Übung 09: Swing, MVC

Abgabetermin: 28. 5. 2015, 8:15

Name: _____ **Matrikelnummer**: _____

Informatik: ☐ G1 (Prähofer) ☐ G2 (Prähofer) ☐ G3 (Grimmer) ☐ G4 (Grimmer)

WIN: ☐ G1 (Khalil) ☐ G2 (Kusel) ☐ G3 (Kusel)

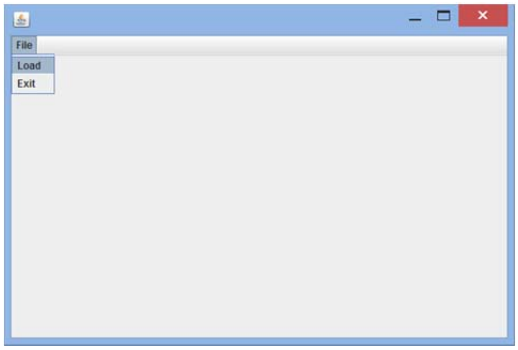
Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 9	24	Java-Programm, Ausgabe eines Spielablaufs	Java-Programm	☐	

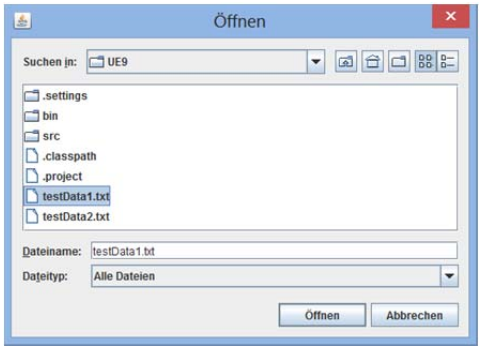
RememBrain GUI nach dem MVC Prinzip

Nachdem in Übung 8 die Spielelogik sowie eine rein textuelle Benutzerschnittstelle für *RememBrain* entwickelt wurden, soll in dieser Übung eine graphische Benutzerschnittstelle mit Swing nach dem MVC-Prinzip implementiert werden.

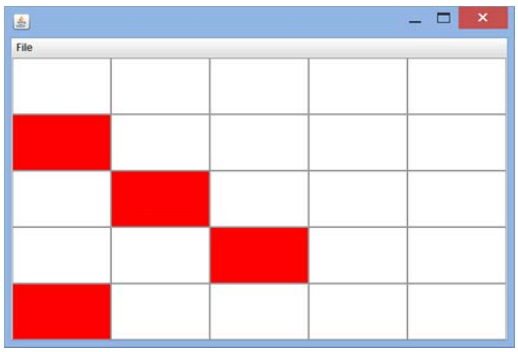
Die Applikation soll folgendes leisten:

- Nach dem Starten der Applikation kann der Benutzer über einen Menüeintrag `"Load"` einen gültigen Initialzustand aus einer Datei laden.

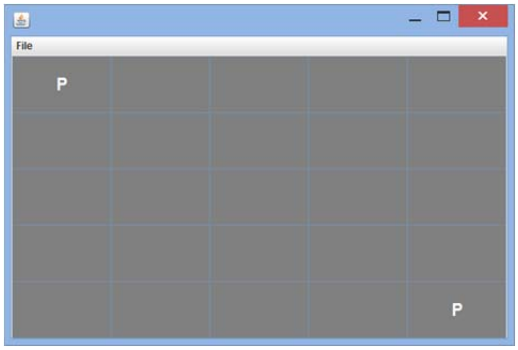




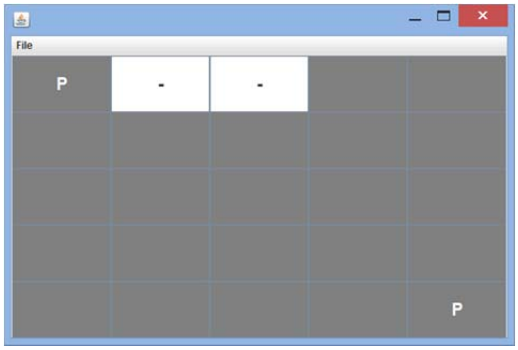
- Nach dem Laden des Initialzustands startet das Spiel mit der ersten Spielphase, d.h., dem Einprägen der Bomben.



- Nach Ablauf der Einprägezeit, startet die zweite Spielphase, d.h., die Bomben werden verdeckt und die zu verbindenden Punkte werden angezeigt → d.h., hier muss sich die Anzeige ändern.



- Fortan kann der Spieler den Pfad zwischen den Punkten erstellen.



- Schließlich soll nach dem Erstellen eines Pfades das Ergebnis des Spiels entsprechend visualisiert werden.

- Darüber hinaus kann das Spiel jederzeit über den Menüeintrag `"Exit"` beendet werden.

Hinweise:

- Verwenden Sie für die Übung u. a. folgende Komponenten und Features von Swing:
  - `JFrame`
  - `JMenu, JMenuBar, JMenuItem`
  - `JPanel, JButton`
  - `BorderLayout, GridLayout`
  - `JOptionPane`
  - `JFileChooser`

- Die Zellen im Spielfeld können als `JButtons` realisiert werden. Das heißt, jede Zelle ist ein eigener Button, der, wenn er geklickt wird, die entsprechende Zelle auswählt. Dadurch wird die Anzeige entsprechend geändert. Hierfür stehen eine Reihe von Methoden zur Verfügung, z.B.:
  - `button.setBackground(Color.RED);` //setzt die Hintergrundfarbe eines Buttons auf Rot
  - `button.setForeground(Color.WHITE);` //setzt die Textfarbe eines Buttons auf Weiß
  - `button.setText("some text");` //setzt einen Text, der auf dem Button angezeigt wird
  - `button.setFont(new Font("Arial", Font.BOLD, 20));` //setzt eine Schriftart für den Button
  - Beachten Sie weiters, dass in der ersten Spielphase die Buttons deaktiviert werden müssen. Dies kann mittels – `button.setEnabled(false);` – erreicht werden.
- Bei einem Button-Klick muss man wissen, auf welcher Position sich die ausgewählte Zelle befindet. Das heißt, Sie brauchen die Information, welcher Zelle im Spielfeld welcher Button entspricht. Das kann man z.B. erreichen, indem man eine Ableitung der Klasse `JButton` realisiert, die die Zellkoordinaten speichert.
- Bitte beachten Sie, dass wenn sich die Komponenten eines GUI-Elements ändern (z.B. Buttons zu einem Panel hinzugefügt werden), die Komponente darüber informiert werden muss, damit sie z.B. neu gezeichnet wird → dies kann durch einen Aufruf von `validate()` erreicht werden.

- Beachten Sie ferner, dass ein direkter Aufruf der Methode `field.timeToRemember(int millis)` durch den enthaltenen Aufruf von `Thread.sleep(int millis)` die GUI zum „Einfrieren“ bringen würde und damit ein unerwünschtes Verhalten erzeugen würde. Um dies zu umgehen, verwenden Sie bitte die Klasse `javax.swing.Timer`, welche für die korrekte Realisierung von Animationen existiert.

```
Timer t = new Timer(0, new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                      field.timeToRemember(...);
                }
          });
          t.setRepeats(false);
          t.start();
```

- Realisieren Sie schließlich die GUI nach dem *MVC-Prinzip*
  - Model-Implementierung:
    Realisieren Sie den Modellteil für *RememBrain* – d.h. erweitern Sie Ihr Modell um einen `FieldChangedListener` und ein `FieldChangedEvent`, sodass sich andere Klassen beim Modell registrieren können und Ereignisse empfangen können.
  - View-Implementierung:
    Konstruieren Sie die View, sodass die View auf Änderungen im Modell horcht und sich entsprechend anpasst, d.h., implementieren Sie einen entsprechenden `FieldChangedListener` und registrieren Sie diesen beim Modell. Folglich dürfen Änderungen in der Anzeige ausschließlich durch Ereignisse des Modells veranlasst werden.
  - Controller-Implementierung:
    Implementieren Sie schließlich einen Controller, der auf die Button-Klicks horcht (Implementierung von `ActionListener`) und die Zellen entsprechend auswählt, d.h., das Modell entsprechend manipuliert.