Bernhard Dietrich


```
//
// Bernhard Dietrich (6256800), Lars Fernhomberg (6256030), Sebastian
Kniesburges (6257120) Gruppe 20
//
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

// Ein einfaches Memory-Spiel
// Autoren Falke, Kastens, bd

public class Memory extends Applet{
    private int s=5;
    private int z=5;
    public int labelsLength;
    public void init(){ // Konstruktor
        if (getParameter("s")!=null)
            s=Integer.parseInt(getParameter("s"));
        if (getParameter("z")!=null)
            z=Integer.parseInt(getParameter("z"));
        labelsLength = z*s;
        setSize(300,400);
        setLayout(new GridLayout(s,z));

        // Nur ein Listener-Objekt fuer alle Felder:
        MemoryListener memListener = new MemoryListener();

        // Buttons fuer die Felder erzeugen:
        CharButton b;
        int a =labelsLength;
        if(a%2!=0)
            labelsLength--;
        int zahlen[]=randomizedArray(labelsLength);
        for (int i=0;i<zahlen.length;i++) {
            b = new CharButton(String.valueOf(zahlen[i]));
            add(b); b.addActionListener(memListener);
            b.hide();
        }
    }

    // Innere Hilfsklasse:
    // Button-Unterklasse mit verdeckbarer Beschriftung:
    static class CharButton extends Button {
        private String myLabel;
        CharButton(String lab) {
            super(lab);
            myLabel = lab;
        }
        public void show() { setLabel(myLabel); }
        public void hide() { setLabel(""); }
        // ueberschreibt getLabel () aus Button:
        public String getLabel() { return myLabel; }
    }

    // Die bisher aufgedeckten Felder:
    private CharButton firstSelection = null;
    private CharButton secondSelection = null;

    // Innere Klasse fuer den eingebetteten Listener:
    class MemoryListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            // Das gerade gewaehlte Feld:
            CharButton flipped = (CharButton) e.getSource();
```

```
            // Drei Zustaende:
            //   kein, ein, zwei Feld(er) ist/sind aufgedeckt:
            if (firstSelection == null) {
                // kein Feld - nun das erste:
                firstSelection = flipped;
                flipped.setEnabled(false); flipped.show();
            }
            else if (secondSelection == null) {
                // ein Feld - nun das zweite:
                flipped.setEnabled(false); flipped.show();
                secondSelection = flipped;
                if (firstSelection.getLabel().equals(secondSelection.getLabel
())) {
                    // Die Marken sind gleich:
                    firstSelection.setBackground(Color.BLACK);
                    secondSelection.setBackground(Color.BLACK);
                    // Neue Wahl vorbereiten:
                    firstSelection = null; secondSelection = null;
                }
            }
            else {
                // Nach einem Fehlversuch wurde drittes Feld aufgedeckt:
                flipped.setEnabled(false); flipped.show();
                firstSelection.hide(); firstSelection.setEnabled(true);
                firstSelection = flipped;
                secondSelection.hide(); secondSelection.setEnabled(true);
                secondSelection = null;
            }
        }
    }
    private int[] randomizedArray(int anzahlKarten) {
        int[] zahlen = new int[anzahlKarten];
        for(int i=0; i<=zahlen.length/2;i++) {
            zahlen[i]=i;
            zahlen[anzahlKarten-1-i]=i;
        }
        for(int i=0; i<zahlen.length;i++) {
            int rnd = (int)((zahlen.length-1) * Math.random());
            int temp = zahlen[i];
            zahlen[i]=zahlen[rnd];
            zahlen[rnd] = temp;
        }
        return zahlen;
    }
}
```