Termin 5: Wer wird Millionär – Kollaborative Code-Entwicklung

Aufgabestellung (1)

- **Programmablauf**
  - **Die Fragen werden von der Fragen-DB eingelesen**
    - **git/dhbw-ss-2020/Termin_4-Million/Fragen-DB**
  - **Format der einzelnen Frage**

```
Fragetext
<Leerzeile>
+- Antwort_1
+- Antwort_2
+- Antwort_3
+- Antwort_4
```

Aufgabestellung (2)

- **Programmablauf**
  - **Spieler gibt seinen Namen ein**
  - **Spielerkonto wird ursprünglich auf 0 Eur. gesetzt**
  - **Auf jeder der 7 Spielrunden bekommt der Spiele eine Frage (und 4 Antworten dazu), die zufällig aus der Fragen-DB (aber nur 1 Mal pro Spiel) ausgewählt wird.**
    - **Achtung: Die Antworten müssen auch in unterschiedliche Reihenfolge ausgegeben werden, als in der DB eingetragen ist (z.B. 1, 4, 3, 2)**
  - **Das Spiel läuft über 7 Runden, in jeder dem Spieler 1 Frage mit 4 Antwortmöglichkeit gestellt wird.**

Aufgabestellung (2)

- **Programmablauf**
  - **Der Spieler wählt entweder eine Antwort (A, B, C, D) oder nimmt eine 50-50 Hilfe (es werden eine richtige und eine falsche Antwort widerholt angezeigt).**
    - **Achtung: Die Hilfe kann nur 1 Mal im Spiel genutzt werden**
  - **In jeder Runde kann der Spieler 10, 100, 1K, 10K, 100K, 500K, 1M Euro bei richtigen Antworten gewinnen**

Ablauf der Entwicklung

- Studierenden werden in Gruppen eingeteilt
- Jede Gruppe erstellt ein Verzeichnis im Git, in welchem das Programm gehalten wird
  - git/dhbw-ss-2020/Termin_4-Million/Gruppe_X
- In jeder Gruppe wird ein Teamleader ausgewählt
  - Die Rolle kann auch gerne abwechselnd übernommen werden
- Teamleader koordiniert die Erstellung des Programm-Gerüsts und lädt es hoch
- Das Team entscheidet, welcher Team-Mitglied um welche Funktion kümmert
- Die Teammitglieder fügen ihre entwickelten Code-Teile ins Git hinzu

Beispiele der einzelnen Funktionen

- Einlesen aller Fragen aus der Fragen-DB ins Fragen-Catalogue
- Zufällige Auswahl einer Frage aus dem Catalogue und Ausgabe auf Bildschirm
- Bearbeitung der Nutzereingaben und Kontoverwaltung
- …

Beispiele des Programm-Gerüsts

```
struct Fragen_Catalogue_Eintrag {
    char* Frage;
    char* Antworten[4];
    int nr_correct;
}

Fragen_Catalogue_Eintrag [] Catalogue; // Array von allen Catalogue-Einträge

void read_fragen_db (Fragen_Catalogue_Eintrag* Catalogue, int* nr_entries)

int frage_auswahl (Fragen_Catalogue_Eintrag* Catalogue, int nr_entries)
    // wählt zufälligerweise eine Frage (als Index) aus dem Catalogue

void frage_ausgabe (Fragen_Catalogue_Eintrag* Catalogue, int index)

void frage_ausgabe_50-50 (Fragen_Catalogue_Eintrag* Eintrag, int index)
```