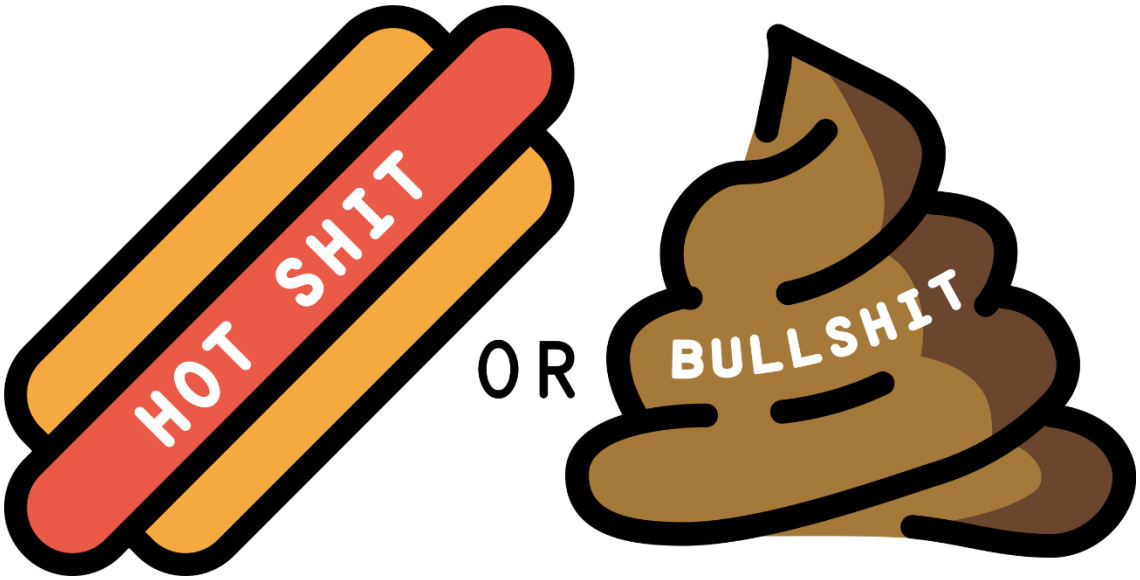
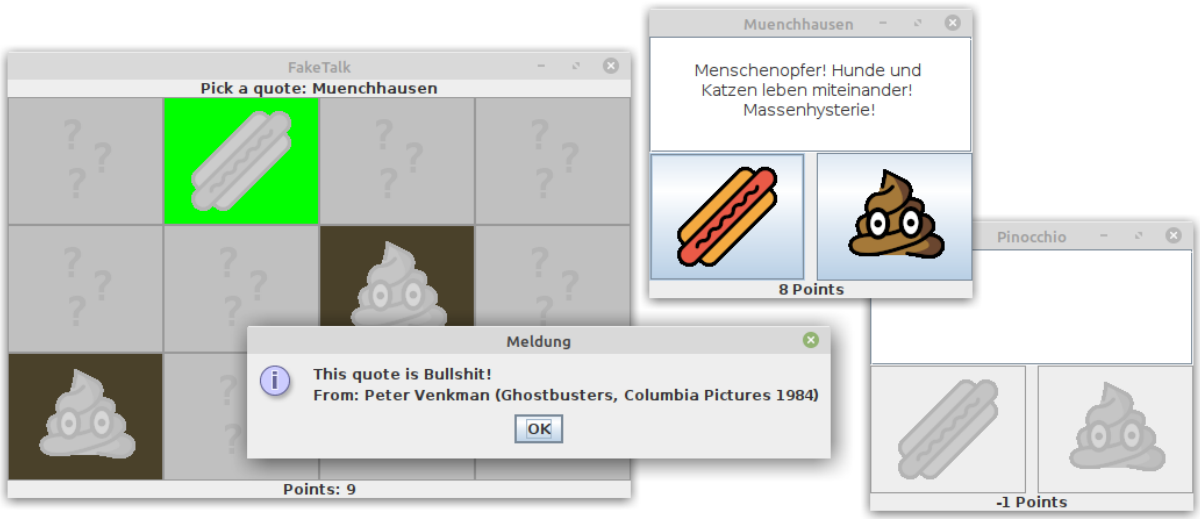
FÄIK TÄLK





Hinweis zur Bewertung:

- 1/4 der Punkte (25%) wird nach Funktionstests Ihrer Lösung vergeben.
- 3/4 der Punkte (75%) werden entsprechend des in den Teilaufgaben angegebenen Schlüssels auf Basis des Quellcodes vergeben.

DHBW Karlsruhe, Vorlesung Programmieren I+II,
Probe-sProgrammentwurf
Bearbeitungszeit: 120 Minuten

Aufgabe

Die letzten Jahre fällt es immer schwerer, Aussagen, die auf Social-Media-Plattformen die Runde machen oder im Fernsehen getätigt werden, richtig einzuordnen. Zitate können dabei entweder zum Zeitpunkt der Äußerung schon problematisch sein, sich im Nachhinein als falsch herausstellen oder schlicht rein fiktional sein, da sie bspw. aus Film-Dialogen stammen. Zur korrekten Bewertung ist eine gute Medienkompetenz gefragt!

Sie sollen in Form der grafischen Java-Anwendung **`FakeTalk`** einen spielerischen Ansatz realisieren, der die Einordnung von Zitaten ermöglicht. Die Aufmachung soll sich an ein eher junges Publikum richten und ist in Sachen Design und Sprache bewusst plakativ und überspitzt (quasi doppelter Test der Medienkompetenz 😉).

Teilaufgabe a) **[6%]**

Um verschiedene Einordneungen von Zitaten im Programm zu ermöglichen soll zunächst ein *komplexer Aufzählungstyp* **`QuoteType`** realisiert werden.

Der Aufzählungstyp soll hierbei neben der Konstante ein menschenlesbares Label (`label`), ein Bild für die spätere Darstellung (`icon`) sowie eine Farbe (`color`) realisieren. Die Icons sollen vom Typ `javax.swing.ImageIcon` sein und sind in der mitgelieferten Klasse **`FakeTalkIcons`** (s. USB-Stick) bereitgestellt.

Wir unterscheiden inhaltlich nur zwischen non-fiktionalen Zitaten, die bis heute Bestand haben („der heiße Scheiß“), und dem Rest („Blödsinn“). Hierfür sind folgende drei `QuoteType`-Werte sind zu realisieren:

QuoteType	Label	Icon	Farbe
`UNKNOWN`	`Unknown`	`FakeTalkIcons.ICON_UNKNOWN`	`java.awt.Color.LIGHT_GRAY`
`HOT_SHIT`	`Hot shit`	`FakeTalkIcons.ICON_HOT_SHIT`	`java.awt.Color.GREEN`
`BULLSHIT`	`Bullshit`	`FakeTalkIcons.ICON_BULLSHIT`	`new java.awt.Color(74,65,42)`

Teilaufgabe b) **[7%]**

Zur Repräsentation eines Zitats soll die Klasse **`Quote`** umgesetzt werden. Ein Zitat besteht aus dem Text (`text`), der Name der Person, welche dies gesagt bzw. geschrieben hat (`person`), deren Funktion (`role`), der Quelle des Zitats (`source`) sowie dem Typ des Zitats (`type`, vgl. *Teilaufgabe a)*).

Realisieren Sie einen Konstruktor, welcher die Attribute in exakt dieser Reihenfolge entgegennimmt und in den zugehörigen Instanzvariablen speichert.

Ebenso ist eine Methode **`getCitation`** zu realisieren, welche als Zeichenkette die Informationen zur Person, ihrer Funktion und der Quelle des Zitats liefert (vgl. Dialog im Screenshot auf dem Deckblatt).

Hinweis: der Konstruktor wird in der bereitgestellten Methode `FakeTalk.parseQuote` verwendet.

Teilaufgabe c) **[5%]**

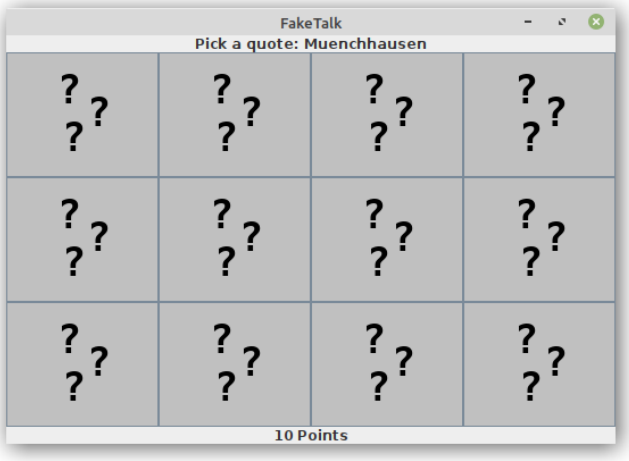
Um neben unserem Beispiel-Client später weitere Clients problemlos ergänzen zu können soll eine Java-*Schnittstelle* **`FakeTalkClient`** realisiert werden, welche folgende Methoden für spätere Implementierungen definiert:

- `String getPlayerName()`: soll den Namen des Spielers zurückgeben
- `void setQuote(Quote q)`: setzt dem Client das Zitat und aktualisiert ggf. die Anzeige
- `void addPoints(int points)`: fügt dem Client Punkte hinzu und aktualisiert ggf. die Anzeige
- `int getPoints()`: liefert den aktuellen Punktstand

Teilaufgabe d) **[30%]**

Wie auf dem Deckblatt zu sehen sollen Spieler ein Zitat auswählen können. Schreiben Sie hierfür eine Klasse **`QuoteSelectionTerm`**, die dies in Form einer grafischen Oberfläche ermöglicht. Am oberen Rand ist eine Anzeige vorzusehen, die den aktuellen Spieler zur Auswahl auffordert, darunter eine Matrix-artige Struktur für die Auswahl und am unterem Rand eine Anzeige für die Punkte, die für eine korrekte Antwort vergeben werden (zunächst: `Points: 10`). Der Titel soll entsprechend dem auf der nächsten Seite folgenden Screenshot gesetzt werden.

Realisieren Sie als Knopf für die Auswahl der Fragen sollen Sie eine eigene Klasse **`QuoteButton`**, welche von `JButton` erbt. Ein `QuoteButton` hat ein zusätzliches Attribut, welches den aktuellen Typ (vgl. *Teilaufgabe a)*), für den der Button steht, repräsentiert.



Entsprechend dem aktuellen Typ sind immer die Hintergrundfarbe (vgl. Hinweise) und das Icon des `QuoteButtons` auf die Werte des aktuellen Typs zu setzen! Initial sind alle `QuoteButtons` vom Typ `QuoteType.UNKNOWN`.

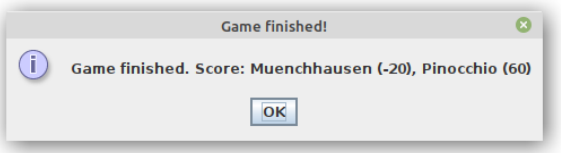
Ebenso soll es eine Methode **`boolean`** `isUnknown()` geben, die genau dann **`true`** zurückliefert wenn der Typ des Buttons `QuoteType.UNKNOWN` ist, in jedem anderen Fall **`false`**.

Der Konstruktor von `QuoteSelectionTerm` nimmt die Liste aller Zitate sowie die Anzahl der Zeilen und Spalten für die Auswahl entgegen (in dieser Reihenfolge, vgl. Verwendung in Klasse `FakeTalk` vom USB-Stick).

Sollte die Größe der übergebenen Zitatliste nicht ausreichen um genügend Zitate für die gegebenen Zeilen und Spalten auswählen zu können, ist eine *selbst zu schreibende* **`FakeNewsException`** zu werfen, der die Nachricht „`Provided quote catalog does not contain enough (hot|bull)shit!`“ übergeben wird.

Sind genügend Zitate vorhanden wählen Sie die korrekte Anzahl *zufällig* aus dem Katalog aus und nutzen Sie diese für das aktuelle Spiel. Duplikate sind hierbei zu vermeiden. Ebenso sind folgende Methoden zu realisieren:

- **`void`** `register(FakeTalkClient client)`: Ermöglicht das Registrieren eines Clients. Diese müssen in einer geeigneten Datenstruktur abgelegt werden. Sollte das Spiel bereits gestartet worden sein (siehe darunter) ist eine `FakeNewsException` mit einer entsprechenden Nachricht zu werfen.
- **`void`** `start()`: Starten des Spiels. Sollten sich beim Aufruf der Methode weniger als 2 Clients in der o.g. Datenstruktur befinden, ist eine `FakeNewsException` mit entsprechender Fehlermeldung zu werfen. Wenn alles passt soll das Spiel gestartet werden und der erste Client zur Auswahl eines Zitats aufgefordert werden (vgl. Screenshot oben). Wurde das Spiel bereits gestartet soll nichts passieren.
- **`void`** `answerSelected(FakeTalkClient client, Quote q, QuoteType selectedType)`: Teilt eine selektierte Antwort mit. Dabei sind folgende Schritte durchzuführen
  - Es ist ein Dialog mit dem eigentlichen Typ des Zitats sowie dessen Quellenangabe (`getCitation`-Methode) anzuzeigen (vgl. Screenshots unten links).
  - Ist der ausgewählte Zitat-Typ `selectedType` richtig, sind die entsprechenden Punkte (aktuell immer 10, siehe Hinweise) dem Spieler hinzuzufügen, ansonsten sind sie abzuziehen. Das Label für die Punkte ist ebenfalls entsprechend zu aktualisieren.
  - Der `QuoteButton` für das übergebene Zitat ist auf den korrekten Zitat-Typ zu setzen
  - Alle `QuoteButtons`, die noch im Zustand `UNKNOWN` sind, sind wieder zu aktivieren
    - Ist kein `QuoteButton` mehr im Zustand `UNKNOWN` (d.h. alle Zitate wurden bereits gespielt) ist ein Dialog mit den Punkten aller Spieler anzuzeigen, der über das Ende des Spiels informiert (vgl. Screenshot unten rechts).
    - Ansonsten ist der nächste Spieler an der Reihe und soll zur Auswahl eines Zitats aufgefordert werden (Label am oberen Rand).



Wird ein QuoteButton ausgewählt, ist dies dem aktuellen Client mittels der `setQuote`-Methode mitzuteilen und alle QuoteButtons sind zu deaktivieren.

Hinweis 1: UI-Elemente können mit `setEnabled(false)` deaktiviert bzw. mit `setEnabled(true)` aktiviert.

Hinweis 2: Die Anzeige und Vergabe der Punkte wird in Teilaufgabe i) erweitert.

Hinweis 3: `ImageIcon`-Instanzen können einfach mittels der Methode `setIcon` auf einem Button angezeigt werden.

Hinweis 4: Die Hintergrundfarbe eines Buttons kann mit der Methode `setBackground(color)` gesetzt werden.

Teilaufgabe e) **[12%]**

Als grafische Oberfläche zur Einordnung von Zitaten durch Spieler ist eine Klasse `QuoteTerm` umzusetzen. Diese ist natürlich eine *Ausprägung der Schnittstelle* `FakeTalkClient` und realisiert die hierdurch vorgegebene Funktionalität wie in *Teilaufgabe c)* beschrieben.

Der Konstruktor nimmt hierbei den Namen der Person entgegen, die in diesem `QuoteTerm` spielt, sowie die Referenz auf das `QuoteSelectionTerm`, um diesem später die Kategorie-Selektion mitzuteilen (vgl. Verwendung in bereitgestellter Klasse `FakeTalk`, s. USB-Stick).

Der Titel des Fensters ist auf den Namen der spielenden Person zu setzen. Im oberen Bereich sehen Sie die Anzeige für das später zu bewertende Zitat vor. Nutzen Sie hierfür die mitgelieferte Komponente `QuoteDisplay` (s. USB-Stick) welche sich um Mindestgröße, Schriftgröße, Zeilenumbruch, Ausrichtung und ggf. Scroll-Balken kümmert. Den Zitattext können Sie später mit Hilfe der Methode `setText(String)` setzen.

Darunter sind zwei Buttons vorzusehen, welche die Auswahl von „Heißem Scheiß“ oder „Blödsinn“ für ein Zitat erlauben und die initial *deaktiviert sind*. Nutzen Sie für die Anzeige die jeweiligen Icons (vgl. Screenshot)! Darunter ist ein Label für die Anzeige der aktuell gesammelten Punkte vorzusehen.

Beim Aufruf der durch die Schnittstelle definierte `setQuote`-Methode ist das Zitat in der Anzeige sichtbar zu machen und die Buttons müssen für die Auswahl des Zitat-Typs aktiviert werden.

Die Buttons müssen bei Auswahl eines Typs dem `QuoteSelectionTerm` mittels der Methode `answerSelected` mitteilen, welcher Typ für das Zitat ausgewählt wurde. (Es müssen der aktuelle Client, das aktuelle Zitat und der auswählte Typ je nach gewähltem Button übergeben werden.)

Teilaufgabe f) **[5%]**

Ändern Sie die Implementierung der `loadQuotes`-Methode in der bereitgestellten Klasse `FakeTalk` (s. USB-Stick) so, dass die auf dem USB-Stick bereitgestellte Datei „`corona.csv`“ (CSV = „Catalog of shit verses“ 😉) zeilenweise eingelesen wird. Fehler müssen zwar abgefangen aber nicht weiter behandelt werden.

Dabei ist jede Zeile der Datei in ein `Quote`-Objekt umzuwandeln und anstatt der Beispieldaten der Liste der Zitate hinzuzufügen. Nutzen Sie für das Konvertieren die bereitgestellte Methode `parseQuote`, welche bereits für die Umwandlung der Beispieldaten genutzt wird.

Teilaufgabe g) **[5%]**

Erweitern Sie die `answerSelected`-Methode von `QuoteSelectionTerm` (siehe *Teilaufgabe d)*) nun so, dass zum Ende des Spiels die zuvor lediglich im Dialog angezeigte Nachricht zusätzlich in eine Textdatei mit dem Namen „`fake-score.txt`“ gespeichert wird. Sollte die Textdatei bereits existieren, ist die Nachricht als neue Zeile an das Ende anzuhängen. Fehler müssen zwar abgefangen aber nicht weiter behandelt werden.

Teilaufgabe h) **[5%]**

Erweitern Sie die Klasse `QuoteSelectionTerm` (vgl. *Teilaufgabe d)*) nun so, dass alle *zwei Sekunden* die Punktzahl für die Beantwortung des Zitattyps um einen Punkt reduziert wird. Aktualisieren Sie auch die Anzeige entsprechend!

Diese Aktion soll *nebenläufig* durchgeführt werden. Die nebenläufige Ausführung endet, nachdem entweder eine Antwort für das aktuelle Zitat gegeben wurde oder die zu vergebende Punktzahl nur noch einen (1) Punkt beträgt (und dann auch nicht mehr kleiner wird).

Allgemeine Hinweise

Starten
Starten Sie die Anwendung mit der gegebenen Klasse `FakeTalk` (siehe USB-Stick).

Schließen eines Fensters
Beim Schließen eines Fensters soll die komplette Anwendung beendet werden.

Sichtbarkeit von Instanz-Attributen
Sämtliche Instanz-Attribute sind als privat zu definieren und von außerhalb der Klasse ggf. mittels Getter- und/oder Setter-Methoden zu verwenden.