Aufgabenblatt 6

Kompetenzstufe 1 & Kompetenzstufe 2

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Donnerstag, 15.06.2023 12:00 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, ob Sie die Aufgabe gelöst haben.
- Ihr Programm muss kompilierbar und ausführbar sein.
- Ändern Sie bitte **nicht** die **Dateinamen** und die **vorhandene Ordnerstruktur**.
- Bitte beachten Sie die Vorbedingungen! Sie dürfen sich darauf verlassen, dass alle Aufrufe die genannten Vorbedingungen erfüllen. Sie müssen diese nicht in den Methoden überprüfen.
- Bitte achten Sie auch darauf, dass Sie eine eigenständige Lösung erstellen. Wir werden bei dieser Aufgabe wieder auf Plagiate überprüfen und offensichtliche Plagiate nicht bewerten.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- Ein- und zweidimensionale Arrays
- Methoden
- Grafische Darstellung
- Spiellogik

Aufgabe 1 (6 Punkte)

Implementieren Sie folgende Aufgabenstellung:

- Bei dieser Aufgabe sollen Sie das Spiel *Mastermind*[1] implementieren. Bei diesem Spiel versucht eine spielende Person, die vom Computer generierte Farbkombination zu erraten. In Abbildung 1 sehen Sie das Spielfeld mit den klickbaren Buttons an der rechten Seite und dem Spielfeldbereich links, der aus den weißen Kreisen plus dem grauen Bereich für zusätzliche Hinweise besteht.

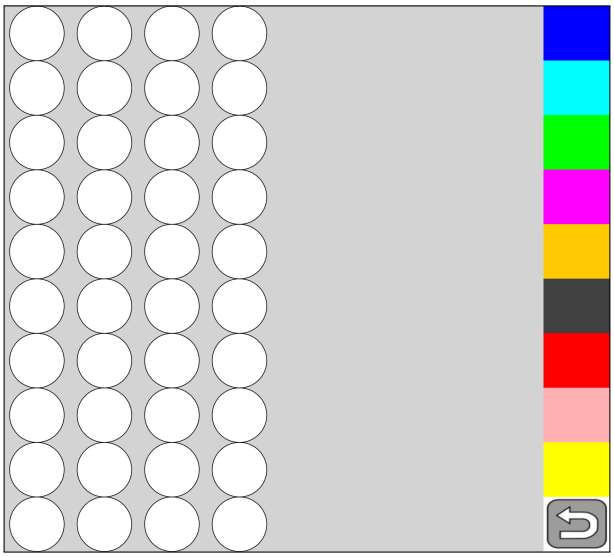

Abbildung 1: *Mastermind* Spielfeld mit klickbaren Buttons rechts und dem Visualisierungsbereich links, bestehend aus weißen Kreisen und einem grauen Bereich für Hinweise.

- Sie haben die Methode `main` bereits vorgegeben. In der Methode `main` werden die Einstellungen für das `CodeDraw`-Ausgabefenster definiert, sowie ein `EventScanner` zum verarbeiten der Mausklicks angelegt. Zusätzlich sind bereits Methodenaufrufe für das Spiel vorhanden.

  Beschreibung der globalen Konstanten:

  - `NUMBER_OF_TURNS`: Gibt die maximale Anzahl der Versuche für das Erraten der korrekten Farbkombination an.
  - `CODE_LENGTH`: Gibt die Länge der Farbkombination an, die erraten werden muss.
  - `NUMBER_OF_COLUMNS`: Gibt die Anzahl der Spalten an, in die das gesamte Ausgabefenster aufgeteilt wird. Diese Spaltenanzahl entspricht der doppelten Codelänge plus eins für die Spalte mit den klickbaren Buttons auf der rechten Seite.
  - `COLORS`: Repräsentiert das Array mit den verwendbaren Farben für den Code, die auch ganz rechts bei den klickbaren Buttons verwendet werden.

  Das Spiel muss mit den vordefinierten Werten für die Konstanten funktionieren. Sie dürfen die Implementierung generisch gestalten, sodass das Spiel mit adaptierten Konstanten ebenfalls funktioniert.

[1] `https://de.wikipedia.org/wiki/Mastermind_(Spiel)`

Beschreibung der globalen Variablen:

- `playField`: Dieses Array beschreibt das Spielfeld und ist `NUMBER_OF_TURNS` × `CODE_LENGTH` groß (10×4 in Abbildung 1). Jedes Element des Arrays ist ein ganzzahliger Wert im Intervall `[0, COLORS.length]`: Der Wert 0 codiert, dass die spielende Person noch keine Farbe ausgewählt hat, die anderen Werte entsprechen den Indizes der ausgewählten Farben im Array `COLORS` plus eins.
- `tips`: Dieses Array dient zur Speicherung der Hinweise nach jeder Runde und ist ebenfalls `NUMBER_OF_TURNS` × `CODE_LENGTH` Einträge groß. Hier wird 1 als Codierung für einen roten und 2 für einen weißen Pin verwendet.
- `turn`: Dieser Wert dient als Zähler für die Spielrunden.
- `pin`: Diese Variable beschreibt die aktuelle Position innerhalb einer Spielrunde.
- `solution`: Dieses Array wird für die Speicherung des zufällig generierten Farbcodes verwendet. Es werden vier verschiedene Zahlen von 1 bis `COLORS.length` abgelegt, die die Indizes der Farben im Array `COLORS` plus 1 darstellen.

Spielablauf: In `main` wird nach dem Aufruf der Methode `initGanme` und die initiale Zeichnung des leeren Spielfeldes mit `drawGame` das Spiel gestartet. Dazu wird die Methode `playGame` aufgerufen wo dann in einer Schleife auf Mausklicks gewartet wird. Diese Methode wertet aus, ob das Spiel gewonnen (Farbkombination erraten), verloren (Rateversuche aufgebraucht), oder durch das Drücken der Taste `'q'` abgebrochen wurde. Wir nehmen für die Erklärung an, dass die zu erratende Farbkombination jene ist, die in Abbildung 2 dargestellt ist.

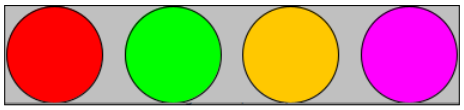

Abbildung 2: Eine zufällig generierte Farbkombination.

Das Spiel beginnt damit, dass die spielende Person auf der rechten Seite die Buttons mit den Farben anklickt. Es müssen vier verschiedene Farben angeklickt werden, um einen Rateversuch zu beenden: Klickt die spielende Person auf eine bereits ausgewählte Farbe, wird der Klick einfach ignoriert. Während eines Rateversuchs können Eingaben rückgängig gemacht werden (siehe Abbildung 3a und Abbildung 3b). Dazu muss der Button ganz rechts unten angeklickt werden. Diese Funktion ist solange aktiv, bis die letzte Farbe eines Rateversuchs eingegeben wurde, denn dann wird der Rateversuch automatisch beendet und ausgewertet (siehe Abbildung 3c). Bei dieser Auswertung werden Hinweise für die spielende Person generiert. Für jede Farbe an einer richtigen Position bekommt die spielende Person einen roten Pin (roter Farbkreis). Ist die Farbe im zu erratenden Code vorhanden, aber an einer anderen Stelle, dann wird dies mit einem weißen Pin (weißer Farbkreis) als Hinweis angezeigt. Danach beginnt der nächste Rateversuch eine Zeile weiter oben (siehe Abbildung 3d) und es können wieder die Farbbuttons rechts angeklickt werden. Wenn nach Eingabe einer Farbkombination alle Farben an der richtigen Stelle erraten wurden, dann wird dies mit vier roten Pins angezeigt und das Spiel wurde gewonnen (siehe Abbildung 3e). Wenn der letzte Rateversuch ganz oben beendet wurde und bis dahin die richtige Farbkombination nicht gefunden wurde, dann hat die spielende Person verloren (siehe Abbildung 3f). Für die Ausgaben in den Abbildungen 3e und 3f können Sie die bereits vorhandene Methode `showMessage` verwenden.

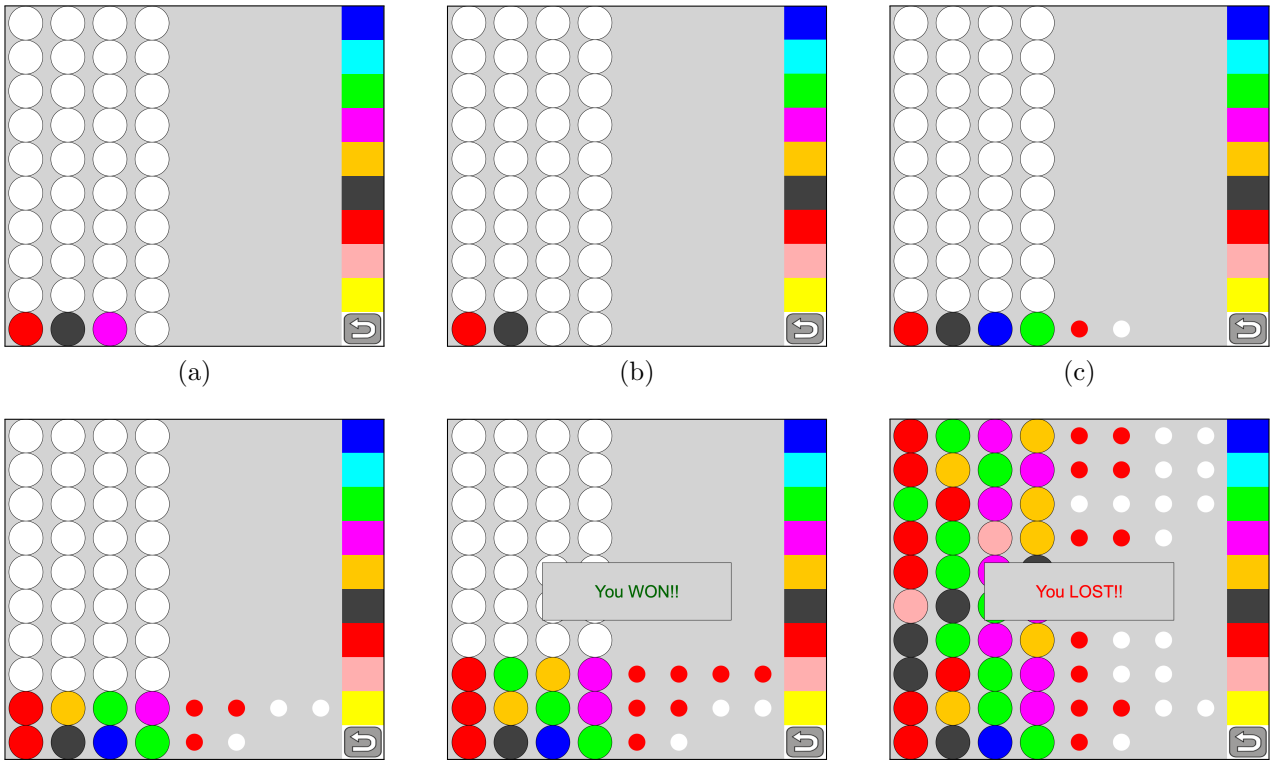

Abbildung 3: Verschiedene Spielzustände des Spiels Mastermind.

Für die Komplettierung des Spiels müssen Sie die folgenden fünf Methoden implementieren[2]:

- Implementieren Sie eine Methode `generateCode`:

  ```
  int[] generateCode()
  ```

  Die Methode generiert eine zufällige Farbkombination. Dazu werden `CODE_LENGTH` Farben aus dem Array `COLORS` gewählt, die als ganzzahlige Werte codiert (`1` == `Color.BLUE`, `2` == `Color.GREEN`, ... , `9` == `Color.YELLOW`) und in ein neues Array abgespeichert werden. Jede Farbe darf nur einmal vorkommen. Anschließend wird das neu erstellte Array zurückgegeben.

- Implementieren Sie eine Methode `updateTips`:

  ```
  void updateTips()
  ```

  Diese Methode vergleicht die beiden Arrays `solution` und `playfield[turn]` und speichert im Array `tips[turn]` Hinweise für die spielende Person. In `playfield[turn]` befindet sich die Farbkombination der spielenden Person im aktuellen Spielzug. Zuerst wird verglichen, ob sich Farben im Array `playfield[turn]` bereits an der korrekten Position befinden. Dazu wird jeder Arrayeintrag von `playfield[turn]` (gespeicherte Farbe des Rateversuchs) mit dem gleichen (gleicher Indexwert) Arrayeintrag von `solution` (gespeicherte Farbe der zu erratenden Farbkombination) verglichen. Wenn eine Übereinstimmung gefunden wurde, dann

[2]Sie dürfen daneben auch eigene Hilfsmethoden implementieren.

wird in `tips[turn]` ein roter Pin (Wert 1) hinterlegt. Die Reihenfolge der Pins innerhalb des Arrays ist nicht relevant. Falls nicht alle Farben an der richtigen Position sind, dann wird überprüft, ob Farben der zu erratenden Kombination `solution` in `code` an einer anderen Position vorhanden sind. Wenn eine solche Farbe gefunden wurde, dann wird in `tips[turn]` ein weißer Pin (Wert 2) hinzugefügt.

Vorbedingungen: $0 \leq$ `turn` $<$ `NUMBER_OF_TURNS`, `tips[turn] != null`, `playfield[turn] != null`, `solution != null`, `tips[turn].length == CODE_LENGTH`, `solution.length == CODE_LENGTH` und `playfield[turn].length == CODE_LENGTH`.

- Implementieren Sie eine Methode `drawGame`:

```
void drawGame(CodeDraw myDrawObj)
```

Die Methode zeichnet das Spielfeld des Mastermindspiels in ein CodeDraw-Ausgabefenster. Das Fenster ist bereits vordefiniert und hat eine Größe von 880×800 Pixel. Die Breite ist um 80 Pixel größer als die Höhe, da diese 80 Pixel für die Spalte mit den klickbaren Buttons verwendet wird. Die größeren weißen Kreise zeigen nach dem Anklicken die eingegebene Farbkombination der spielenden Person an. Rechts neben den Kreisen für die Farbkombinationen ist ein Platz für die Anzeige der Hinweise nach jedem Rateversuch vorgesehen. Hier werden nebeneinander die roten und weißen Pins, falls vorhanden, gezeichnet. Die genauen Abmessungen, Abstände und Positionen der Kreise bleiben Ihnen überlassen und können frei gestaltet werden.

Ganz rechts ist die Spalte mit den zur Verfügung stehenden Farben und ganz rechts unten der Button, um eine Eingabe rückgängig zu machen. Jeder dieser klickbaren Buttons hat in dieser Spielkonfiguration eine Größe von 80×80 Pixel.

Vorbedingungen: `playField != null`, `tips != null` und die Dimensionen der Arrays `playField` und `tips` sind gleich groß.

- Implementieren Sie eine Methode `playGame`:

```
void playGame(CodeDraw myDrawObj, EventScanner myEventSC)
```

Die Methode wird in `main` aufgerufen und beinhaltet das oberste Spielmanagement. Es werden nach einem Mausklick die entsprechenden Mauskoordinaten ausgelesen und dann ein Spielzug durchgeführt, wie im Abschnitt *Spielablauf* erklärt. Wird das Spiel nach dem Zug gewonnen oder verloren, wird der spielenden Person eine entsprechende Nachricht für 3 Sekunden[3] angezeigt (wie in den Abbildungen 3e und 3f veranschaulicht). Nach dieser Pause von 3 Sekunden wird das Spielfeld gelöscht (siehe Beschreibung der nächsten Methode) und danach ein neues Spiel gestartet. Das Spiel kann aber jederzeit durch Drücken der Taste `'q'` abgebrochen werden.

[3]Dazu können Sie die Methode `show` der Klasse `CodeDraw` mit einem Parameter vom Typ `int` aufrufen, der die Anzahl der zu wartenden Millisekunden angibt.

- Implementieren Sie eine Methode `clearBoard`:

  ```
  void clearBoard(CodeDraw myDrawObj)
  ```

  Diese Methode muss in Ihrer Implementierung der Methode `playGame` aufgerufen werden, nachdem das Spiel gewonnen oder verloren wurde und eine entsprechende Nachricht für 3 Sekunden angezeigt wurde. Die Methode löscht dann das Spielfeld zeilenweise von oben nach unten und zeigt nach dem Löschen jeder Zeile das Spielfeld für 400 Millisekunden an, wie in Abbildung 4 veranschaulicht.

  Vorbedingungen: $0 \leq$ `turn` $<$ `NUMBER_OF_TURNS`, `playField != null`, `tips != null` und die Dimensionen der Arrays `playField` und `tips` sind gleich groß.

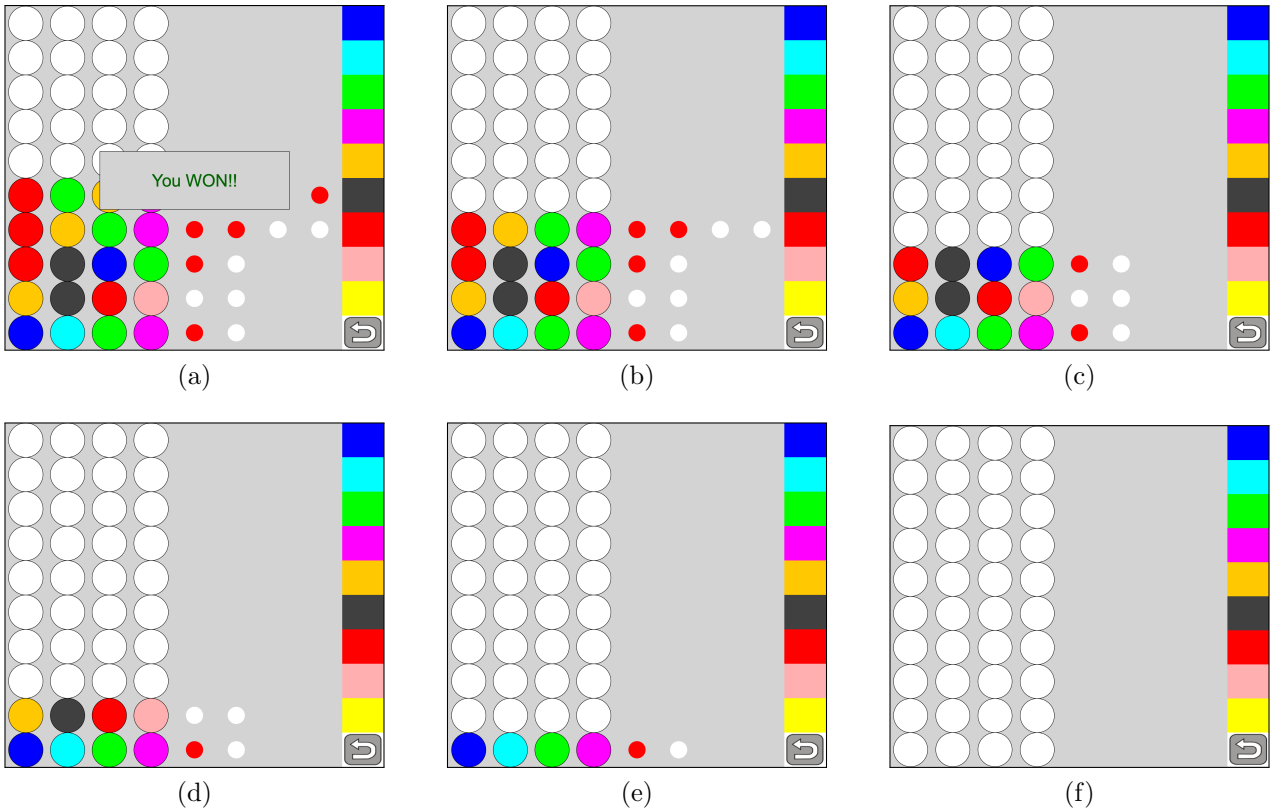

Abbildung 4: Arbeitsweise der Methode `clearBoard`.