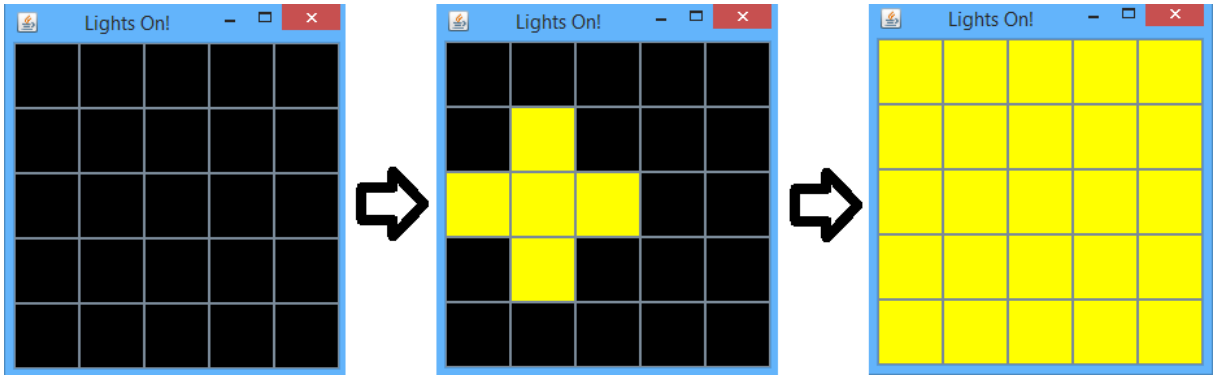
Programmierung A/B (in Java) – Übung 11

Thema: *GUIs, Felder, Klassen*

Lernziele: *Sie sollen sich mit der Erstellung von grafischen Nutzeroberflächen vertraut machen*

Aufgabe 1: Hello World in Swing
Erstellen Sie eine Swing Anwendung, die ein Fenster erzeugt. Auf diesem Fenster soll ein Text zur Begrüßung stehen („Hello World“) sowie ein Eingabefeld und ein Knopf. Wird der Knopf gedrückt, soll der Inhalt des Eingabefeldes an der Stelle des Begrüßungstextes angezeigt werden.

Aufgabe 2: Ein kleines Puzzle-Spiel
In dieser Aufgabe sollen Sie eine Swing Anwendung erstellen, die mit Hilfe der MVC-Architektur ein einfaches Puzzle-Spiel realisiert. Innerhalb eines Fensters soll sich ein n*n großes Feld von Lichtern (Buttons) befinden, welche entweder ein- oder ausgeschaltet (schwarz oder gelb gefärbt) sein können. Zu Beginn sollen alle Lichter ausgeschaltet sein (siehe Bild). Durch einen Klick auf eines der Lichter sollen die Zustände dieses Lichtes, sowie der direkt angrenzenden Lichter gewechselt werden. Ziel dieses Spiels soll es sein, das gesamte Feld zu erleuchten.



Zur Realisierung erstellen Sie zunächst vier Klassen: **Model**, **View**, **Controller** und **Main**

Beginnen Sie mit der Klasse **Model**, der Datenrepräsentation Ihres Spiels. Ergänzen Sie dort eine private Membervariable eines zweidimensionalen Arrays vom Typ boolean. Dort wird der Zustand der Lichter abgespeichert. Die Größe des Arrays soll über den Konstruktor festlegbar sein. Weiterhin implementieren Sie noch zwei weitere Methoden:

- *void toggleField(int x, int y)*, um den Zustand eines Lichtes an der gegebenen Stelle zu wechseln. Um einer möglichen IndexOutOfBounds Exception zu entgehen, prüfen Sie als erstes die Gültigkeit der Parameter und führen die Operation nur aus, wenn sich diese innerhalb des Arrays befinden.
- *boolean isCleared()*, um zu prüfen, ob alle Lichter angeschaltet sind und damit das Rätsel gelöst wurde.

Widmen Sie sich nun der Klasse **View**, die Sie von der Klasse **JFrame** erben lassen. Dies wird die grafische Komponente Ihres Spiels werden. Innerhalb der Klasse benötigen Sie eine private Membervariable eines zweidimensionalen Arrays vom Typ **JButton**, welche die grafische Repräsentation der Lichter übernehmen wird. Zusätzlich implementieren Sie noch folgenden Konstruktor und folgende Methode:

- *View(Controller c, int size)*, um dort alle benötigten grafischen Komponenten zu initialisieren. Dazu erzeugen Sie zunächst ein **JPanel**, dass Sie mit einem **GridLayout** der Größe size in x und y-Richtung initialisieren. In diesem Panel werden die Buttons beim Einfügen gitterförmig angeordnet. Als nächstes instanziieren Sie das zweidimensionale **JButton** Array ebenfalls mit der Größe size in beiden Dimensionen. Anschließend durchlaufen Sie das Array und erstellen für jede Stelle im Array einen neuen **JButton**, dessen anfängliche Farbe Sie auf Schwarz (*setBackground(Color.BLACK)*) und Größe auf 50 Pixel (*setPreferredSize(new Dimension(50,50)*) setzen. Diesen Button fügen Sie in Ihr Array an der aktuellen Stelle ein und Ihrem Panel hinzu (*add(button)*). Zuletzt muss der Button noch beim **Controller** registriert werden. Hierzu rufen Sie auf dem **Controller** die Methode *registerButton* mit dem Parameter des Buttons, sowie der aktuellen Position des Arrays (x und y) auf. Diese Methode werden Sie gleich noch in der Klasse **Controller** implementieren.
  Zuletzt machen Sie ihre **View** noch anzeigebereit (denken Sie daran, **View** ist auch ein **JFrame**), indem Sie einen geeigneten Titel setzen (*setTitle(text)*), das Inhaltspanel auf Ihr gerade mit Buttons bestücktes Panel festlegen (*setContentPane(panel)*), die Standard-Schließungsoperation setzen (*setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)*), das Fenster richtig zusammensetzen lassen (*pack()*) und es anschließend sichtbar machen (*setVisible(true)*).

- *void toggleButton(int x, int y)*, um die Farbe eines Buttons an der gegebenen Stelle zu wechseln. Prüfen Sie dazu die aktuelle Farbe und setzen dann davon abhängig die neue Farbe. Achten Sie auch hier auf die Gültigkeit der Parameter, um Exceptions zu verhindern.

Wechseln Sie als nächstes zur Klasse **Controller**, welche die Interaktion zwischen der Ansicht und dem Datenmodell steuert. Diese Klasse muss sowohl eine Referenz auf das **Model**, als auch auf die **View** speichern (→ private Member). Übergeben Sie im Konstruktor nur den Parameter vom Typ **Model** und setzen Ihn dort (warum, werden Sie spätestens in der main-Methode sehen) und ergänzen noch folgende Methoden:

- *void setView(View v)*, um die übergebene **View** zu speichern. Da eine wechselseitige Abhängigkeit zwischen **View** und **Controller** besteht, kann eine der Referenzen erst gesetzt werden, nachdem die andere Instanz erstellt wurde. Dazu wird entsprechend diese set-Methode benötigt.

- *void registerButton(JButton b, int x, int y)*, um dort die Abläufe zu implementieren, die beim Anklicken des Buttons an der Position x/y geschehen sollen. Rufen Sie hierzu auf dem Button die Methode *addActionListener(ActionListener listener)* auf und übergeben der Methode eine anonyme Klasse oder noch eleganter einen Lambda-Ausdruck als Parameter (**ActionListener** ist ein funktionales Interface). Innerhalb dieses Blocks soll Folgendes passieren:

  - In der Datenrepräsentation des **Model**s soll der Zustand des Arrays an der Stelle x/y geändert werden, ebenso wie bei allen direkt angrenzenden Feldern.

  - In der grafischen Repräsentation der **View** soll die Farbe des Buttons an der Stelle x/y geändert werden, ebenso wie bei allen direkt angrenzenden Feldern.

  - Nach Durchführung der vorherigen Schritte sollen Sie abschließend prüfen, ob das Rätsel gelöst wurde und im positiven Fall eine Benachrichtigung ausgeben (**JOptionPane**.*showMessageDialog(view, „Glückwunsch, gewonnen!“)*)

Nach einer erfolgreichen Implementierung der vorherigen Schritte dürfte Ihr Spiel nun bereit sein, gespielt zu werden. Zum Abschluss schreiben Sie in der Klasse **Main** die main-Methode, in der Sie zuerst ein **Model** mit der Größe 5 instanziieren, dann einen **Controller** mit dem **Model** als Parameter und schließlich eine **View**, mit dem **Controller**, sowie wieder der Größe 5 als Parameter. Zuletzt rufen Sie noch die Methode *setView* auf dem **Controller** auf und übergeben dieser die **View**, damit alle Referenzen korrekt gesetzt sind. Jetzt können Sie Ihr Spiel ausführen und losrätseln. Viel Spaß! ☺