Aufgabe 1: Verwendung des Debuggers

Bearbeiten Sie die Aufgaben 3.43 und 3.44 auf Seite 85.

3.43:

Nein, keine neue Erkenntnisse.

3.44:

done.

Aufgabe 2: Weitere Aufgaben aus dem Buch

Bearbeiten Sie in kleinen Gruppen folgende Aufgaben aus dem Buch.
Falls Sie nicht sicher sind, dass Ihre erarbeitete Lösung richtig ist, notieren Sie sich die Frage und Ihre möglichen Antworten. Anschliessend überprüfen Sie Ihre Antworten innerhalb des gesamten Lernteams. Bei Unstimmigkeiten bereiten Sie die Frage mit allen möglichen Antworten (inkl. Begründungen) für das Lernteam-Coaching vor.

Seite 65: Aufgaben 3.9 - 3.12

3.9:

false
true
false
false
true

3.10:

(a && b) || (!a && !b) oder *(a && b) || ! (a || b)*

3.11:

a^b
(^ = XOR)

3.12:

! (a || b)

Seite 66: Aufgaben 3.13 und 3.14

3.13:

Es wird davon ausgegangen das es nur Zahlen grösser 0 sind. Weiter wird davon ausgegangen, dass es nur zweistellige Zahlen sind, andernfalls müsste z.B. für dreistellige Zahlen bei Zahlen kleiner 10 zwei führende 0 eingefügt werden.

3.14:

Wenn ein Whitespace dazwischen ist macht es einen Unterschied, andernfalls nicht.

Seite 67: Aufgaben 3.18 und 3.20

3.18:

0, 1, 2, 3 oder 4

3.20:

Zuerst wird zum aktuellen Wert 1 dazugezählt. Danach mittels dem Modulo Operator den „Überlauf“ überprüft.
Bsp: limit = 60; value = 59;
59 % 60 = 59
60 % 60 = 0
(61 % 60 = 1) etc...

Seite 67: Aufgaben 3.24 und 3.25

3.24:

60 mal, oder setTime(1, 0); aufrufen.

3.25:

```
NumberDisplay nd = new NumberDisplay(80);

nd.getValue()
0   (int)

nd.setValue(79);

nd.getValue()
79   (int)

nd.increment();
nd.increment();
nd.getValue()
1   (int)
```

Seite 68: Aufgaben 3.26 und 3.27

3.26:

Editor(String fileName, int newInt)

3.27:

private Rectangle window;
window = new Rectangle(5, 15);

Aufgabe 3: Herausfordernde Aufgaben aus dem Buch

Bearbeiten Sie die Aufgaben 3.31 und 3.32 auf Seite 76.
Erstellen Sie dabei die zwei unterschiedlichen Implementationen für Aufgabe 3.32 und vergleichen Sie diese anschliessend.

3.31:

```
public class NumberDisplay
{
    private int limit;
    private int value;
    private int minvalue;

    /**
     * Constructor for objects of class NumberDisplay.
     * Set the limit at which the display rolls over.
     */
    public NumberDisplay(int rollOverLimit, int initMinValue)
    {
        limit = rollOverLimit;
        value = initMinValue;
        minvalue = initMinValue;
    }

    /**
     * Increment the display value by one, rolling over to zero if the
     * limit is reached.
     */
    public void increment()
    {
        if ((value + 1) >= limit) {
            value = minvalue;
        }
        else {
            value = value + 1;;
        }
    }

[...]
}

public class ClockDisplay
{
    private NumberDisplay hours;
    private NumberDisplay minutes;
    private String displayString;    // simulates the actual display

    /**
     * Constructor for ClockDisplay objects. This constructor
     * creates a new clock set at 00:00.
     */
    public ClockDisplay()
    {
        hours = new NumberDisplay(13,1);
        minutes = new NumberDisplay(60,0);
        updateDisplay();
    }

    /**
     * Constructor for ClockDisplay objects. This constructor
     * creates a new clock set at the time specified by the
     * parameters.
     */
    public ClockDisplay(int hour, int minute)
    {
        hours = new NumberDisplay(13,1);
        minutes = new NumberDisplay(60,0);
        setTime(hour, minute);
    }
[...]
}
```

3.32:

```
/**
 * Update the internal string that represents the display.
 */
private void updateDisplay()
{
    int currentHour;
    int displayHours;

    currentHour = hours.getValue();

    // displayString = hours.getDisplayValue() + ":" + minutes.getDisplayValue();
    if (currentHour >= 12) {
        //pm
        if (currentHour == 12) {
            displayHours = currentHour;
        }
        else {
            displayHours = currentHour - 12;
        }
        displayString = displayHours + ":" + minutes.getDisplayValue() + "pm";
    }
    else {
        //am
        displayString = currentHour + ":" + minutes.getDisplayValue() + "am";
    }
}
```

Aufgabe 4: Programmieraufgaben (optional)

Bearbeiten Sie die Aufgaben 3.45 und 3.46 auf Seite 85.

3.45:

```
public class MailItem
{
    // The subject of the message.
    private String subject;

    public MailItem(String from, String to, String subject, String message)
    {
        this.from = from;
        this.to = to;
        this.subject = subject;
        this.message = message;
    }

    /**
     * @return The subject of the message.
     */
    public String getSubject()
    {
        return subject;
    }


    /**
     * Print this mail message to the text terminal.
     */
    public void print()
    {
        System.out.println("From: " + from);
        System.out.println("To: " + to);
        System.out.println("Subject: " + subject);
        System.out.println("Message: " + message);
    }
}

    public void sendMailItem(String to, String subject, String message)
    {
        MailItem item = new MailItem(user, to, subject, message);
        server.post(item);
    }
```

3.46:

```
private Screen myScreen1;
myScreen1 = new Screen(1024, 768);

if (myScreen1.numberOfPixels() > 2000000) {
  myScreen1.clear(false);
}
```