# Dokumentation zu Aufgabe 6: „Mastermind Teil 1“

Um diese Aufgabe zu lösen schrieben wir drei Dateien mit insgesamt drei Klassen.

1. Datei und Klasse: „codemaker“
2. Datei und Klasse: „codebreaker“
3. Datei und Klasse: „daten“

Die Klasse „codemaker“ erstellt einen Random-Code.

Die Klasse „codebreaker“ enthält die Logik zum vergleichen des Codes, der vom „codemaker“ kommt mit dem Code, der geraten wird.

Die Klasse „daten“ lässt den Nutzer sie Daten eingeben, ist also die Schnittstelle zwischen Programm und Mensch.

Als letztes haben wir noch die Exceptiones und den Ruby-Test geschrieben.

**Schwierig** ist es gewesen, sein vergleichen zu optimieren, sodass ich Arrayinhalte nicht mehrfach miteinander vergleiche.

# Dokumentation zu Aufgabe 7: „Mastermind Teil 2“

Diese Aufgabe setzt auf die Klassen von Aufgabe 6 auf.

Dazu kamen noch:

1. Datei und Klasse: „knuth“
2. Datei: „mastermind\_menu“

In Knuth wird der Knuth’sche Minimax Algorithmus zur Lösung des Mastermind Codes implementiert:

1. Alle Möglichkeiten bilden
   1. Erster Rateversuch wird gebildet
   2. Alle Rateversuche, die nicht mind. die selbe Anzahl an white und black pins haben
2. Berechnung für alle Codemöglichkeiten die Anzahl black und white hits berechnen
3. Zurück zu Schritt 2. bis Lösung gefunden

Methoden die wir größtenteils von Laurence Jennings übernommen haben:

* verify\_input
* s\_list
* pin\_possibilities
* remove\_wrong\_guesses
* score\_guesses
* fast\_score\_guesses

Als letztes haben wir noch die Exceptiones und den Ruby-Test geschrieben.

**Schwierig** ist es gewesen, den Algorithmus zu implementieren und ihn dann konfigurierbar zu gestalten.