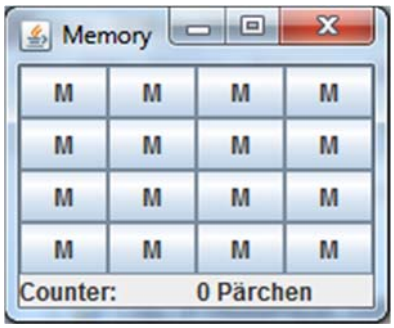
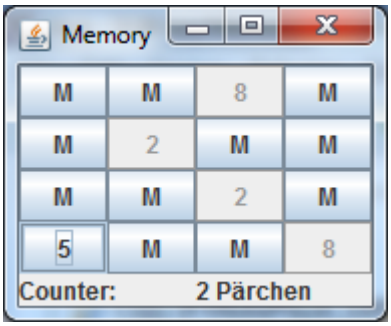
Übungsaufgabe zu GUI- und Thread-Programmierung

Memory ist ein Spiel zum Gedächtnistraining, bei dem jeweils Pärchen (hier Zahlenpärchen) verdeckt in einem Spielfeld liegen (die Rückseite zeigt ein „M“). Man klickt zunächst auf ein Feld und sieht dann den Zahlenwert. Anschließend klickt man auf ein weiteres Feld. Stimmt das Pärchen überein, bekommt man einen Punkt, die Felder werden ausgegraut. Falls nicht, werden die Karten nach 2 Sekunden wieder automatisch umgedreht. Dieses Spiel soll nun von Ihnen in einer vereinfachten Version mit Hilfe von Java umgesetzt werden.

Beispiel 1 für Teilaufgabe a)
Ein ganz neues Spielfeld; die einzelnen Spielfelder (Klasse `MemoryFeld`) zeigen ein „M“ als „Rückseite“



Beispiel 2 für Teilaufgabe b)
Zwei Pärchen wurden bereits aufgedeckt und die Felder ausgegraut; das Feld mit der „5“ wurde zuletzt angeklickt, der nächste Klick entscheidet, ob sie gleich sind oder wieder umgedreht werden



Hierzu ist die Klasse **`MemoryFeld`** und ein Auszug der Klasse **`Memory`** vorgegeben:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class MemoryFeld extends JButton{
    private int wert;

    MemoryFeld(int wert){
        setText("M"); // ein "M" symbolisiert die Rückseite
        this.wert=wert;
    }

    int getWert(){
        return wert;
    }
}

class Memory extends JFrame implements ActionListener{

    private MemoryFeld letzterKlick = null;
    private JLabel counter;
    private int paerchen = 0;

    public static void main(String args[]){
        int felder[][] = { // ein Feld sei immer 4 x 4 groß
            {1, 4, 8, 3},
            {5, 2, 7, 4},
            {3, 6, 2, 6},
            {5, 7, 1, 8}
        };
        new Memory(felder);
    }
    // ...
}
```

Teilaufgabe a)

Ergänzen Sie die Klasse `Memory` um einen Konstruktor, der ein Spielfeld mit fest vorgegebener Größe von 4 x 4 `MemoryFeld`-Objekten und der Anzeige des Pärchen-Counters, wie oben im Beispiel angezeigt, erstellt. Die Zahlenwerte für die Felder werden übergeben (siehe `main`-Methode). Jedem `MemoryFeld` ist die aktuelle Objektinstanz als ActionListener zuzuweisen (Implementierung des `ActionListener` siehe nächste Teilaufgabe).

Teilaufgabe b)

Die Klasse **`Memory`** implementiert das Interface **`ActionListener`**. Definieren Sie hierzu die notwendige Methode **`actionPerformed`**.

Für das zeitversetzte wieder Verdecken zweier Felder mit ungleichen Zahlen ist die unten angegebene Klasse **`Umdreher`** zuständig. Verwenden Sie diese für das wieder Umdrehen der Felder in der Methode **`actionPerformed`**.

```
class Umdreher extends Thread {
    private MemoryFeld f1, f2;

    Umdreher(MemoryFeld f1, MemoryFeld f2){
        this.f1=f1; this.f2=f2;
    }

    public void run(){
        try {
            Thread.sleep(2000);
        } catch (InterruptedException ex) { /* ignorieren */ }
        f1.setText("M");
        f2.setText("M");
    }
}
```

Tipps:

- Nutzen Sie die Attribute `letzterKlick`, `counter` und `paerchen` aus der Klasse `Memory` (siehe oben) für diese Aufgabe
- Bei einem Klick auf jeden Fall mal den Zahlenwert aufdecken (dafür brauchen Sie den Umdreher nicht)
- Das „Ausgrauen“ der Felder geschieht mit Hilfe der Methode `public void setEnabled(boolean b)`, die `MemoryFeld` anbietet (diese Felder reagieren dann automatisch nicht mehr auf Klicks)
- Wurde ein zweites Feld angeklickt und ist es nicht gleich, dann den Umdreher-Thread starten (Parameter im Konstruktor beachten!)
  (Sie können davon ausgehen, dass in den 2 Sekunden Wartezeit nie geklickt wird!)