Projektblatt 10 (23 P)

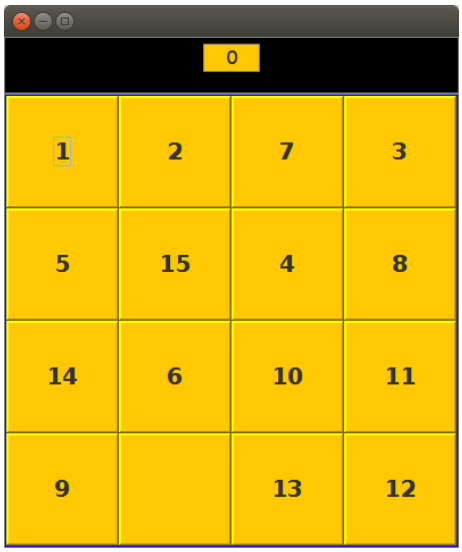
Abgabe: Donnerstag 21. November 2024, 12h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p10.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben befinden. Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der hinzuzufügende Java-Sourcecode sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die von Ihnen bearbeiteten `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P10.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P10.zip  *.java
```

Nutzen Sie beim Bearbeiten der Aufgaben nur die Klassen und Methoden aus der Java API, welche explizit durch den Aufgabentext erlaubt sind.

Aufgabe 1: Swing Komponenten und Layout 15 Punkte

In dieser Aufgabe sollen Sie eine grafische Oberfläche für ein Schiebepuzzle (`https://de.wikipedia.org/wiki/15-Puzzle`) entwickeln. In diesem Spiel müssen 15 nummerierte Teile auf einem 4×4 Raster in die richtige Reihenfolge gebracht werden. Ein Feld in diesem Raster ist stets leer und kann durch Verschieben der horizontal und vertikal benachbarten Teile auf dieses Feld dazu verwendet werden, die Reihenfolge der Teile zu verändern.
Sie sollen nun eine Graphische Oberfläche für dieses Puzzle implementieren, welche neben einem Panel für die Puzzleteile noch ein Label besitzt, das die aktuelle Anzahl der Züge des Spielers anzeigt. Die Oberfläche sollte etwa so aussehen wie in dem folgenden Bild:



Folgende Klassen sind (teilweise) vorgegeben:

- Die von JButton abgeleitete Klasse `Zahlbutton` repräsentiert ein einzelnes Puzzleteil. Die von diesem Teil repräsentierte Zahl ist in dem Attribut `zahl` gespeichert. Intern wird die Zahl 0 verwendet, um ein Leerfeld zu repräsentieren. Über zwei Konstruktoren kann entweder ein Puzzleteil mit einer Zahl oder ein Leerfeld generiert werden. Die Beschriftung des Buttons erfolgt entweder mit der Zahl oder mit einem leeren String (für das Leerfeld). Die Klasse bietet dazu noch folgende öffentliche Methoden:

- Die Methode `getZahl` liefert den Zahlenwert eines Puzzleteils.
- Die Methode `istLeerfeld` gibt den Wert `true` zurück, wenn die Instanz, für die die Methode aufgerufen wird, das Leerfeld repräsentiert.
- Die Methode `tauscheMit` vertauscht den Inhalt zweier Buttons (d.h. deren Beschriftungen): den der aktuellen Instanz und den der Instanz, die als Parameter übergeben wird. Hierzu werden die Attributwerte ür das Attribut `zahl` sowie die Aufschrift ausgetauscht. Anmerkung: Dies verhindert, dass man die zugehörigen grafischen Komponenten auf dem Spielfeld austauschen und das Spielfeld nach jedem Zug neu erzeugen muss.

- Die Klasse `Schiebepuzzle` gibt einen Rahmen für die Oberfläche vor. Über die im Konstruktor aufgerufene Methode `createComponents` werden zwei Panels generiert: Das obere Panel (siehe Bild) enthält lediglich ein Label mit dem Wert des Zählers für die Anzahl der Züge. Die folgenden öffentlichen Methode der Klasse ermöglichen es, den Zählerwert von außen zu lesen bzw. ihn zu verändern und auch die Aufschrift des Labels ggf. anzupassen:
  - Die Methode `getAnzahlZuege` gibt den aktuellen Zählerwert zurück
  - Die Methode `erhoeheZugZaehler` erhöht den Zähler um 1
  - Die Methode `resetZugZaehler` setzt den Zähler auf 0 zurück

  Das untere Panel stellt das Spielfeld mit dem 4×4 Raster dar, auf dem die Puzzleteile (in Form von Instanzen der Klasse `ZahlButton`) arrangiert werden. Hierzu wird eine Instanz der von Ihnen noch zu vervollständigenden Klasse `PuzzlePanel` generiert und dessen bevorzugte Größe gesetzt.

- Die Klasse `PuzzlePanel` stellt das eigentliche Spielfeld mit den Puzzleteilen dar. Neben den Attributen für die Dimension `dim` des quadratischen Spielfelds und dem zweidimensionalen Feld von Puzzleteilen `teile` besitzt die Klasse ein Attribut `parent`, welches mit einer Referenz auf den Elternframe initialisiert wird. Die Referenz wird benötigt, um beim Spielen Zugriff auf den (Zug-)Zähler zu erhalten. Diese Attribute werden in dem Konstruktor der Klasse bereits initialisiert.

Die vorgegebenen öffentlichen Methoden im unteren Teil der Klasse werden von der Klasse `ZahlButton` aufgerufen, um entweder den Zählerwert im Hauptframe zu aktualisieren oder um ein Update der Oberfläche zu initiieren (entweder nach einem Zug oder um ein neues Spiel zu beginnen.

Vervollständigen Sie nun die Klasse `PuzzlePanel` wie im Folgenden beschrieben. Die Methoden (c) und (d) werden für das fertige Spiel benötigt um prüfen zu könne, ob eine Aktion des Users gültig ist bzw. der Endzustand erreicht wurde. Sie können Ihre Implementierung aber, wie in den Teilaufgaben beschrieben, durch zusätzliche Aufrufe im Konstruktor der Klasse testen.

(a) Vervollständigen sie den Konstruktor der Klasse `PuzzlePanel`, indem Sie zunächst das Feld `teile` in der Größe $n \times n$ initialisieren und zeilen- und spaltenweise mit `ZahlButton` Instanzen für die Zahlen 1 bis $\texttt{dim}^2 - 1$ (hier 15) und einem Leerteil füllen. Das Leerteil soll sich in der rechten unteren Ecke befinden. Durchmischen Sie den Inhalt des Felds dann mit Hilfe der Methode mischen (siehe Aufgabenteil (b)) Setzen Sie das Layout des Panels auf ein `GridLayout` mit n Zeilen und n Spalten und füllen Sie das Panel schließlich mit den `ZahlButton` Instanzen aus dem Feld `teile`.

(b) Implementieren Sie die Methode `mischen`, welche mit Hilfe der Methode `tausche` aus der Klasse `ZahlButton` 50 Vertauschungen von je zwei zufällig gewählten Puzzleteilen vornimmt. Wählen Sie hierzu 50 Mal je zwei zufällige Werte für den Zeilen- und den Spaltenindex und tauschen Sie den Inhalt der zugehörigen `ZahlButton` Instanzen in dem Feld `teile` aus.

Anmerkung: Durch diese Vertauschungen kommen stets Anordnungen der Teile zustande, die auf eine von zwei möglichen Weisen lösbar sind: Hierbei sind alle nummerierten Steine zeilenweise aufsteigend angeordnet und das Leerfeld befindet sich entweder links oben oder rechts unten.

(c) Implementieren Sie die Methode `checkLeerteilNachbar`, die überprüfen soll, ob einer der horizontalen und vertikalen Nachbarn des als Parameter übergebenen Teils das Leerfeld ist. Bestimmen Sie zunächst den Zeilen- und Spaltenindex des Teils in dem Feld `teile`, indem Sie das Teil in dem Feld suchen. Bestimmen Sie dann die die Nachbarn des

Teils in dem Feld `teile` (je nach Lage des als Parameter übergebenen Puzzleteils 2 - 4 Stück). Ist einer dieser Nachbarn tatsächlich das Leerfeld, geben Sie die Referenz darauf als Ergebnis zurück und ansonsten den Wert `null`.

Testen Sie Ihre Implementierung, indem Sie die Methode im Konstruktor für einzelne (oder alle) Puzzleteile in dem Feld aufrufen und das Ergebnis jeweils in der Konsole ausgeben.

(d) Implementieren Sie die Methode `fertig`, die überprüfen soll, ob die aktuelle Anordung der Teile einen Endzustand darstellt. Hierzu müssen alle Teile zeilen- und spaltenweise aufsteigend sortiert sein und das Leerfeld muss sich entweder links oben oder rechts unten befinden. Die Methode soll den Wert `true` zurückgeben, wenn ein Endzustand vorliegt und den Wert `false` sonst.

Hinweis: Zum Prüfen der Reihenfolge können Sie die Zahlen auf allen Puzzleteilen, die nicht dem Leerfeld entsprechen, nacheinander in ein Feld der Länge 15 kopieren und dann prüfen, ob die Zahlen darin geordnet sind oder nicht.

Zum Testen Ihrer Implementierung rufen Sie die Methode im Konstruktor jeweils einmal vor und nach dem Mischen auf und geben Sie das Ergebnis aus.

Wenn Sie nun die Klasse Schiebepuzzle starten, sollte die Oberfläche (bis auf die Reihenfolge der Zahlen) wie in dem obigen Screenshot aussehen.

Aufgabe 2: Swing Komponenten und Layout 8 Punkte

In dieser Aufgabe sollen Sie eine Oberfläche erstellen, mit deren Hilfe Sie sich Daten zu den Oscar-Nominierungen in den Jahren 1928 - 2024 für eine ausgewählte Menge von Kategorien ansehen können. Hierzu ist Folgendes vorgegeben:

- Die Datei `academyAwards.csv` enthält die Namen zu 2390 für den Oscar nominierten Personen in den Kategorien bester Schauspieler / beste Schauspielerin in einer Hauptrolle bzw. Nebenrolle sowie Regie in den Jahren 1928 - 2024. Hierzu wird jeweils der zugehörige Film genannt und ob die Person den Preis gewonnen hat oder nicht.

- Die Klasse `Nominee` repräsentiert eine Nominierung einer Person für einen Oscar und wird durch Attribute für den Namen der Person und für den zugehörigen Film sowie durch die Kategorie und das Jahr der Nominierung beschrieben. Ein weiteres Attribut gibt an, ob die Person den Preis gewonnen hat. Neben einem Konstruktor, mit dem alle Attribute initialisiert werden, besitzt die Klasse folgende Attribute und Methoden.

  - Die öffentliche Klassenvariable `CATEGORIES` stellt ein Feld mit Strings für alle hier betrachteten Kategorien dar.
  - Für die Attribute `year`, `category` und `hasWon` gibt es Getter-Methoden, um von außen auf die Werte zuzugreifen.
  - Die `toString` Methode ist überschrieben und gibt eine einfache String-Repräsentation einer Nominierung zurück.
  - Die Klassenmethode `getNominees` liest die Daten aus der Datei `academyAwards.csv` ein, erzeugt dann aus jeder Zeile dieser Datei eine Instanz der Klasse `Nominee` und gibt schließlich ein Feld mit diesen Instanzen zurück.

- Die Klasse `FilterUtils` stellt eine Klassenmethode zur Verfügung, mit deren Hilfe alle Nominierungen aus einem als Parameter übergebenen Feld von `Nominee` Objekten gefiltert werden können, die in einem ebenfalls als Parameter übergebenen Jahr stattgefunden haben. Über zwei weitere Parameter kann darüberhinaus bestimmt werden, für welche Kategorie die Nominierungen stattgefunden haben (dabei steht "ALL" für beliebige Kategorien) und ob alle Nominierten oder nur die Gewinner einer Kategorie ausgfiltert werden sollen.

- Die von Ihnen zu ergänzende Klasse `OscarGUI` stellt die GUI dar. Diese Klasse ist wie folgt ausgestattet:

  - Das Feld `nominees` beinhaltet alle Nominierungen (Objekte der Klasse `Nominee`).
  - Für die beiden, in dieser GUI benötigten Comboboxen sind bereits zwei Instanzvariablen vorgegeben, die mit einer leeren Combobox initialisiert werden.

- Im Konstruktor wird der Basisframe erzeugt und die Methode `createComponents` aufgerufen, in der die GUI-Komponenten erstellt und zusammengebaut werden. In der `main`-Methode wird dieser Konstruktor dann aufgerufen und die GUI auf den Bildschirm gebracht.
- Die Methode `createComponents` ruft zwei weitere Methoden auf, in denen jeweils ein Panel erstellt wird und die dann mit einem Rahmen versehen und der `ContentPane` hinzugefügt werden.
- Die Methode `createText` erzeugt aus einem Feld von Nominierungen einen Text für die Anzeige in dem Textbereich der GUI.
- Die Methode `createCombinedBorder` umgibt eine als Parameter übergebene Komponente mit einem (zweiteiligen) Rahmen.
- In den Methoden `createControlPanel` und `createTextPanel` werden aktuell nur zwei leere Panels erzeugt, für die Sie noch Komponenten generieren und dem jeweiligen Panel hinzufügen müssen (siehe Aufgabenstellung).

Ergänzen Sie die Klasse `OscarGUI` nun wie im Folgenden beschrieben:

(a) Ergänzen Sie die Methode `createControlPanel` wie folgt:

- Fügen Sie der (als Instanzvariable vorgegebenen) Combobox `years` Items für die Jahre 1928 - 2024 hinzu. Sorgen Sie dafür, dass das Jahr 2024 in der Combobox ausgewählt ist.
- Fügen Sie der (als Instanzvariable vorgegebenen) Combobox `categories` als Items zunächst den String "ALL" sowie danach alle Strings aus dem Feld `CATEGORIES` (in der Klasse `Nominee`) hinzu. Sorgen Sie dafür, dass das erste Item in der Combobox ausgewählt ist.
- Definieren Sie eine weitere Instanzvariable vom Typ `JCheckBox` und initialisieren Sie diese in der Methode mit einer Checkbox ohne Text und unselektiert.
- Legen Sie lokal drei `JLabel` Objekte an, die Sie mit den Beschreibungen für die beiden Comboboxen und der Checkbox initialisieren (siehe Screenshot unten). Der Text in den Labels sollte dabei rechtsbündig dargestellt werden.

- Setzen Sie für das Panel `p` ein `GridLayout` mit drei Zeilen und zwei Spalten. Sorgen Sie dafür, dass zwischen den Spalten jeweils 20 Pixel und zwischen den Zeilen jeweils 5 Pixel Platz gelassen wird. Fügen Sie dann die Labels, die beiden Comboboxen und die Checkbox dem Panel hinzu, so dass die Labels in der linken Spalte und die Comboboxen sowie die Checkbox passend dazu in der rechten Spalte angeordnet werden (siehe Screenshot).
- Damit später das Panel mit den von Ihnen generierten Komponenten nicht die ganze Breite des Frames einnimmt (welcher sich hier durch die Breite des unteren Panels ergibt), legen Sie nun noch ein weiteres `JPanel p2` an. Setzen Sie die bevorzugte Größe des Panels `p` auf 350×100 und fügen Sie es dem Panel `p2` hinzu und geben Sie letzteres als Ergebnis der Methode zurück.

(b) Implementieren Sie die Methode `createTextPanel` wie folgt:

- Legen Sie zunächst in der Klasse eine weitere Instanzvariable vom Typ `JTextArea` an, die Sie in der Methode dann mit einem Textbereich initialisieren, der 15 Zeilen und 40 Spalten haben soll. Umgeben Sie den Textbereich mit Hilfe der Methode `createCombinedBorder` mit einem Rahmen.
- Legen Sie dann lokal in der Methode eine `JScrollPanel` Instanz an, welche den Textbereich enthält.
- Füllen Sie den Textbereich mit einem String, welcher sich beim Aufruf der Methode `createText` mit dem vorgegebenen Feld `nmArr` als Parameter ergibt.
- Fügen Sie das `JScrollPane` Objekt dem vorgegebenen Panel `p` hinzu.

(c) Definieren Sie sich nun noch zwei Hintergrundfarben (für die Panels bzw. den Textbereich) als Klassenkonstanten in der Klasse `OscarGUI` und ergänzen Sie die Methoden `createControlPanel` und `createTextPanel` so, dass die Hintergrundfarben für die Komponenten in jedem Panel einheitlich gesetzt werden. Setzen Sie die Hintergrundfarbe in der Methode `createControlPanel` nur für das Hauptpanel (welches am Ende zurückgegeben wird) und machen Sie das zusätzliche Panel undurchsichtig. Verwenden Sie hierbei nicht die vordefinierten Farben der Klasse `Color`. Hinweis: Um die RGB Werte für die Farben Ihrer Wahl zu

bestimmen, können Sie ein sogenanntes Color Picker Tool verwenden, wie Sie es z.B. auf der Seite `https://www.w3schools.com/colors/colors_picker.asp` finden.

Die GUI sollte beim Start der Klasse OscarGUI (bis auf die verwendeten Farben) wie in folgendem Screenshot aussehen: