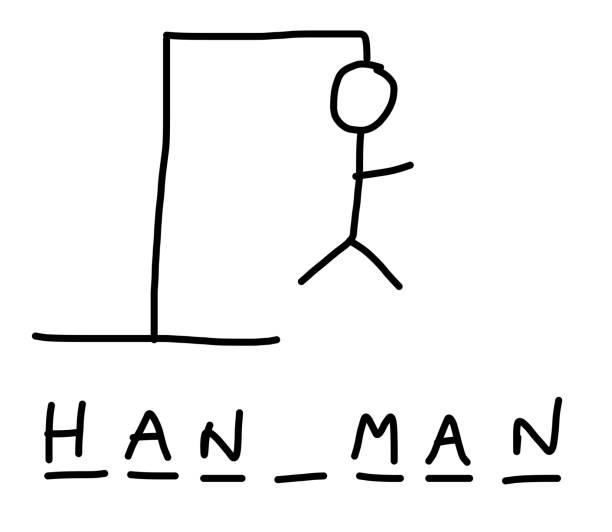
**Dokumentation zum Projekt**

**GALGENMÄNNCHEN**



Projektarbeit von

Celine Gugel Nr. 2707682

Lena Steinbrink Nr. 5513041

Valentin Müller Nr. 5461128

Jannik Oßwald Nr. 1527904

und Thi Tuong Vy Nguyen Nr. 8610085

Inhalt

[1. Ideenfindung 2](#_Toc122348663)

[**2.** Spielerklärung 2](#_Toc122348664)

[3. Benötigte Funktionen 2](#_Toc122348665)

[3.1 Auswahl Spielmodus​ 2](#_Toc122348666)

[3.2 Buchstabeneingabe 3](#_Toc122348667)

[3.3 Ausgabe der Konsole 3](#_Toc122348668)

[3.4 Spielablauf 3](#_Toc122348669)

[4. Spielaufbau 4](#_Toc122348670)

[4.1 Draw Hangman 4](#_Toc122348671)

[4.2 Hangman Java 4](#_Toc122348672)

[4.3 Spiel.java 6](#_Toc122348673)

[**5.** Zusammenarbeit 7](#_Toc122348674)

# Ideenfindung

Um eine Idee davon zu bekommen, welches Spiel wir umsetzen wollen, haben wir uns zunächst selbst informiert, um uns anschließend für ein Spiel zu entscheiden. Zuerst kam uns hierbei das Spiel Flappy Bird in den Sinn. Da hierzu jedoch eine Grafik benötigt werden würde und wir vor allem das in den Vorlesungen angeeignete Wissen nutzen wollten, haben wir uns schlussendlich für das altbekannte Spiel Galgenmännchen entschieden.

1. **Spielerklärung**

Bei dem Spiel Galgenmännchen muss ein Begriff erraten werden. Es sind jedoch nur Striche in der Anzahl der zu erratenen Buchstaben des Wortes gegeben. Der Spieler muss nun einen Buchstaben eingeben. Ist der Buchstabe im Wort vorhanden, erscheint er an der passenden Stelle. Bei falsch geratenen Buchstaben bildet sich nach und nach eine Abbildung eines Galgens. Wurden zu viele Buchstaben falsch genannt, sodass der Galgen vervollständigt ist, so hat der Spieler das Spiel verloren. Sollte der Spieler jedoch vorher das Wort erraten, so kann er sich als Sieger verkünden.

# Benötigte Funktionen

Damit wir mit der Programmierung beginnen können, muss zunächst festgelegt werden, wie wir das Spiel aufbauen wollen.

3.1 Auswahl Spielmodus​

Wenn der Spieler das Spiel startet, soll er zunächst die Auswahl haben, ob er gegen den Computer spielen möchte oder ein Spiel mit eigener Worteingabe starten möchte. Bei einem Spiel gegen den Computer soll aus einem Array mit vorgegebenen Wörtern ein Wort zufällig ausgegeben werden und das Spiel startet. Wird ein Spiel mit eigener Worteingabe gewählt, so wird über die Scanner-Funktion ein Wort als String abgefragt. Damit bei dem Beginn des Spiels die Spielauswahl nicht mehr in der Konsole zu sehen ist, fügen wir 60 Leerzeichen ein. Es gibt auch noch einen dritten Modus, mit welchem man das Spiel verlassen kann.

## 3.2 Buchstabeneingabe

Wir legen eine neue Klasse Spiel. Somit kann man die Funktion zur Buchstabeneingabe programmieren mit den verschiedenen Bedingungen. Die erste Bedingung ist ein Char, damit nur der erste Buchstabe genommen wird. Die zwei anderen Bedingungen sind, dass das Wort zum einen mindestens zwei Zeichen hat und zum anderen, dass es keine Leerzeichen oder Sonderzeichen enthält. Zudem braucht man eine Funktion, die den Buchstaben mit dem gesuchten Wort vergleicht und uns später diesen schon benutzten Buchstaben auch ausgibt. Außerdem benötigen wir ein Array mit den Buchstaben an der entsprechenden Stelle vom Wort für die Ausgabe in der Konsole.

## Ausgabe der Konsole

Wenn die Eingabe des Spielers richtig ist, soll „Hurra! Deine Eingabe war richtig!” auf der Konsole erscheinen. „Falsche Eingabe“ erscheint, wenn Eingabe falsch ist​. Um den Spieler zu informieren, dass das Spiel soeben gestartet wurde, soll "Das Spiel Hangman wurde gestartet! “ ausgegeben werden. Um den Nutzer die Auswahlmöglichkeiten zu zeigen wird "Spielmodus 1: Spiele gegen den Computer"​ und "Spielmodus 2: Gib ein eigenes Wort ein (Multiplayer)"​ angezeigt. Falls der Spieler das Spiel frühzeitig beenden möchte, sollte es eine Option geben, mit welcher er das Spiel beenden kann. Beispielsweise „Wenn du das Spiel beenden möchtest, kannst du exit eingeben.“ Hat der Spieler das Spiel nicht beendet und einen der beiden Spielmodi ausgewählt, erscheint „Es wurde Spielmodus 1/2 gewählt“, um dem Spieler mitzuteilen, in welchem Spielmodus er sich befindet​. Als nächstes soll „Ein Buchstabe bitte:“ erscheinen, als Aufforderung für den Spieler ein Wort einzugeben​. „Das Spiel wird beendet“, wenn der Spieler das Spiel beendet.

## 3.4 Spielablauf

Beispiele für benötigte variablen sind unter anderem *anzahlFalscheBuchstaben (int), spielVerloren (boolean), spielGewonnen (boolean), arrFalscheBuchstaben, arrGegebeneWörter.* Mögliche Zusatzfunktionen sind auch, dass die Konsole eine Nachricht ausgibt bei falscher Eingabe oder wenn man Sonderzeichen/Zahlen eingibt.

# Spielaufbau

## 4.1 Draw Hangman

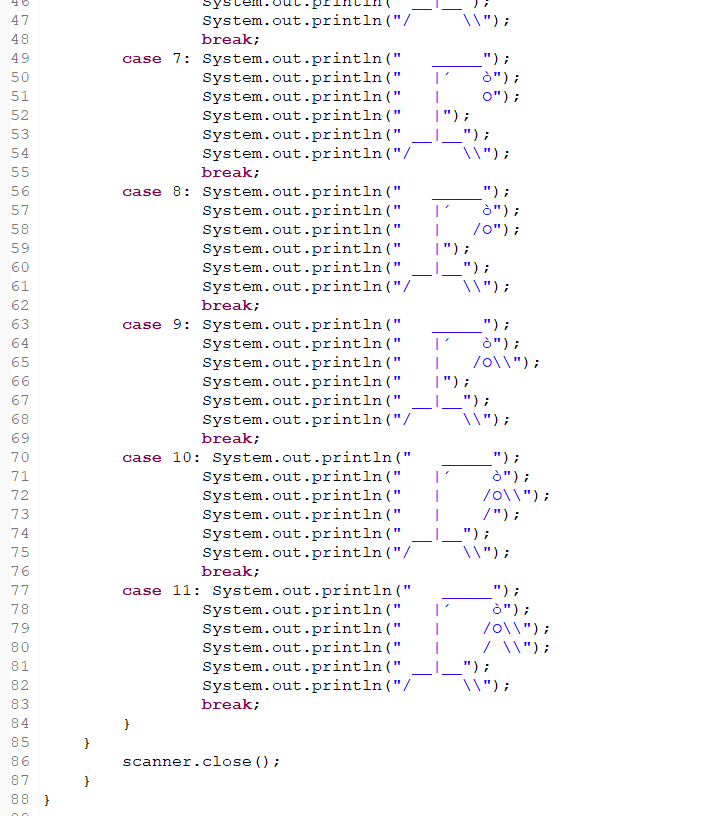
Wir haben uns Gedanken darüber gemacht, wie wir den Galgen und die kleine Person darstellen möchten. Dazu haben wir zunächst einige Optionen ausprobiert und uns am Ende für die Folgende entschieden. Um diese nun darstellen zu können, haben wir zunächst eine Klasse namens DrawHangman erzeugt.

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner scanner = new Scanner(System.in);

int a=0;

**** while (a<13) {

System.out.println("get new number:");

a = scanner.nextInt();

switch (a) {

Wir haben eine switch-Anweisung angelegt und 11 cases erstellt. Bei jedem case kommt immer ein weiterer Strich für den Galgen und das Strichmännchen hinzu. Durch System.out.println werden die Striche abgebildet. Nach jedem break wird die Schleife komplett verlassen und nach der Schleife fortgesetzt. Ist der letzte case durchlaufen, so wird das Spiel durch scanner.close() beendet.

## 4.2 Hangman Java

Wir haben eine Klasse Namens Hangman erstellt und einen Scanner angelegt. Zuerst soll es eine Abfrage des Spielmodi geben. Durch System.out.println angezeigt, dass das Spiel gestartet wurde. In der nächsten Zeile wird der „Spielmodus1: Gegen den Computer“ und eine Zeile darunter „Spielmodus 2: Gib ein eigenes Wort (Multiplayer)“ ausgegeben.

Damit es keine Fehlermeldung gibt, wird \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Um eine Auswahl zwischen Spielmodus 1 und 2 zu gewährleisten, haben wir eine switch-Anweisung, also eine Mehr-Wege-Verzweigung angelegt. Der Ausdruck „spielmodus“ lässt sich somit in case 1 und 2 untergliedern. Case 1 steht für den Spielmodus 1. Es wird angezeigt „Es wurde Spielmodus 1 gewählt“. Wir haben eine Auswahl an Wörtern hinterlegt und eine zufällige Zahl für die Wortauswahl. Wichtig hierbei ist es, die Wörterlänge als Maximum anzugeben, da \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Damit der Computer nun ein zufälliges Wort von den vorher angelegten Wörtern aussucht, haben wir eine random Methode angelegt.

Durch System.out.println(woerter[randomNumber] wird nun eines der hinterlegten Wörter angezeigt. Die Variable input haben wir auf das zufällig wewählte Wort gesetzt, weil \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In Case zwei ist der Spieler selbst für die Erstellung eines zu ratenden Wortes verantwortlich. Wieder wird ausgegeben, in welchem Spielmodus man sich befindet, jedoch wird der Spieler aufgefordert ein Wort einzugeben. Die Variable wird auf das eingegebene Wort gesetzt.

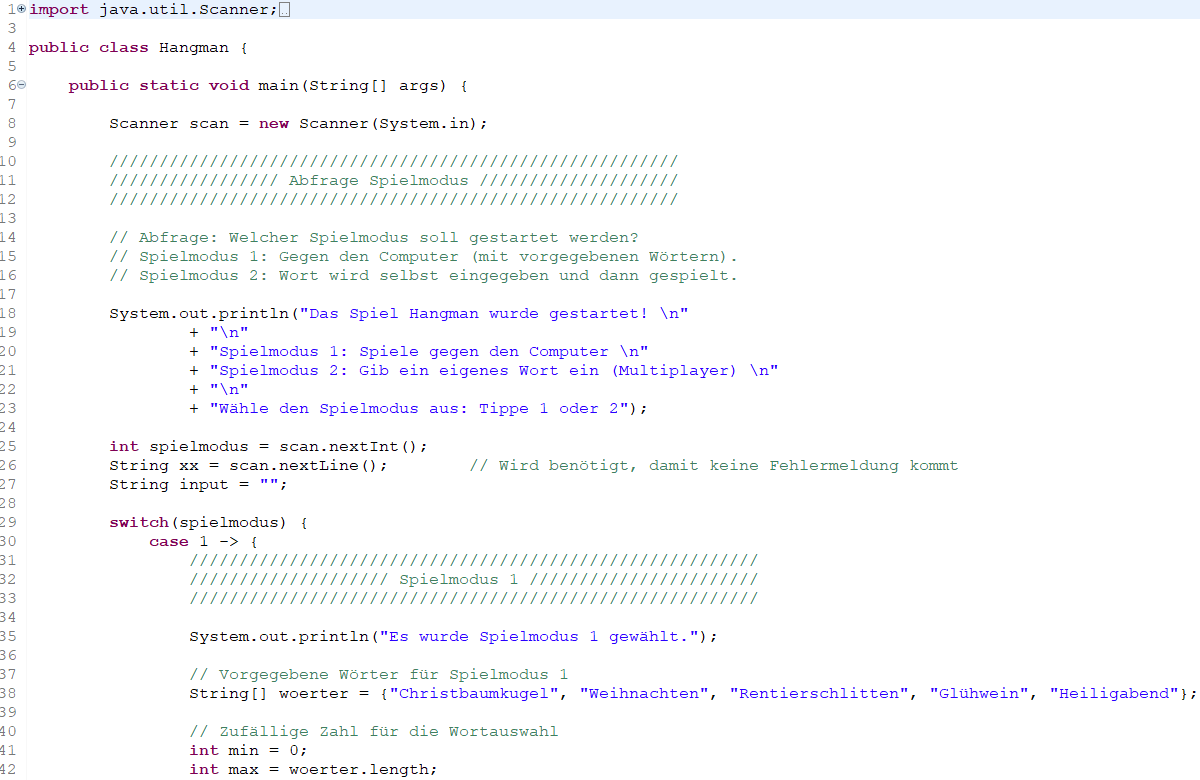
default -> System.out.println("Falsche Eingabe!")

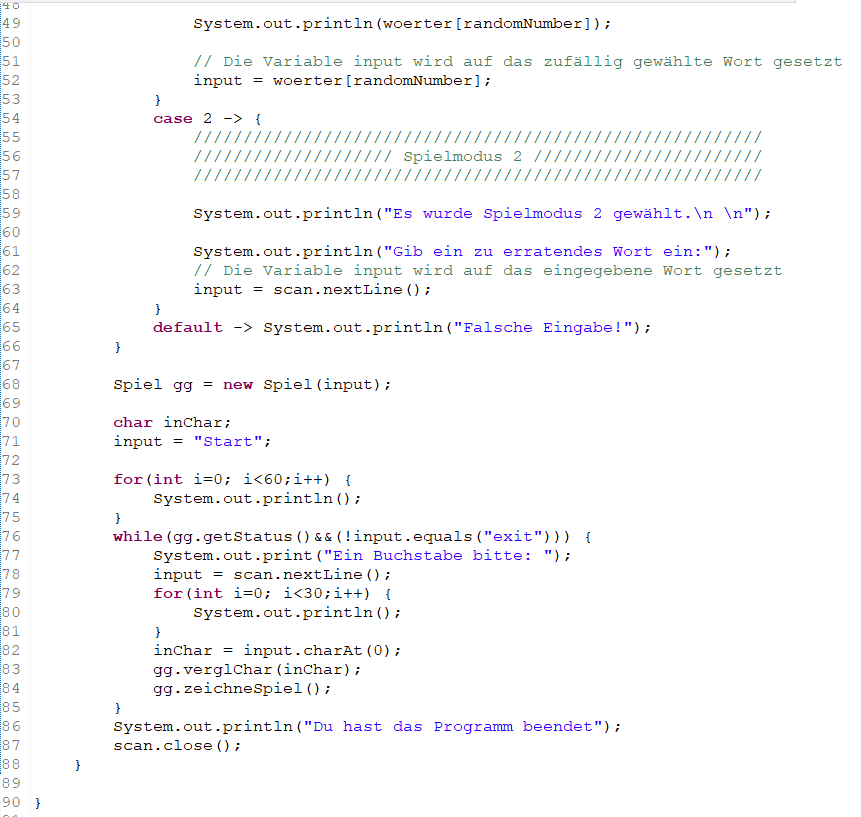
Spiel gg = new Spiel(input);

char inChar;

input = "Start";

Damit der Spieler immer weiter raten kann, haben wir eine for-Schleife angelegt.

Zudem haben wir die abweisende while-Schleife angelegt, welche, so lange der Spieler nicht „exit“ schreibt, in Kraft tritt. Während der Spieler sich in dieser Schleife befindet, wird „Ein Buchstabe bitte“ beim Spieler angezeigt. Sollte der Spieler „exit schreiben, so wird „Du hast das Spiel beendet“ angezeigt und das Spiel wird durch „scan.close();“ geschlossen.



## 4.3 Spiel.java

Wir haben eine Klasse Spiel erstellt. In dieser haben wir ein String gesWort angelegt, welcher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Durch int fehlerAnzahl = 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. Damit \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ haben wir einen char [] wortArr angelegt.

private ArrayList <Character> schonVersucht = new ArrayList<Character>();

private boolean gewonnen = false;

private boolean verloren = false;

Mit private static final String validLetters = "abcdefghijklmnopqrstuvwxyzäöü" haben wir festgelegt, welche Buchstaben eingegeben werden können.

Spiel(String gesWort){

this.gesWort = gesWort.toLowerCase();

this.wortArr = new char[gesWort.length()];

for(int i = 0; i<wortArr.length; i++) {

wortArr[i] = '\_';

}

}

Damit der eingegebene Buchstabe mit dem gesuchten Wort und dem schon versuchten Buchstaben verglichen werden kann, haben wir public void verglChar(char eingabe) verwendet.

eingabe = Character.toLowerCase(eingabe);

if(validateInput(eingabe)) {

if (schonVersucht(eingabe)) {

System.out.println("Diesen Buchstaben hast du schon versucht");

}else if(contains(eingabe, gesWort)) {

for(int i = 0; i<gesWort.length(); i++) {

if(eingabe == gesWort.charAt(i)){

wortArr[i]=eingabe;

}

}

}else{

fehlerAnzahl++;

schonVersucht.add(eingabe);

}

}else {

System.out.println("Eingabe ist nicht gueltig, bitte erneut versuchen");

}

}

Mit private boolean schonVersucht(char test) wird überprüft, ob der Buschstabe zuvor schonmal eingegeben wurde.

for(int i=0; i < schonVersucht.size(); i++){

if(test == schonVersucht.get(i)){

return true;

}

}

//schonVersucht.add(test);

return false;

}

Damit überprüft wird, ob das gesuchte Wort den Buchstaben enthält, wird private static boolean contains(char in, String pruefWert) benötigt.

for(int i =0; i<pruefWert.length(); i++) {

if(in == pruefWert.charAt(i)) {

return true;

}

}

return false;

}

private static boolean validateInput(char input) { überprueft, ob die Eingabe im definierten Wertebereich liegt.

input = Character.toLowerCase(input);

if(contains(input,validLetters)) {

return true;

}else{

return false;

}

}

Um den aktuellen Spielstand zu zeichnen, haben wir public void zeichneSpiel() in unseren Code integriert.

System.out.println("\n");

switch (fehlerAnzahl) {

case 0: break;

case 1: System.out.println(" \_\_\_\_\_");

System.out.println("/ \\");

break;

case 2: System.out.println(" |");

System.out.println(" |");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 3: System.out.println(" |´");

System.out.println(" |");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 4: System.out.println(" \_\_\_\_\_");

System.out.println(" |´");

System.out.println(" |");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 5: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ `");

System.out.println(" |");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 6: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" |");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 7: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" | O");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 8: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" | /O");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 9: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" | /O\\");

System.out.println(" |");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 10: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" | /O\\");

System.out.println(" | /");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

case 11: System.out.println(" \_\_\_\_\_");

System.out.println(" |´ ò");

System.out.println(" | /O\\");

System.out.println(" | / \\");

System.out.println(" \_\_|\_\_");

System.out.println("/ \\");

break;

}

System.out.println();

for(int j =0; j<wortArr.length; j++) {

System.out.print(wortArr[j] + " ");

}System.out.println();

System.out.println();

System.out.print("Schonmal geratene Buschstaben: ");

System.out.println(schonVersucht);

System.out.println("\n");

}

public boolean getStatus(){

if(fehlerAnzahl >= 11) { // wurden mehr als 11 Fehler gemacht gilt das Spiel verloren

verloren = true;

}

gewonnen = true;

for (int i = 0;i < gesWort.length();i++) {

/\*Wenn alle Buchstaben erraten wurden ist das spiel gewonnen.

\* Dies wird über eine for-Schleife geprüft die alle erratenen Buchstaben mit dem gesuchten wort abgleicht

\* wenn ein Buchstabe fehlt wird die gewonnen Variable wieder auf falsch gesetzt\*/

if(wortArr[i] != gesWort.charAt(i)) {

gewonnen = false;

break;

}

}

/\*sofern das Spiel noch nicht gewonnen oder verloren wurde wird der Status true ausgegeben

\* das spiel läuft also noch weiter\*/

if (gewonnen == false && verloren == false) {

return true;

}else if(gewonnen == true) { //wenn das Spiel gewonnen wurde wird eine Siegernachricht ausgegeben und der Spielstatus auf falsch gesetzt

System.out.println("Hurra du hast das Spiel gewonnen!!!!");

return false;

}else{

/\*wenn das Spiel verloren wurde wird eine Trostnachricht ausgegeben und der Spielstatus auf falsch gesetzt

\*trifft keine der vorigen abfragen zu gilt das spiel als verloren dies ist notwendig da java bedingte returns nicht anerkennt

\*und immer ein default das in jedem fall ausgelöst werden kann braucht\*/

System.out.println("Leider hast du dieses Spiel verloren. Vielleicht klappt es ja beim nächsten Mal.");

System.out.println("Das zu erratende Wort war übrigens " + gesWort + ".");

return false;

}

}

public static boolean prüfeEingabe(String Wort){ //Überprüft ein Wort ob es mindestens 2 Buchstaben hat und alle buchstaben akzeptiert sind. Diese Überprüfung muss extern durchgeführt werden.

boolean wert = true;

for(int i=0; i<Wort.length(); i++) {

if(!validateInput(Wort.charAt(i))){

wert = false;

break;

}

}

if(Wort.length() < 2) {

wert = false;

}

return wert;

}

}

1. **Zusammenarbeit**

Für den Start unseres Projektes haben wir uns zusammen Online getroffen. Hier haben wir Java/Eclipse mit GitHub verknüpft und jeder von uns hat einen eigenen Branch angelegt. Des Weiteren hat jeder erstmal über das Wochenende für sich Ideen gesammelt. Unser nächster Gruppentermin hat in Präsenz stattgefunden und wir haben uns alle zusammen und gemeinsam ausgetauscht. Hier haben wir auch festgelegt, wer an welcher Gruppenaufgabe mehr beteiligt sein möchte und wie unser weiteres Vorgehen erfolgt. Insgesamt hat unsere Gruppenarbeit gut funktioniert und wir kamen gut zurecht.

