen: die eine Methode, welche den Spieler aus dem Spiel nimmt, gibt ein boolean Array an die andere Methode zurück, welche sich darum kümmert, alle anderen Spieler darüber zu informieren.

Obwohl das Beheben dieser kleinen Fehler nur sehr wenig Codearbeit erfordert, stellt sich schnell heraus, dass Veränderungen in einem für uns bereits so komplexen Code nur mit äusserster Sorgfalt durchgeführt können. Nur so können wir gleichzeitig kollaterale Schäden verhindern und alle Codeteile finden, welche angepasst werden müssen. Auch die Cheat codes brauchen einige Anpassungen – von einer eigenen Methode in der Game Klasse zu einem neuen Feld in der Player Klasse, welche angibt, ob der Spieler bereits einen Cheat benutzt hat.

Gameplay wird aufgenommen und der Trailer wird erstellt. Der QA Report wird zum Schluss im Plenum diskutiert und gemeinsam abgeschlossen. Dadurch kann jeder seine Sicht zu den QA Massnahmen einbringen und seine Meinung dazu äussern, was uns die Messungen zeigen. Parallel werden die Slides der PowerPoint abgeschlossen und anschliessend die Präsentation noch einige Male geübt. Ein Problem mit der JAR Datei taucht noch auf, welches nicht richtig auf die Ressourcen zugreift: Sound und Icons werden nicht geladen. In der Übungsstunde wird auch noch dieses Problem behoben.