Projektblatt P10 (20 + 5* P)

Abgabe: Freitag 24. November 2023, 12:00h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p10.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben sowie ein Unterverzeichnis mit Bildern 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 `.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
```

Aufgabe 1: ActionListener als externe Klasse 8 Punkte

In dieser Aufgabe sollen Sie nun das Memory Spiel vom Projektblatt P9 so erweitern, dass Karten aktiv ausgewählt und das Spiel gespielt werden kann. Hierzu sind die Klassen `Resources`, `Karte`, `KartenButton`, `ZaehlerLabel` vollständig, sowie die Klassen `MyMemory` und `KartenListener` teilweise vorgegeben. Die Klasse `KartenListener` gibt dabei einen Rahmen vor, der zwei Attribute für die beiden Labels beinhaltet, deren Text vom Listener beeinflußt werden. Diese beiden Attribute werden vom Konstruktor der Klasse initialisiert.
Gehen Sie wie folgt vor:

- Generieren sie in der Methode `createListener` in der Klasse `MyMemory` zunächst eine Instanz der Klasse `KartenListener` und weisen Sie diese Instanz allen `KartenButton` Instanzen als Listener hinzu.

  Hinweis: Die `KartenButton` Instanzen sind in dem vorgegeben Feld `buttons` enthalten und werden mit Hilfe der Methode getComponents aus dem (als Attribut definierten) Panel `buttonPanel` bestimmt. Sie müssen die Objekte allerdings vor dem Hinzufügen eines Listeners passend casten.

  Der Aufruf der Methode `createListener` ist bereits im Konstruktor der Klasse `MyMemory` vorgegeben.

- Realisieren Sie nun den Spielablauf, indem Sie das Verhalten der Buttons beim Anklicken definieren. Ergänzen Sie dazu die Klasse `KartenListener`. Grundsätzlich soll das Spiel so ablaufen, dass mit Hilfe von zwei Buttonklicks zwei verschiedene Karten aufgedeckt werden. Mit einem dritten Mausklick (auf eine beliebige Karte) soll dann die Situation ausgewertet werden: Sind beide Karten gleich, sollen die Karten vom Spielfeld entfernt werden und der Zähler für die Paare um eins erhöht werden. Sind die beiden Karten nicht gleich, sollen die beiden aufgedeckten Karten wieder umgedreht (verdeckt) werden. In beiden Fällen (zwei ausgewählte Karten gleich / ungleich) soll der Zähler für die Anzahl der Versuche um eins erhöht werden. Beim Auswählen der zweiten Karte ist dabei zu berücksichtigen, dass ein Klick auf die zuvor bereits ausgewählte Karte nichts bewirken darf. Erhöhen Sie die Zählwerte der Labels dabei mit Hilfe der Methode `increase` aus der Klasse `ZaehlerLabel`.

Hinweise:

- Nutzen Sie die vorgegebenen Hilfsvariablen `count` und `buttons` :
  - Nutzen Sie die Variable `count`, um die Anzahl der Klicks mitzählen (und ggf. zurücksetzen). Dann können Sie die Reaktion des Listeners davon abhängig machen, wieviele Klicks in einem Zug (der aus 3 gültigen Klicks besteht) bereits erfolgt sind.
  - Nutzen Sie die das Feld `buttons` für die beiden jeweils ausgewählten Karten, die vom User in einem Zug aufgedeckt werden.
- Mit der Methode `getSource` aus der Klasse `java.util.EventObject` (die von der Klasse `ActionEvent` geerbt wird), können Sie herausfinden, welche `KartenButton` Instanz das Event ausgelöst hat.
- Mit der Methode `istPaar` aus der Klasse `KartenButton` können Sie überprüfen, ob zwei Karten ein Paar bilden.
- Mit der Methode `setEnabled` können Sie einen Button deaktivieren bzw. reaktiveren. Ein Klick auf einen deaktivierten Button löst kein `ActionEvent` aus.
- Die Methode `setSelected` ermöglicht es Ihnen, den Zustand (selektiert/nicht selektiert) eines Buttons zu setzen. Hiermit können sie Karten aktiv "umdrehen" (Vorderseite = selektiert / Rückseite = nicht selektiert).
- Um eine Karte (virtuell) zu entfernen, können Sie die Methode `entferneKarte` der Klasse `KartenButton` verwenden. Diese deaktiviert den Button und setzt eine Zustandsvariable, so dass der Bereich für die Karte im Panel auf die Hintergrundfarbe gesetzt wird.

Aufgabe 2: Listener (als innere Klassen) 7 + 5* Punkte

In dieser Aufgabe sollen sie den Terminkalender, für den Sie auf dem letzten Projektblatt eine GUI erstellt haben, mit den Möglichkeiten für eine Interaktion mit einem Benutzer versehen, so dass Termine erstellt und angezeigt werden können. Dazu benötigen Sie als Voraussetzung:

- die Klasse, die Sie für die Aufgabe 2 auf dem Blatt P9 entwickelt haben. Alternativ können Sie die vorgegebene Klasse `TerminGUI` aus dem Projektarchiv verwenden,
- die Klassen `Termin` (zur Repräsentation eines einzelnen Termins) und `Terminliste` (zur Speicherung einer Liste von Einzelterminen) aus dem Projektarchiv.

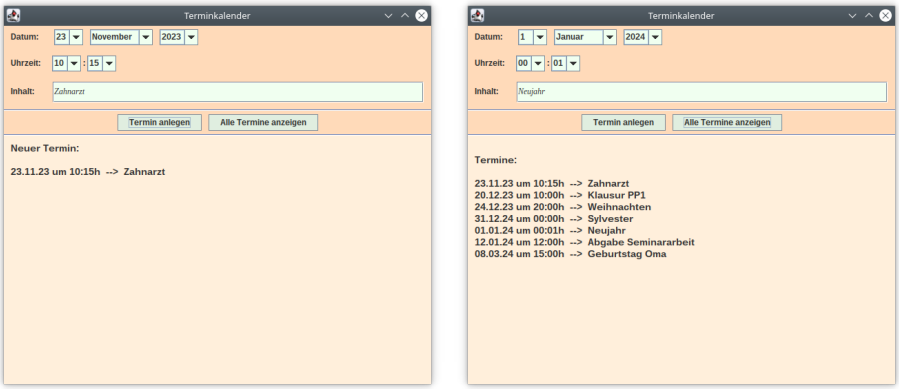
Falls Sie mir ihrer eigenen GUI-Klasse arbeiten

- stellen Sie zunächst sicher, dass alle Komponenten, auf die Sie aus den Methoden der Listener zugreifen müssen (Comboboxen, Tetfelder und -bereiche, Buttons), als (private) Attribute in der Klasse definiert sind.
- Definieren Sie ein weiteres Attribut vom Typ `Terminliste` und initialisieren Sie dieses mit Hilfe eines Konstruktors dieser Klasse.
- Definieren Sie eine (noch leere) parameterlose Methode mit dem Namen `createListener`, die im Konstruktor der GUI-Klasse aufgerufen wird, nachdem die Komponenten alle initialisiert wurden.

Ergänzen Sie die GUI-Klasse dann um zwei innere Klassen, die das Interface `ActionListener` implementieren.

- Mit Hilfe der ersten Listener-Klasse sollen zunächst die Werte aus den Comboboxen für das Jahr, den Monat usw. extrahiert und in passende `int`-Werte umgewandelt werden. Zusammen mit dem Text aus dem Textfeld der GUI, mit dem der Inhalt des Termins beschrieben wird, soll dann eine Instanz der Klasse `Termin` erzeugt werden. Diese Instanz soll in die als Attribut vorgegebene Liste der Termine in der umgebenden Klasse `TerminGUI`) eingefügt werden (mit Hilfe der Methode `addTermin` aus der Klasse `Terminliste`). Geben Sie den Termin dann zusammen mit der Überschrift "Neuer Termin" im dem unterem Textbereich aus. Ein Beispiel ist im linken Bild in der Abbildung unten zu sehen.

- Mit Hilfe der zweiten Listenerklasse soll eine Liste der (nach dem Datum sortierten) Termine in dem unteren Textbereich ausgegeben werden bzw. die Ausgabe "Keine Termine", wenn die Liste leer ist. Ein Beispiel ist im rechten Bild in der folgenden Abbildung zu sehen. Eine Sortierung der Termine innerhalb der Klasse Terminliste kann durch die Methode `sortTermine` (aus der Klasse `Terminliste`) veranlasst werden.



- Ergänzen Sie die Methode `createListener` wie folgt: Fügen Sie dem einen Button eine Instanz der ersten Listener-Klasse als `ActionListener` hinzu und dem zweiten Button eine Instanz der zweiten Listener-Klasse.

Bonusaufgabe (5* P)

Schreiben Sie zwei weitere Listenerklassen (vom Typ `ItemListener`), welche Sie den Comboboxen für das Jahr und den Monat zuordnen und die dafür sorgen, dass bei einer Auswahl eines Jahrs oder eines Monats die Liste der Tage in der Combobox für die Tage ggf. angepasst wird. Beispiel: Aktuell sind für jeden Monat die Wert1 1- 31 auswählbar. Mit Hilfe des Listeners sollten dann beispielsweise für den Februar 2023 und 2025 die Werte 1 - 28 und für den Februar 2024 die Werte 1 - 29 in der Comboxbox zur Verfügung stehen. Sie dürfen hierzu die in der Klasse `TerminGUI` vorgegebene Methode `fillComboBoxTage` verwenden.

Aufgabe 3: JMenu + anonymer Listener 5 Punkte

Die (partiell vorgegebene) Klasse `ImageViewer` soll dazu dienen, verschiedene Bilder in einem `ScrollPanel` anzuzeigen. Hierzu wird eine Instanz der vorgegebenen Klasse `ImagePanel` verwendet, mit deren Methode `setImage` das jeweils anzuzeigende Bild gesetzt werden kann. Die Methode benötigt dazu einen Parameter vom Typ `String` oder vom Typ `File`, welcher den Pfadnamen der Datei (absoluter oder relativer Pfad) repräsentiert.
Ergänzen Sie die noch leere Methode `createMenu` der Klasse `ImageViewer` wie folgt:

- Erzeugen Sie ein Menü mit dem Namen "File". In dem Menü soll es einen Eintrag mit dem Namen "Bild laden" geben. Fügen Sie das Menü dann in einer neuen Menüleiste ein und setzen Sie diese in der Klasse `ImageViewer` als `JMenuBar`.

- Fügen Sie dem Menüeintrag "Bild laden" eine anonyme `ActionListener`-Instanz als Listener hinzu. Wird der Menüpunkt dann zur Laufzeit ausgewählt, soll mit Hilfe der `JFileChooser`-Komponente ein Bild geladen und angezeigt werden können. Gehen Sie dazu in der Methode `actionPerformed` wie folgt vor:

  - Generieren Sie eine Instanz der Klasse `JFileChooser`.
  - Nutzen Sie die Methode `showOpenDialog`, um zur Laufzeit ein Dialogfenster anzuzeigen, mit dem eine zu ladende Datei ausgewählt werden kann. Als Parameter für diese Methode können Sie die `ImageViewer` Instanz in Form des Ausdrucks `ImageViewer.this` verwenden.
  - Hat der User zur Laufzeit eine Datei ausgewählt, können Sie dieses mit Hilfe der Methode `getSelectedFile` aus der `JFileChooser` Instanz auswählen und über die Methode `setImage` der `ImagePanel` Instanz zur Anzeige übergeben.

  Sehen Sie sich dazu auch die Beschreibung der Klasse `JFileChooser` in der API Dokumentation an.

Hinweis: Um aus einer inneren Klasse auf ein Attribut der umgebenden Klasse zuzugreifen, kann man entweder den Namen des Attributs verwenden oder den Namen der umgebenden Klasse und dazu das this-Attribut.

Ist der Name der äußeren Klasse `Outer` und hat das Attribut den Namen `att`, dann lautet der Zugriff `att` oder `Outer.this.att`. Ein Zugriff mit `this.att` funktioniert dagegen nicht, da sich `this` hierbei auf die innere Klasse bezieht. Ist der Attributwert eine Referenz, kann man damit direkt auch auf die Methoden des zugehörigen Objekts zugreifen.