

# CHATTER GPT

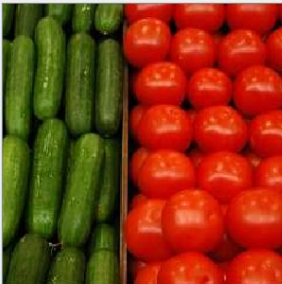



## VALIDATOR

#Söderisst  
Edition



Dateien: <https://www.iai.kit.edu/javavl/cgpt.zip>

Validation Term (5 Seconds)



New Images

Classifications Term

Images to verify: 16

TITLE	SCORE	NUMBER
Balkan-Fleisch	10	1
Bier	10	1
Bratwurst	10	1
Brot	0	0
Döner-Kebap	0	2
Eis	0	0
Erdbeerlimonade	0	0
Fischsemmel	10	1
Gemüse	0	0
Leberkäswack	20	2
Lebkuchen	0	0
Quarktasche	10	1
Salami	0	0
Salamibrot	0	0
Spargel	-10	1
Wassermelone	-10	1

### 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.

Deko-Bild erzeugt mit Adobe Firefly 3, Prompt: „Cartoon-artiges Bild eines Manns im Anzug, der einen Döner isst“. Stil: Schichtpapier

**DHBW Karlsruhe, Vorlesung Programmieren I+II,  
Probe-Programmmentwurf Sommersemester 2025  
Bearbeitungszeit: 120 Minuten**

## **Aufgabe**

Im politischen Geschäft gilt manchem eine gewisse Ernsthaftigkeit als Tugend. Aber auch in der Politik gibt es extrovertierte Menschen, deren Mitteilungsbedürfnis diese Ernsthaftigkeit überlagert. Im Englischen gibt es dafür das Verb "to chatter" (= to talk for a long time about things that are not important). Im süddeutschen Sprachraum wird das gelegentlich als "tratschen" bezeichnet.

In den sozialen Medien wird solcher Tratsch häufig durch Fotos illustriert. Eine neue Künstliche Intelligenz (KI) soll künftig solchen Tratsch automatisiert erkennen und löschen. Der Name der KI: "**chatterGPT**". GPT steht dabei für "Greißlicher proletischer Tratsch".

Um das KI-Modell zu validieren, sollen Sie eine grafische Java-Anwendung entwickeln, die CAPTCHA-artig Daten verifiziert, die von chatterGPT klassifiziert wurden. Als Eingabedaten dienen Social-Media-Posts zu einem echten Hashtag. Die Wahl fiel dabei auf "#söderisst"...

### **Teilaufgabe a) [15%]**

Zum Validieren unserer KI werden Bilder benötigt, die entsprechend ausgezeichnet sind. Schreiben Sie hierfür eine Klasse **ImageDescription**, welche ein Label für ein Bild (`label`), ein Hauptbild (`mainImage`) und ein Referenz-Bild (`referenceImage`, beide vom Typ `javax.swing.ImageIcon`). Ein Konstruktor soll die Argumente in exakt dieser Reihenfolge entgegennehmen und in Instanz-Variablen speichern (siehe Verwendung in bereitgestellter Klasse **ChatterGPT**).

Für die eigentliche Klassifikation soll eine Klasse **ClassificationResult** entwickelt werden, welche die Bildbeschreibung (`image`) und einen Ergebnis-Typ (`type`) (s.u.) speichern kann. Definieren Sie auch hier einen geeigneten Konstruktor, der beide Werte initialisiert.

Der Typ des Ergebnisses soll in Form eines *komplexen Aufzählungstyps* **ClassificationResultType** umgesetzt werden. Neben der eigentlichen Konstante besitzt der Typ ein menschenlesbares Anzeige-Label (`label`) und eine Punktzahl, die für die Antwort vergeben wird (`score`).

Folgende Ergebnis-Typen sollen implementiert werden:

<b>ClassificationResultType</b>	<b>Anzeige-Label</b>	<b>Score</b>
<b>CORRECT</b>	Correct	10
<b>INCORRECT</b>	Incorrect	-10
<b>NOT_CLASSIFIED</b>	Not classified	0

### **Teilaufgabe b) [3%]**

Um Visualisierungen der Validierungsergebnisse später anzuzeigen, soll zunächst eine *Schnittstelle* **ClassificationsDisplay** definiert werden. Sie definiert nur eine Methode, die für ein bestimmtes Bild das Setzen einer Punktzahl sowie der Anzahl der Validierungen, die zu der Punktzahl führten, ermöglicht:

```
void setValues(ImageDescription desc, int score, int number)
```

### **Teilaufgabe c) [10%]**

Zum Erfassen aller Klassifikationsergebnisse soll nun die Klasse **ResultStore** entwickelt werden. Ihr Konstruktor erhält eine Anzeige (`display`, siehe Verwendung in bereitgestellter Klasse **ChatterGPT**), welche für die spätere Verwendung in einer Instanz-Variable gespeichert wird. Weiterhin soll die Klasse in einer geeigneten Datenstruktur die Möglichkeit schaffen, eine beliebige Anzahl von Klassifikationsergebnissen zu speichern.

Darüber hinaus soll **ResultStore** eine Methode bereitstellen, die es erlaubt, ein Ergebnis hinzuzufügen:

```
public void addResult(ClassificationResult res)
```

Die Methode soll das übergebene Klassifikationsergebnis in der o.g. Datenstruktur speichern und anschließend folgende Werte bestimmen:

- Die Summe aller Punkte, die sich aus dem Resultat-Typ der gespeicherten Ergebnisse für diese Bildbeschreibung ergeben.
- Wie viele Ergebnisse für diese Bildbeschreibung vorliegen

Die bestimmten Werte sollen an die gespeicherte Anzeige mittels der `setValues`-Methode weitergegeben werden!

Hinweis: die Methode `addResult` wird in Teilaufgabe g) noch erweitert!

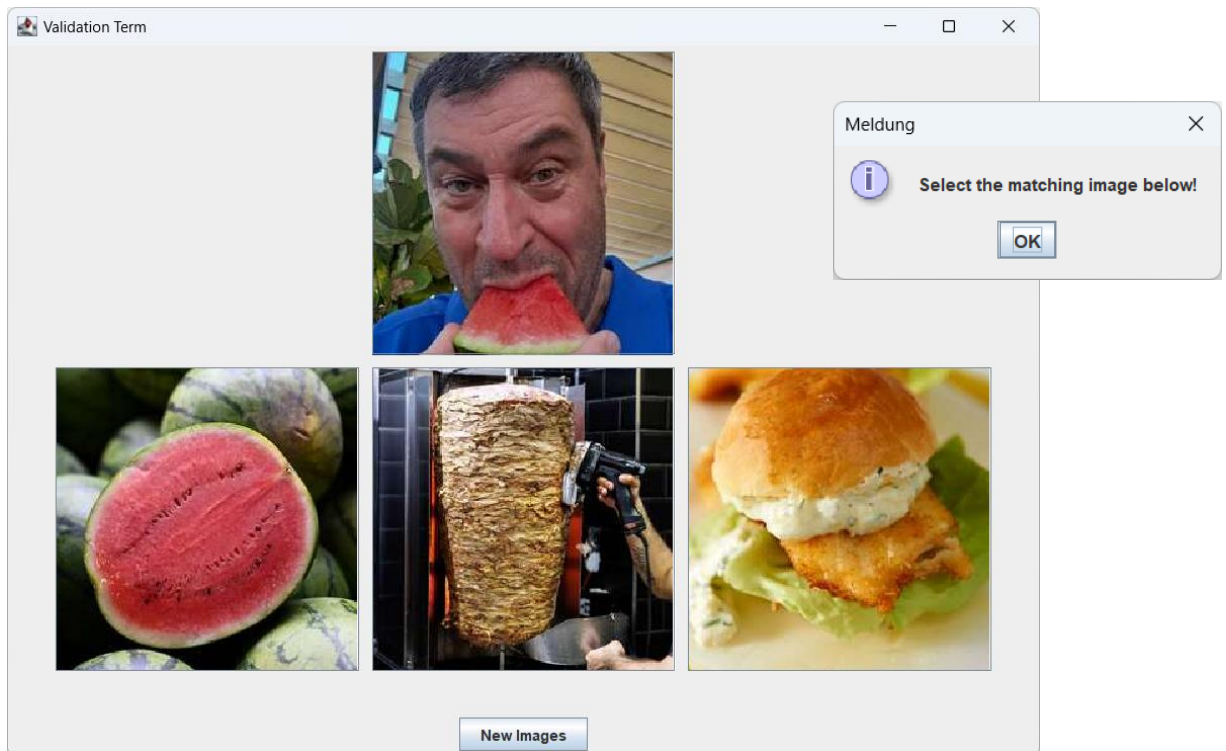
### Teilaufgabe d)

[18%]

Erstellen Sie nun eine Klasse **ValidationTerm**, welche die grafische Oberfläche bereitstellt, um die Klassifikation vorzunehmen, welche zur Validierung der KI-Daten verwendet wird.

Der Konstruktor von **ValidationTerm** bekommt als Parameter die Liste aller Bildbeschreibungen, den Ergebnis-Speicher, sowie eine Anzahl von Bildern, die zur Klassifikation angeboten werden sollen (siehe Verwendung in bereitgestellter Klasse **ChatterGPT**).

Prüfen Sie zunächst, ob die Anzahl Einträge in der Liste der Bildbeschreibungen ausreicht um die übergebene Anzahl an Bildern ohne Doppelungen anzuzeigen. Ist dies nicht der Fall, ist eine *selbst zu schreibende* **ChatterException** zu werfen, der eine entsprechende Fehlermeldung übergeben wird, die auf das Problem hinweist.



Setzen Sie für das Fenster einen passenden Titel und sehen Sie zunächst im oberen Bereich einen (Bild-)Button für das Hauptbild einer Beschreibung vor. Darüber hinaus ist darunter die Anzahl an (Bild-)Buttons, die durch den übergebenen Anzahl-Wert des Konstruktors vorgegeben sind zu realisieren.

Wählen Sie aus der übergebenen Bildbeschreibungsliste die übergebene Anzahl zufällig aus, Duplikate sind zu vermeiden. Nutzen Sie die erste der ausgesuchten Bilderbeschreibungen und zeigen dessen Hauptbild im oberen (Bild-)Button an. Auf die untere (Bild-)Buttonreihe sind alle Referenzbilder zufällig zu verteilen.

Der obere (Bild-)Button soll beim Anklicken einen Dialog anzeigen, der darauf hinweist, dass für die Klassifikation einer der unteren (Bild-)Buttons ausgewählt werden soll.

Wird nun einer der unteren (Bild-)Buttons ausgewählt, ist zunächst zu bestimmen, ob das richtige Bild ausgewählt wurde (d.h., ob das Referenzbild zum Hauptbild gehört). Ist dies der Fall, so erzeugen Sie ein Klassifizierungsergebnis mit dem Typ **CORRECT**, andernfalls mit dem Typ **INCORRECT** und übergeben Sie es der `addResult`-Methode der gespeicherten Ergebnisspeicher-Instanz.

Im unteren Bereich soll durch einen Button ermöglicht werden, die Auswahl eines Bildes neue Bilder für die Klassifikation anzuzeigen. Fügen Sie dann für das aktuelle Bild ein Klassifikationsergebnis mit dem Typ **NOT\_CLASSIFIED** zum Ergebnisspeicher hinzu und wählen Sie nach o.g. Kriterien drei neue Bildbeschreibungen aus und zeigen diese an.

Hinweis: die **ImageIcon**-Instanzen aus **ImageDescription** können mit der **JButton**-Methode `setIcon` angezeigt werden.

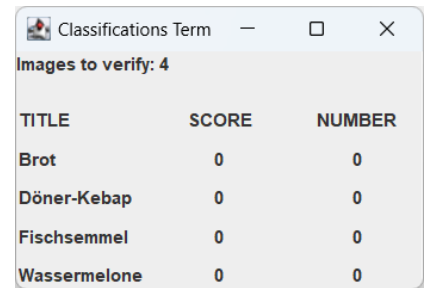
### Teilaufgabe e)

[13%]

In Teilaufgabe b) wurde ja bereits der Grundstein für die Anzeige einer Statistik gelegt. Schreiben Sie nun eine Klasse **ClassificationsTerm** mit einer grafischen Oberfläche, welche eine Ausprägung der Schnittstelle **ClassificationsDisplay** ist.

Der Konstruktor bekommt die Liste aller Bildbeschreibungen übergeben (siehe Verwendung in bereitgestellter Klasse **ChatterGPT**).

Setzen Sie den Titel wie im Screenshot zu sehen und zeigen Sie im oberen Bereich die Anzahl der Bildbeschreibungen der im Konstruktor übergebenen Liste an. Darunter ist eine tabellenartige Anzeige vorzusehen, welche zu jeder Bildbeschreibung den Titel, den aktuellen Punktestand sowie die Anzahl der Klassifikationen anzeigt. Initial steht für Punkte und Anzahl überall 0. Überschreiben Sie die jeweiligen Spalten mit Überschriften analog zum Screenshot.



TITLE	SCORE	NUMBER
Brot	0	0
Döner-Kebap	0	0
Fischsemmel	0	0
Wassermelone	0	0

Wird nun die durch die Schnittstelle vorgegebene Methode **setValues** aufgerufen, so ist die Anzeige für die Zeile der betreffenden Bildbeschreibung entsprechend mit den neuen Werten für Punkte und Anzahl zu aktualisieren!

### Teilaufgabe f)

[6%]

Ersetzen Sie nun die Dummy-Daten, welche vier Bildbeschreibungen in der Methode **loadImageDescriptions** der Klasse **ChatterGPT** bereitstellen. Lesen Sie hierfür die bereitgestellte Datei „**images.txt**“ zeilenweise ein!

Jede Zeile ist dabei analog zu den Dummy-Daten mit der Methode **parseLine** in ein **ImageDescription**-Objekt umzuwandeln und in die Datenstruktur die zurückgegeben wird einzufügen. Fehler beim Lesen der Datei sind abzufangen, müssen aber nicht weiter behandelt werden.

### Teilaufgabe g)

[5%]

Für eine detaillierte Analyse über die angezeigte Statistik hinaus sollen sämtliche Auswahlprozesse nun noch genau protokolliert werden. Ändern Sie die **addResult**-Methode der Klasse **ResultStore** derartig, dass bei jedem Aufruf eine Zeile in die Log-Datei „**classifications.txt**“ geschrieben wird, die den Titel des Bilds und den Typ des Klassifikationsergebnisses enthält, bspw.:

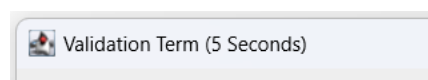
Fischsemmel;Correct

Sollte die Datei bereits existieren, sind weitere Inhalte an das Ende der Datei anzuhängen. Fehler beim Schreiben der Datei sind abzufangen, müssen aber nicht weiter behandelt werden.

### Teilaufgabe h)

[5%]

Abschließend soll nun der Zeitraum für die Zuordnung eines Bildes auf 10 Sekunden eingeschränkt werden. Hierfür soll im Titel des Fensters der Klasse **ValidationTerm** ein Countdown angezeigt werden, der bei 10 Sekunden startet und herunterzählt.



Starten Sie einen nebenläufigen Programmteil, welcher den Countdown sekundlich aktualisiert. Ist der Zähler bei 0 angekommen, soll automatisch das nächste Bild angezeigt werden (selbe Aktion wie „New Images“-Button, inkl. Log-Eintrag mit Typ **NO\_CLASSIFICATION**).

Die Nebenläufigkeit endet, wenn der Countdown abgelaufen ist oder manuell ein Bild ausgewählt wird.

Hinweis: Zur Vereinfachung dürfen Sie quasi-gleichzeitiges manuelles Auswählen und Countdown-Ende vernachlässigen

---

## Allgemeine Hinweise

### Starten

Starten Sie die Anwendung mit der gegebenen Klasse **ChatterGPT** (siehe USB-Stick/Download).

### 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.