Bernhard Dietrich (6256800)
Lars Fernhomberg (6256030)
Sebastian Kniesburges (6257120)

Übung Nr. 20

Freitag, 09:00-11:00 Uhr in E3.301

Aufgabe 6: Listener zuordnen

Hinweis: Einige Zeilen mussten für ein ansprechendes Druckergebnis umgebrochen werden. Dieser Umbruch wurde durch einen Unterstrich („_“) dargestellt und gehört nicht zum eigentlichen Quelltext.

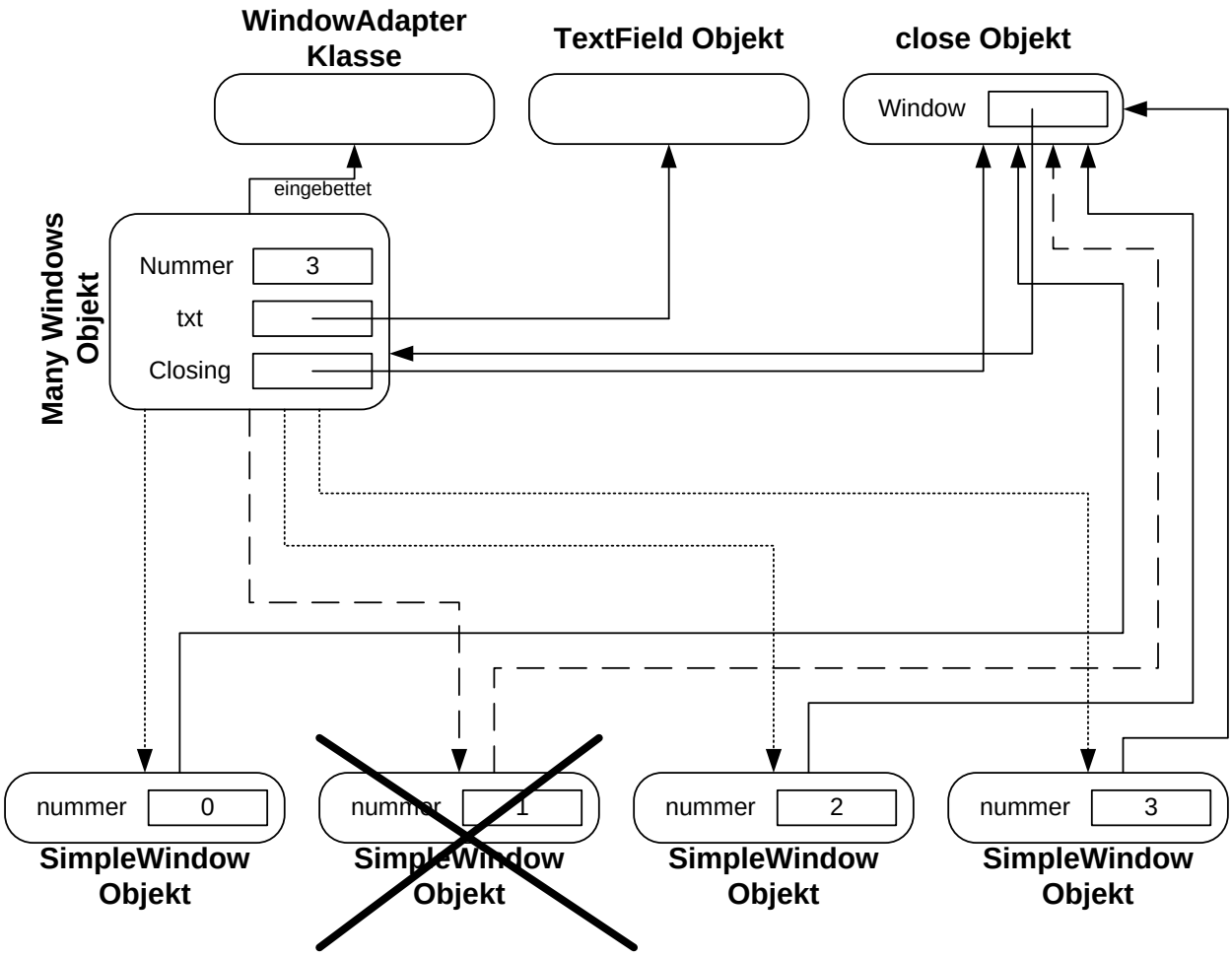
```
import java.awt.*;
import java.awt.event.*;

/**
 * SimpleWindow implementiert die anzuzeigenden Fenster, die durch einen
 * Klick auf "Neues Fenster" des Hauptformulars sichtbar werden.
 */
class SimpleWindow
      extends Frame
{
      //Die Nummer des Fensters als lokale Variable
      private int nummer =0;

      /**
       * Gibt die Nummer des Fensters zurück
       * @return int Nummer des Fensters
       */
      public int getNummer()
      {
            return nummer;
      }

      /**
       * Konstruktur der Klasse SimpleWindow
       * @param nummer int Die Nummer des Fensters
       * @param listener WindowAdapter Der WindowAdapter, der das Fenster
       * überwachen soll
       */
      public SimpleWindow(int nummer, WindowAdapter listener)
      {
            super();    //Initalisierung der Oberklasse
            this.nummer = nummer;   //Festlegen der eigenen Nummer
            this.setTitle("Fenster " + nummer); //Fenstertitel festlegen
            this.setSize(getRandomSize());     //Zufällige Größe setzen
            this.setLocation(getRandomLocation()); //Zufallspos. setzen
            this.addWindowListener(listener); //Listener „anschließen“
            this.setVisible(true);  //Frame anzeigen
      }

      /**
       * Erzeugt eine zufällige Größe für das Fenster, welche sich
       * zwischen 0 und der maximalen Höhe bzw. Breite des Bildschirms
       * bewegt.
       * @return Dimension Eine zufällige Größe für das Fenster
       */
      private Dimension getRandomSize()
      {
```

```
            //Auflösung des Bildschirms feststellen
            Dimension ScreenSize = Toolkit.getDefaultToolkit()_
                  .getScreenSize();
            Dimension d = new Dimension();
            //Höhe und Breite zufällig erzeugen
            d.height = (int)((ScreenSize.height + 1) * Math.random());
            d.width = (int)((ScreenSize.width + 1) * Math.random());
            return d;
      }

      /**
       * Gibt einen zufällige Position innerhalb des Bildschirms zurück,
       * an der das Fenster dargestellt werden kann. Das Fenster ist
       * dabei immer komplett innerhalb des Bildschirms und verlässt ihn
       * nicht.
       * @return Point Punkt der neuen Position für das Fenster
       */
      private Point getRandomLocation()
      {
            //Auflösung des Bildschirms feststellen
            Dimension ScreenSize = Toolkit.getDefaultToolkit()._
                  getScreenSize();
            Dimension WindowSize = this.getSize();
            //X- und Y-Koordinaten zufällig erzeugen
            int x = (int)((ScreenSize.width - WindowSize.width + 1)_
                  * Math.random());
            int y = (int)((ScreenSize.height - WindowSize.height + 1)_
                  * Math.random());
            return new Point(x,y);
      }
}

/**
 * Stellt die Implementation eines WindowAdapter zur Verfügung, welche
 * für das Schließen der einzelnen SimpleForms zuständig ist und darüber
 * hinaus auch die entsprechenden Ausgaben anweist.
 */
class close
      extends WindowAdapter
{
      private Windows Window; //Hauptfenster mit Statustextfeld
      public close(Windows window)  //Konstruktur des ganzen
      {
            this.Window = window;
      }

      public void windowClosing(WindowEvent e)
      {
            //Zu schließendes Formular bestimmen (Source des Events)
            SimpleWindow form = (SimpleWindow)e.getSource();
            //Statusmeldung ausgeben
            Window.setText("Fenster " + form.getNummer()_
                  + " geschlossen.");
            form.dispose();//Fenster schließen
      }
}
```

```
/**
 * Die Hauptklasse des Programms, welches ein Fenster implementiert, in
 * welchem die Statusmeldungen zu sehen sind und mit dem die neuen
 * Fenster erzeugt werden können.
 */
public class Windows extends Frame
{
	private TextField txt = new TextField(); //Textfeld für die
                                                 //Ausgaben
	private int nummer = -1;	//Counter für die Fensternummern
	private close Closing = new close(this); //WindowAdapter für das
                                                 //Schließen der Fenster
	public Windows()
	{
		super(“Many Windows”);
		this.setLayout(new BorderLayout()); //LayoutManager setzen
		this.setSize(370,80);   //Größe setzen
		txt.setEnabled(false);  //Textfeld für Zugriffe sperren
		this.add(txt, BorderLayout.NORTH);  //Textfeld einfügen
		//WindowListener nur für dieses Fenster, der die Anwendung
		//terminiert
		this.addWindowListener(new WindowAdapter()
			{
				public void windowClosing(WindowEvent e)
				{
					System.exit(0);
				}
			}
		);
		Button btnNew = new Button("Neues Fenster");
		//Neue Fenster erzeugen, bei Klick auf den Button
		btnNew.addActionListener(new ActionListener()
			{
				public void actionPerformed(ActionEvent e)
				{
					nummer++;
					SimpleWindow window = new SimpleWindow_
						(nummer, Closing);
					Windows.this.setText("Fenster " + nummer_
						+ " geöffnet.");
				}
			});
		this.add(new Panel(), BorderLayout.CENTER); //Platzhalter
		this.add(btnNew, BorderLayout.SOUTH); //Button hinzufügen

		this.setVisible(true); //Frame anzeigen
	}

	/**
	 * Gibt den aktuellen Text zurück, welcher in dem für Ausgaben
	 * vorgesehen Textfeld steht.
	 * @return String Letzte Textmeldung
	 */
	public String getText()
```

```
	{
		return this.txt.getText();
	}

	/**
	 * Legt eine neue Textmeldung für die Anzeige im dafür vorgesehenem
	 * Textfeld fest.
	 * @param Text String Textmeldung, die angezeigt werden soll
	 */
	public void setText(String Text)
	{
		this.txt.setText(Text);
	}

	/**
	 * Einsprungspunkt der Applikation
	 */
	public static void main(String[] args)
	{
		Frame window = new Windows();
	}
}
```

Aufgabe 7: Zusammenhang der Objekte



Das „Many Windows“ Objekt besitzt eine eingebettete Klasse, welche sich vom WindowAdapter ableitet und welche für die Verarbeitung des „Schließen“-Buttons der Form verantwortlich ist. Da diese Klasse lediglich vom „Many Windows“ Objekt benötigt wird, bietet es sich an, die enstprechende Klasse als annonyme Klasse in dieser Oberklasse zu implementieren.
Die „SimpleWindow“ Objekte werden von dem „Many Windows“ Objekt erzeugt, werden dort allerdings nicht als Variable gespeichert, so dass dort keine Beziehung besteht und das „Many Windows“ Objekt somit auch nicht auf die einzelnen Objekte zugreifen kann. Dieser Sachverhalt wird durch die gepunketeten Linien verdeutlicht.
Das „SimpleWindow“ mit der Nummer 1 ist gemäß Aufgabenstellung bereits beendet worden und existiert somit nicht mehr als ansprechbares Objekt im Speicher. Um zu Zeigen, wie die Verknüpfung während der Existenz dieses Objekts vorhanden waren, wurde das betroffene Objekt hinzugefügt. Die visuellen Markierungen sollen aber die Terminierung des Objekts verdeutlichen.
Gemäß der Aufgabenstellung existiert lediglich ein „WindowListener“ („close“ Objekt) für die „SimpleWindow“-Objekte. Die „SimpleWindow“ Objekte bekommen im Konstruktor einen Verweis auf dieses Objekt, welches bereits im Konstruktor als neuer WindowListener hinzugefügt wird (`this.addWindowListener(close)`).
Das bereits angesprochene „close“ Objekt bekommt eine Referenz auf das „Many Window“ Objekt durch den Konstruktor übergeben, damit die Statusänderungen im entsprechenden Textfeld „txt“ problemlos möglich sind. Ohne diese Referenz könnte das „close“ Objekt nicht auf das Textfeld zugreifen.