Aufgabe A07: Mastermind

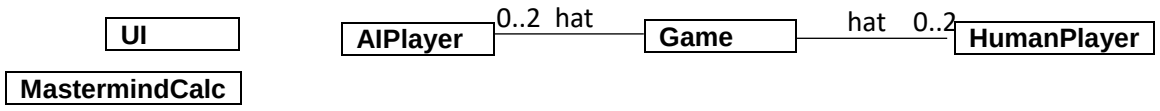
BAI1/W2018

ChillyCrabs / NeamTame

Autoren: Lennart Draeger, Robert Gnehr

Hamburg, 26.11.2018

Aufgabe A06



- **Game**
  Die Game enthält die gesamte Spiellogik. In ihr sind die Anzahl der Versuche, die Codelänge, die gültigen Werte, die ein Code annehmen kann und der Code selbst gespeichert. Außerdem gibt es jeweils ein Spielerobjekt für den Codemaker und -breaker.
  Die Methode game_loop steuert den groben Spielablauf: Es wird abgefragt, ob Codemaker und -breaker jeweils von einem Menschen oder dem Computer gespielt werden, dann wird ein neuer Code erstellt, anschließend der Code geraten und dann nach einer neuen Spielrunde gefragt.
  In der Methode guessing_loop läuft das Erraten des Codes in einer Schleife ab. Dazu wird vom Codebreaker ein neuer Versuch abgefragt, dieser dann per Mastermind.calc.match_hits evaluiert und dem Codebreaker mitgeteilt. Falls der erraten wurde (4 Balck hits) oder die Rateversuche aufgebraucht wurden, endet die Schleife mit dem entsprechenden Hinweis.
  Alle Eingaben von und Ausgaben durch die Konsole laufen dabei über die Utility-Klasse UI ab.

- **UI**
  Enthält sämtliche Ein- und Ausgabemethoden des Spiels. Diese sind als Klassenmethoden implementiert, so dass es kein gesondertes UI Objekt gibt.

- **HumanPlayer**
  Enthält die Steuerungslogik für menschliche Spieler. Speichert als Attribute die Gültigen Codewerte, die Codelänge, alle bisherigen Versuche samt Hit-Feedback und einen Array, in dem alle möglichen Codevarianten enthalten sind.
  In der generate_code Methode wird in einer Schleife über die UI nach einem neuen Code gefragt (und dabei die Rahmenbedingungen angegeben) und so lange mit den gültigen Werten und der geforderten Länge abgeglichen, bis ein gültiger Code eingegeben wurde. Dieser wird anschließend zurückgegeben.
  In der guess_code Methode wird in einer Schleife über die UI nach einem neuen Rateversuch gefragt. Dieser Versuch wird auf Gültigkeit (Länge, Werte) geprüft. Erst gültige Versuche werden zurückgegeben. Die Eingabeschleife gibt bei Eingabe von ‘h‘ einen kurzen Hilfstext aus. Bei Eingabe von ’c‘ wird ein sinnvoller nächster Rateversuch generiert: Alle bisherigen Versuche werden jeweils mit allen Codevarianten aus dem Array aller Möglichkeiten abgeglichen und dabei auf Hits geprüft: Wenn eine Codevariante bei einem Versuch dessen Hits erzielt, bleibt sie im Array, sonst wird sie entfernt. Über die UI wird schließlich die erste der verbleibenden Codemöglichkeiten ausgegeben.
  Die Methode pass_feedback fügt einen Codeversuch und dessen zugehörigen Hits dem entsprechenden @previous_guess Attribut hinzu.

- **AIPlayer**
  Enthält die gleichen Attribute und öffentlichen Methoden wie HumanPlayer, errechnet dabei den Code ohne Konsolenabfrage selber (Gibt aber entsprechende Rückmeldung über die UI). Die Erechnung eines Rateversuch erfolgt genau wie die Tipberechnung bei HumanPlayer, findet aber automatisch in jedem Aufruf von guess_code statt.

- **MastermindCalc**
  Enthält als Utility Klasse ausschließlich eine Klassenmethode zur Berechnung von schwarzen und weißen Hits: Falls zwei übergebene Arrays am jeweils gleichen Index übereinstimmen, zählt das als Black Hit. Sind danach noch Werte im guess Array, die auch im Code Array enthalten sind (und nicht als black hit gezählt wurden), zählen diese als Whit Hits.