Lösungen

AUFGABE 1: REGULÄRE AUSDRÜCKE (5 PUNKTE)

AUFGABE 1.1 (3 PUNKTE)

Lösung [3 Punkte total - jeweils 1 Punkt für gute Begründung der korrekten Ausdrücke, 1 Punkt für die mind. 1 korrektes Gegenbeispiel pro falschem Ausdruck]

Regulärer Ausdruck	Kurze Begründung, warum der Ausdruck korrekt ist **oder** jeweils zwei Gegenbeispiele, die zeigen, warum der Ausdruck nicht korrekt ist
[012][0-9]:[0-5][0-9]	25:00, 29:59
([01][0-9]|2[0-3]):([0-5][0-9])	Der Teil vor dem Doppelpunkt beschreibt *alle* Stunden 00..19 bzw. 20..23 (nicht mehr, nicht weniger). Der Teil nach dem Doppelpunkt beschreibt *alle* Minuten 00..59.
\d\d:\d\d	34:56, 78:89
([01][0-4]|[01][5-9]|2[0-3]):([0-5][0-9])	Der Teil vor dem Doppelpunkt beschreibt *alle* Stunden 00..19 bzw. 20..23. Der Teil nach dem Doppelpunkt beschreibt *alle* Minuten 00..59.

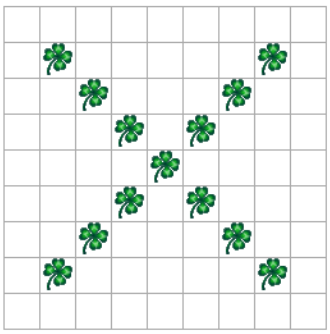
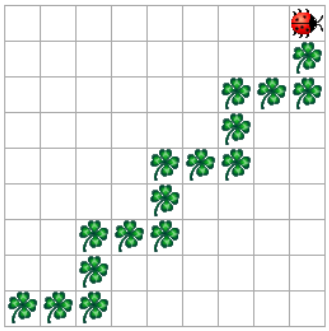
AUFGABE 1.2 (2 PUNKTE):

Mögliche Lösungen [2 Punkte - 1 Punkt für Muster ohne Berücksichtigung der Anzahl Wiederholungen, 1 Punkt für die korrekte Anzahl Wiederholungen]

[A-Z]{1,2}\d{2} \d[A-Z]{2} oder [A-Z][A-Z]?\d\d \d[A-Z][A-Z]

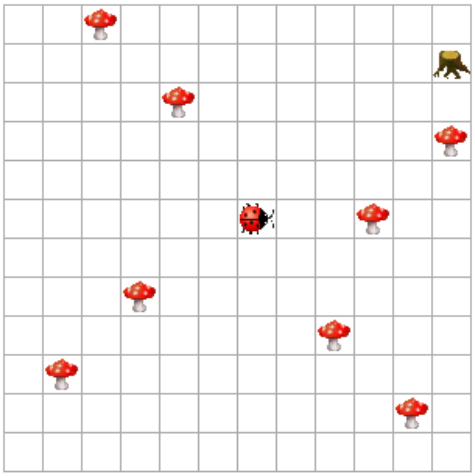
AUFGABE 2: PROGRAMM LESEN – PROGRAMM ERGÄNZEN UND LESEN (5 PUNKTE)

AUFGABE 2.1 (3 PUNKTE)

	1 Punkt, wenn mindestens 1 Diagonale korrekt ist.
	2 Punkte, wenn Treppe vollständig korrekt ist; 1 Punkt, wenn mindestens 1 Stufe korrekt ist.

AUFGABE 2.2 (2 PUNKTE)

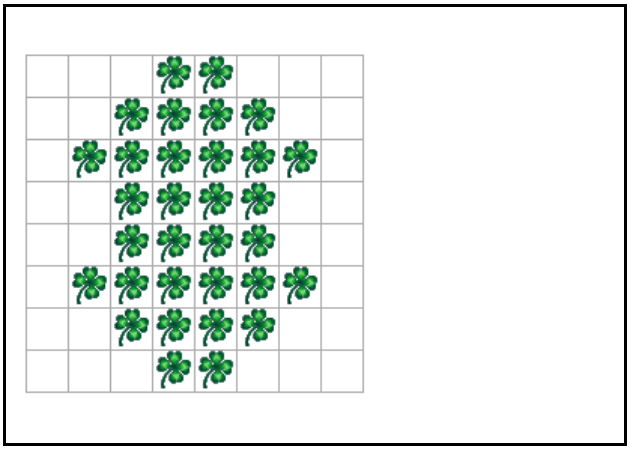
1 Punkt, wenn mindestens die ersten 2 Pilze korrekt; 2 Punkte, wenn mehr Pilze korrekt.



AUFGABE 3: JAVAKARA-PROGRAMM LESEN (5 PUNKTE)

AUFGABE 3.1 (2 PUNKTE)

1 Punkte, wenn eine Hälfte korrekt; zwei Punkte, wenn ganzes Bild korrekt.



AUFGABE 3.2 (2 PUNKTE)

Mögliche Lösung [2 Punkte - je einen Punkt]

legeBlaetter(...)	Die Methode legt in Zeile y Kleeblätter: *anzahl* Kleeblätter links der Mitte, *anzahl* Kleeblätter rechts der Mitte.
myMainProgram()	Das Hauptprogramm erfragt vom Benutzer eine Zahl. Es legt dann links und rechts von der vertikalen Mitte in Zeilen von oben und von unten her Kleeblätter. Das Prozedere wird so oft wiederholt, bis jede Zeile der Welt befüllt ist.

AUFGABE 2.3 (1 PUNKT)

Mögliche Lösung [1 Punkt]

Die Welt, die das Programm zeichnet, ist vertikal symmetrisch: Von oben und von unter her wird jeweils das gleiche Muster in die Zeilen gelegt.

AUFGABE 4: JAVAKARA-PROGRAMM SCHREIBEN (5 PUNKTE)

AUFGABE 4.1 (4 PUNKTE)

Mögliche Lösung [4 Punkte - 1 Punkt für while-Schleife, 1 Punkt für if-else - 1 Punkt für alle Kleeblätter minus 1 aufnehmen, 1 Punkt für alle Kleeblätter - auch 2 Punkte vergeben, wenn Laufen nicht korrekt implementiert ist, aber daran gedacht wurde, alle Kleeblätter aufzunehmen]

```
public void myMainProgram() {
        while (!kara.treeFront()) {
                entferneKleeblatt();
                if (kara.mushroomFront()) {
                        laufeUmPilz();
                } else {
                        kara.move();
                }
        }
        entferneKleeblatt();
}
```

AUFGABE 4.2 (1 PUNKT)

Mögliche Lösung, stark anhängig von Lösung der ersten Teilaufgabe [1 Punkt]

Die Methode laufeUmPilz müsste angepasst werden. Sie müsste so erweitert werden, dass Kara nicht einfach um einen Pilz läuft, sondern neben den Pilzen herläuft.

AUFGABE 5: PROGRAMMIEREN ERKLÄREN (5 PUNKTE)

3 Punkte für Inhalt – jeweils ein Punkt für if-else, while, und Boole'scher Ausdruck
2 Punkte für Verständlichkeit, Beispiele – jeweils 1 Punkt für Verzweigungen und für Wiederholungen

Erklären Sie Verzweigungen (if-else): Wie kann JavaKara Entscheidungen treffen?

„Wir haben diese Software mit dem Marienkäfer, den wir steuern können. Der Marienkäfer kann zum Beispiel Kleeblätter aufnehmen und ablegen. Stell' Dir vor, wir wollen ein Programm schreiben, wo der Marienkäfer ein Blatt aufnimmt, falls er auf einem drauf steht, und andernfalls eines hinlegt. Das heisst, unser Programm muss eine *Entscheidung* treffen können. In Java würden wir das wie folgt schreiben: if (kara.onLeaf()) { kara.removeLeaf(); } else { kara.putLeaf(); } Das nennt man auch Verzweigung oder Fallunterscheidung: Je nachdem macht das Programm etwas anderes.

Die *Logik* für den Entscheid kann beliebig kompliziert werden: Da können wir beliebige logische Aussagen formulieren – wenn das und das zutrifft, wenn das oder nicht das zutrifft und so weiter. Diese logischen Aussagen heissen auch Boole'sche Ausdrücke."

Erklären Sie Wiederholungen (while): Wie kann JavaKara Dinge wiederholen?

„Stell' Dir nun vor, Du willst den Käfer laufen lassen, solange er nicht links und rechts von sich einen Baum hat. Dazu müssen wir den Käfer etwas wiederholen lassen, immer wieder, bis er das Ziel erreicht hat. In Java würden wir das wie folgt schreiben: while (!(kara.treeLeft() && kara.treeRight())) { kara.move(); } Das ist eine *Schleife*, eine Wiederholung. Wichtig ist hier die Bedingung, wie lange der Käfer etwas wiederholen muss. Diese Bedingung kann wie bei den Verzweigungen beliebig kompliziert sein."