Name: *Matrikelnummer:*

Programmierung

Klausur 1 (Wintersemester 2017/18) 14. März 2018

Answer: *With Solutions*

Hinweise (*English translation below*)

- **Bitte schlagen Sie die Klausur erst auf, sobald Sie dazu aufgefordert werden.**
- Einlesezeit: 15 Minuten. Fragen zur Klausur sollten innerhalb dieser Zeit gestellt werden. Wenn Sie eine Frage haben, melden Sie sich, und es wird jemand zu Ihnen kommen. **Anschließende** Bearbeitungszeit: 120 Minuten
- Mögliche Gesamtpunktzahl: 100 Punkte (+ 10 Bonuspunkte). Die Punktzahl jeder Aufgabe entspricht etwa der vorgesehenen maximalen Bearbeitungszeit in Minuten.
- Überzeugen Sie sich davon, dass Ihre Klausur vollständig ist.
- Legen Sie Ihren Studierendenausweis sowie einen Lichtbildausweis vor sich auf den Tisch. Während der Klausur werden wir Ihre Identität kontrollieren.
- Es sind keinerlei Hilfsmittel (Taschenrechner, Skripte, Bücher, Notizen, Telefone, etc.) für diese Klausur erlaubt. Falls Sie Papier brauchen, geben Sie uns Bescheid, statt eigenes zu benutzen. Bitte schalten Sie Ihre Telefone ab und verstauen Sie Smart Watches in Ihrem Rucksack. Wenn Sie gegen diese Regeln verstoßen, riskieren Sie, von der Prüfung ausgeschlossen zu werden und durchzufallen.
- Schreiben Sie Ihre Antworten auf die Aufgabenzettel. Sie können Antworten gegebenenfalls auf der letzten leeren Seite fortsetzen, bitte weisen Sie dann entsprechend darauf hin. Sollte der Platz immer noch nicht ausreichen, fragen Sie uns nach zusätzlichem Papier.
- Sie dürfen die Fragen auf deutsch oder englisch beantworten.
- Bitte schreiben Sie auf **jedes Blatt**, das Sie abgeben, Ihren Namen und (falls vorhanden) Ihre Matrikelnummer.
- Bis 20 Minuten vor Ende der Klausur dürfen Sie vorzeitig abgeben. Wenn Sie nicht vorzeitig abgeben, warten Sie bitte an Ihrem Platz bis Ihre Klausur zusammengeheftet ist und eingesammelt wird.
- Die korrigierten Klausuren können am 19.03.2018 zwischen 14 und 17 Uhr in CAP 4, Raum 715, eingesehen werden. Die exakte Zeit für Sie hängt von Ihrer Rechnerübung ab. Bitte schauen Sie dafür in's iLearn.

English translation of instructions above — such translations are also provided for the exam problems

- **Please do not turn the page before you are told to do so.**
- Reading time: 15 minutes. Questions concerning the problems should be asked during that time. If you have a question, raise your hand, and somebody will come to you to listen to your question. **Afterwards** time for problem solving: 120 minutes.
- The total maximum score: 100 points (+ 10 bonus points). The points given for each problem roughly correspond to the assumed maximum time to work on that problem.
- Ensure that your copy of the exam is complete.
- Put a photo ID card and your student ID card in front of you. We will examine it during the exam.
- You are not allowed to use anything other than a pen to write with. In particular, you are not allowed to use the book, any printouts, or electronic devices (which includes phones and smart watches). Do not use your own paper to write on; instead, ask us for paper, we have plenty. Breaking these rules means risking a failing grade.
- Use the space provided in the assignments to write down your answers. You may continue your answers on the very last, empty page; please make a note if you do so. If you still run out of space, ask us for additional paper.
- You may answer in English or German.
- Please put your name and immatriculation number (*Matrikelnummer*), if you have one, onto **every sheet** that you hand in.
- You are allowed to hand in early until 20 minutes before the end of the exam. If you do not hand in early, please wait at your seat until your exam has been collated and collected.
- You can examine your corrected exam on March 19 2018 between 2pm and 5pm in CAP 4, room 715. The exact time you should show up at depends on which practical class you were assigned to during the semester. The course description in the iLearn system contains further details.

Aufgabe 1 (16 Punkte)

1. Was ist eine *Variable*?

   What is a *variable*?

   Answer: *A placeholder for a piece of data. A variable has a name (or identifier), a type, and a value.*

2. Was ist ein *Objekt*?

   What is an *object*?

   Answer: *An instances of a class.*

3. Was ist ein *Konstruktor*?

   What is a *constructor*?

   Answer: *A special method that creates new objects (or that initializes new objects).*

4. Was ist eine Variablendeklaration?

   What is a *variable declaration*?

   Answer: *Declaration establishes name and type of variable, possibly also initial value.*

5. Was ist *Assoziativität*?

   What is *associativity*?

   Answer: *Associativity determines the order of evaluation among operators with the same precedence.*

6. Wie würden Sie die erste Zeile einer `for`-Schleife für die folgenden Situationen schreiben?

   What `for` loop control line would you use in each of the following situations:

   a) Von 1 bis 10 zählen (jeweils einschließlich).

      Counting from 1 to 10 (inclusive). ***Answer:*** *for (int i = 1; i <= 10; i++)*

   b) Von 0 an in Fünfer-Schritten zählen, solange die Zählvariable maximal dreistellig ist.

      Counting by fives starting at 0 as long as the number has at most three digits. ***Answer:*** *for (int i = 0; i <= 1000; i += 5)*

7. Was sind *zusammengesetzte Anweisungen* (auch *Blöcke* genannt)?

   What are *compound statements* (also called *blocks*)?

   Answer: *A sequence of statements in curly braces.*

8. Was ist das *repeat-until-sentinel pattern*?

   What is the *repeat-until-sentinel pattern*?

   Answer: *If the input contains a specific sentinel, use the if and break statements to exit the loop.*

9. Wie hängen *Argumente* und *(formale) Parameter* zusammen?

   What is the relationship between *arguments* and *(formal) parameters*?

   Answer: *Formal parameters are used in declaration of method, arguments are passed in by caller of method.*

10. Was sind *lokale Variablen*?

    What are local variables?

    Answer: *Variables declared in a method.*

11. Welche Perspektiven auf das Schreiben von Klassen unterscheiden wir?

    Which perspectives on class design do we distinguish?

    Answer: *Implementor ("How does this thing work internally?") vs. Client ("How do I use this thing?").*

12. Was ist *Überladen*?

    What is *overloading*?

    Answer: *Using the same method name with different arguments.*

13. Was bedeutet das Schlüsselwort `this`?

    What does the keyword `this` mean?

    Answer: *Evaluates to a pointer to the current object.*

14. Was sind *ArrayLists*, wann sollten sie eingesetzt werden?

    What are *ArrayLists*, when should they be used?

    Answer: *Extensable arrays, should be used when size is not known in advance.*

15. Wie überprüfen wir Strings auf Gleichheit?

    How do we check strings for equality?

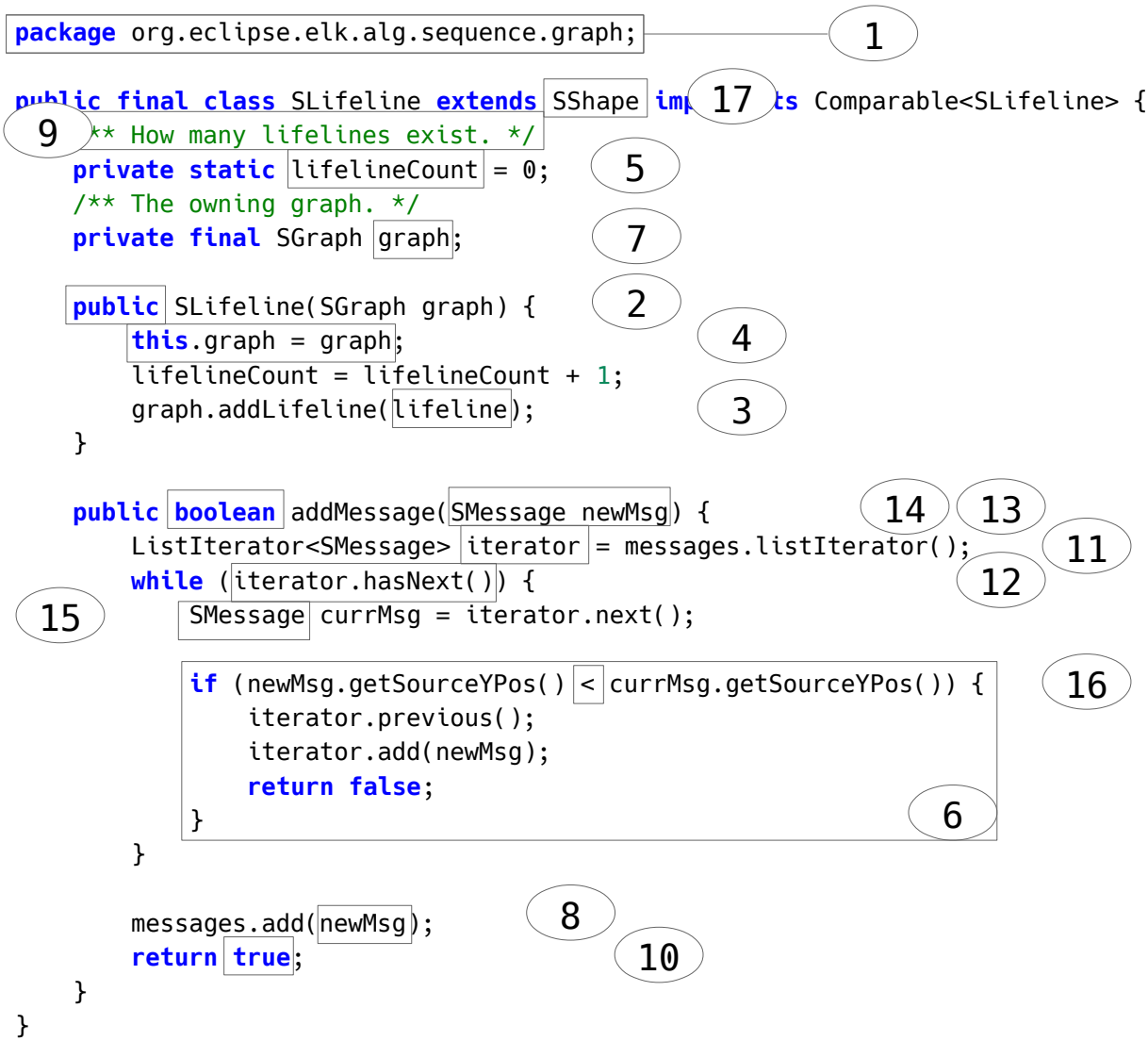
    Answer: *With equals method if string is not null! (With string literals, can also use ==, because JVM uses pool of string literals.)*

Aufgabe 2 (8 Punkte) Im Folgenden sehen Sie ein Stück Java-Code. Finden Sie für jeden der nachfolgenden Begriffe ein Vorkommen in dem Code. Markieren Sie die Stelle im Code entsprechend mit der Nummer des Begriffs (siehe 1 als Beispiel). Es reicht aus, pro Begriff **ein Vorkommen im Code zu markieren**.

Consider the following piece of Java code. For each of the listed terms, annotate an appropriate part of code with the number of the term (see term 1 as example). It is sufficient to mark **one example per term**.

1. Package declaration
2. Access modifier
3. Argument
4. Assignment
5. Class variable
6. Conditional statement
7. Constant
8. Expression
9. Javadoc comment
10. Literal
11. Local variable
12. Method call
13. Parameter
14. Return type
15. Type
16. Operator
17. Superclass

Answer:



Aufgabe 3 (8 Punkte)

1. (4 Punkte) Im Folgenden sehen Sie eine Liste von Ausdrücken. Nehmen Sie in jeder Zeile an, dass `i` und `j` den Typ **`int`** und vor Auswertung jedes Ausdrucks den Wert `0` haben.

   Schreiben Sie hinter jeden Ausdruck das Ergebnis seiner Auswertung in Java so hin, dass sowohl Wert als auch Typ des Ergebnisses ersichtlich werden. Sollte während der Berechnung ein Fehler auftreten, markieren Sie die Zeile stattdessen mit „ERROR".

   Consider the following list of expressions. Assume in each line that `i` and `j` are of the type **`int`** and their values are set to `0` before evaluating each expression.

   Compute the value of each expression as interpreted by Java and write it down such that both the value and its type are apparent. If an error occurs during an evaluation, write "ERROR" on that line.

   `5 % 10 + Math.sqrt(0)` ***Answer:*** *5.0*

   `i = 5 & (j <= -5 | 5 <= j)` ***Answer:*** *ERROR*

   `i != j ? ++i * (3 + 0.5): ++i` ***Answer:*** *1.0*

   `++i - 7 * i++` ***Answer:*** *-6*

2. (4 Punkte) Ergänzen Sie die folgenden vier Ausdrücke um genau einen Operator an der angegebenen Stelle so, dass die Ausdrücke zu den angegebenen Werten evaluiert werden. Die Variable `i` ist vom Typ **`int`** und hat zu Beginn der Evalution jedes Ausdrucks den Wert 0.

   Complete the following four expressions by writing a single operator into the blanks such that the expressions evaluate to the given values. Assume that the variable `i` is of type **`int`** and has the value 0 before evaluating each expression.

   `6 +` ***Answer:*** `- 6 * 8` $\Rightarrow -42$

   `(true ^ false)` ***Answer:*** `|| (!true & false)` $\Rightarrow$ `true`

   `i` ***Answer:*** `!= 0 ? Math.sin(0): 1337` $\Rightarrow 1337$

   `true` ***Answer:*** `|| ++i > 0 ? i : 1` $\Rightarrow 0$

Aufgabe 4 (16 Punkte) Unten sehen Sie die Definition einer simplen, listenbasierten Statistik-Klasse, in etwa wie Sie sie aus den Hausaufgaben kennen. Schreiben Sie eine neue Version der Klasse, welche zum Speichern der Zahlen keine Liste, sondern ein Array verwendet. Ergänzen Sie Ihre Implementationen der Methoden `addNumber(int)` und `removeNumber(int)` um geeignete Exceptions. Denken Sie daran, dass Ihre Implementierung nicht in der Anzahl hinzufügbarer Zahlen beschränkt sein soll.

Um die volle Punktzahl zu erreichen denken Sie daran, Ihren Code zu kommentieren. Sie brauchen benutzte Klassen nicht importieren.

Below you will find a simple, list-based statistics class, not unlike the one you developed as part of your homework. Write a new version of this class that uses an array instead of a list to store the numbers. Extend your implementations of the methods `addNumber(int)` and `removeNumber(int)` to throw exceptions where appropriate. Note that your implementation should allow an arbitrary number of numbers to be added.

To receive full credit, remember to comment your code. You do not need to import classes you use.

```
import java.util.ArrayList;
import java.util.List;

public class Collector {
  private List<Integer> numbers = new ArrayList<>();

  public void addNumber(int number) {
    if (number > 0) {
      numbers.add(number);
    }
  }

  public void removeNumber(int index) {
    numbers.remove(index);
  }

  public double mean() {
    int sum = 0;
    for (int number : numbers) {
      sum += number;
    }
    return sum / (double) numbers.size();
  }
}
```

```
/**
 * One possible solution. Important parts are that we need to count
 * how many numbers are in the array (since we can have unused spots),
 * that we need to reset unused spots to zero because we sum up the
 * whole array in mean(), and that we need to copy stuff around in
 * addNumber(int) and removeNumber(int).
 */
```

Aufgabe 5 (10 Punkte)

1. (6 Punkte) Gegeben sei das folgende Programm. Geben Sie an, was auf der Konsole ausgegeben wird.

   Consider the following program. Write down what will be printed out on the console.

```
import java.util.ArrayList;
import java.util.List;

public class DrStrange {
  private static List<Integer> numbers;

  public DrStrange() {
    numbers = new ArrayList<>();
  }

  public void add(int number) {
    numbers.add(number);
  }

  public int sum() {
    int sum = 0;
    for (int number : numbers) {
      sum += number;
    }
    return sum;
  }

  public static void main(String[] args) {
    DrStrange[] mysteries = new DrStrange[5];

    for (int i = 0; i < 25; i++) {
      int index = i / 5;
      if (i % 5 == 0) {
        mysteries[index] = new DrStrange();
      }
      mysteries[index].add(i);
    }

    for (int i = 0; i < mysteries.length; i++) {
      System.out.println(mysteries[i].sum());
    }
  }
}
```

   Answer: *110 110 110 110 110 (on separate lines)*

2. (4 Punkte) Die Ausgabe entspricht nicht dem Verhalten, welches man eigentlich haben möchte. Erklären Sie, wie die Ausgabe zustande kommt, und geben Sie an, was zu tun ist, um das erwartete Verhalten zu erreichen, ohne die `main`-Methode zu verändern.

   The output is not what one would expect. Explain the output and suggest a way to change the program such that it works as expected without changing the `main` method. ***Answer:*** *Creating a new `DrStrange` object creates a new list, which, being* ***`static`****, is shared by all instances. Turning `numbers` into an instance variable solves the problem.*

Aufgabe 6 (12 Punkte) Eine *FIFO-Queue* ist eine Datenstruktur, die nach dem *first-in-first-out*-Prinzip funktioniert. Die Queue hat eine begrenzte, bei ihrer Erstellung anzugebende Kapazität > 0. Ihr können dann Elemente hinzugefügt (`push`) und entfernt (`pop`) werden. Die Reihenfolge, in welcher letztere Operation die Elemente zurückliefert, entspricht der Reihenfolge, in welcher sie hinzugefügt wurden.

Beispiel: Nach den Aufrufen `push(4)` und `push(2)` würde `pop()` die 4 zurückliefern und aus der Queue entfernen.

Vervollständigen Sie die Klasse `Queue`, um eine generische, kapazitätsbeschränkte FIFO-Queue bereitzustellen. Werfen Sie eine `IllegalArgumentException`, falls die Kapazität nicht valide ist. Werfen Sie jeweils eine `IllegalStateException` falls `push` auf einer vollen oder `pop` auf einer leeren Queue aufgerufen wird. Benutzen Sie zum Speichern der Queue-Elemente ein Array.

A *FIFO queue* is a data structure that works according to the *first-in first-out* principle. It can store up to a certain number of elements > 0 which has to be defined upon its creation. Elements can added and removed through the `push` and `pop` operations, respectively. The latter returns elements in the order in which they were added.

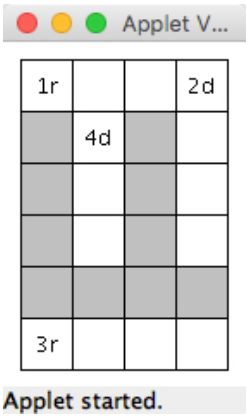
Example: after having called `push(4)` and `push(2)`, `pop()` would remove and return 4.

Complete the `Queue` class below such that it implements a generic queue with limited capacity. If the capacity is invalid, throw an `IllegalArgumentException`. If `push` is called on a full queue or if `pop` is called on an empty queue, throw an `IllegalStateException`. Use an array to store the queue's elements.

Answer:

```
package programming.exam.final1;

public class Queue<T> {
  T[] elements;
  int size = 0;
  /** Front points to the first free slot. */
  int front = 0;
  /** Back points to the last used slot. */
  int back = 0;

  public Queue(int capacity) {
    elements = (T[]) new Object[capacity];
  }

  public void push(T item) {
    if (size == elements.length) {
      throw new IllegalStateException("Queue is full");
    }

    elements[front] = item;
    front = (front + 1) % elements.length;
    size++;
  }

  public T pop() {
    if (size == 0) {
      throw new IllegalStateException("Queue is empty");
    }

    T result = elements[back];
    back = (back + 1) % elements.length;
    size--;

    return result;
  }
}
```

Aufgabe 7 (30 Punkte) Rechts sehen Sie ein Spielfeld für ein simples Kreuzworträtsel. Zu ratende Worte sind ab 1 durchnummeriert und mit einem Buchstaben versehen, der angibt, ob das Wort rechts neben der Bezeichnung (`r`) oder darunter (`d`) eingetragen werden soll.



Ihre Aufgabe ist es, ein Programm zu schreiben, welches derlei Spielfelder zeichnet. Die Spezifikation des Spielfeldes soll das Programm vom Benutzer erfragen, in folgender Reihenfolge (nicht valide Eingaben sollen erneut abgefragt werden):

- Breite (≥ 1 und ≤ 25)
- Höhe (≥ 1 und ≤ 25)
- Anzahl der Worte (≥ 1 und ≤ 15)
- Für jedes Wort:
  - Richtung (`r` oder `d`)
  - Reihe (≥ 0 und so, dass das Wort mindestens Länge 1 haben kann)
  - Spalte (≥ 0 und so, dass das Wort mindestens Länge 1 haben kann)
  - Länge (≥ 1 und maximal so, dass es auf das Spielfeld passt).

Das abgebildete Spielfeld enthält folgende Worte (Richtung, Reihe, Spalte, Länge):

(r, 0, 0, 2) (d, 0, 3, 3) (r, 5, 0, 3) (d, 1, 1, 2)

Ihr Programm muss nicht verhindern, dass sich Wort überkreuzen. Um die volle Punktzahl zu erreichen denken Sie daran, Ihren Code zu kommentieren. Sie brauchen benutzte Klassen nicht importieren.

Above is an image of a simple crosswords field. The words are numbered starting at 1 and annotated with a letter which specified whether the words should be filled in rightwards (`r`) or downwards (`d`).

Your assignment is to write a program that draws such crossword fields. Users should be able to input the field's specifications, in the following order (where invalid input must be re-input until valid):

- Width (≥ 1 and ≤ 25)
- Height (≥ 1 and ≤ 25)
- Number of words (≥ 1 and ≤ 15)
- For each word:
  - Direction (`r` or `d`)
  - Row (≥ 0 and such that the word can have at least one character)
  - Column (≥ 0 and such that the word can have at least one character)
  - Length (≥ 1 and such that it fits on the field)

The field above contains the following words (direction, row, column, length):

(r, 0, 0, 2) (d, 0, 3, 3) (r, 5, 0, 3) (d, 1, 1, 2)

Your program does not have to prevent words from crossing each other. To receive full credit, remember to comment your code. You do not need to import classes you use.

Useful classes, constructors, and methods:

```
public GRect(double x, double y, double width, double height)
public GLabel(String text, double x, double y)

public int readInt(String prompt, int lowerBound, int upperBound)
```

```
public class Crosswords extends GraphicsProgram {
  /** Top left cell's coordinates are (BORDER_SPACING, BORDER_SPACING). */
  private static final int BORDER_SPACING = 10;
  /** Width and height of each cell. */
  private static final int CELL_SIZE = 30;
```

```
public void run() {
```

Answer:

```
package programming.exam.final1;

import java.awt.Color;

import acm.graphics.GLabel;
import acm.graphics.GRect;
import acm.program.GraphicsProgram;

public class Crosswords extends GraphicsProgram {

  /** Top left cell's coordinates are (BORDER_SPACING, BORDER_SPACING). */
  private static final int BORDER_SPACING = 10;
  /** Width and height of each cell. */
  private static final int CELL_SIZE = 30;

  @Override
  public void run() {
    int width = readInt("Field width: ", 1, 25);
    int height = readInt("Field height: ", 1, 25);

    GRect[][] field = createField(width, height);

    addWords(field);
  }

  private GRect[][] createField(int width, int height) {
    GRect[][] field = new GRect[height][width];

    for (int row = 0; row < height; row++) {
      for (int col = 0; col < width; col++) {
        field[row][col] = new GRect(CELL_SIZE, CELL_SIZE);
        field[row][col].setLocation(
            col * CELL_SIZE + BORDER_SPACING,
            row * CELL_SIZE + BORDER_SPACING);

        field[row][col].setFilled(true);
        field[row][col].setColor(Color.BLACK);
        field[row][col].setFillColor(Color.LIGHT_GRAY);

        add(field[row][col]);
      }
    }

    return field;
  }

  private void addWords(GRect[][] field) {
    int wordCount = readInt("Word count: ", 1, 15);

    for (int word = 1; word <= wordCount; word++) {
      // Word direction
      String dir;
      do {
        dir = readLine("Direction (r, d): ");
      } while (!(dir.equals("r") || dir.equals("d")));
```

```
      boolean right = dir.equals("r");

      // Row and column of the word's description field
      int maxRow = right ? field.length - 1 : field.length - 2;
      int row = readInt("Start row: ", 0, maxRow);

      int maxCol = right ? field[row].length - 2 : field[row].length - 1;
      int col = readInt("Start column: ", 0, maxCol);

      // Word length
      int maxLength = right ? field[row].length - 1 - col : field.length -
          1 - row;
      int length = readInt("Length: ", 1, maxLength);

      // Make rectangles white
      if (right) {
        for (int x = col; x <= col + length; x++) {
          field[row][x].setFillColor(Color.WHITE);
        }
      } else {
        for (int y = row; y <= row + length; y++) {
          field[y][col].setFillColor(Color.WHITE);
        }
      }

      // Draw label
      GLabel label = new GLabel(word + dir);
      label.setLocation(
          BORDER_SPACING + col * CELL_SIZE + (CELL_SIZE - label.getWidth())
              / 2,
          BORDER_SPACING + row * CELL_SIZE + (CELL_SIZE -
              label.getHeight()) / 2 + label.getAscent());
      add(label);
    }
  }

}
```

Aufgabe 8 (10 Bonus Punkte)

1. Wie lautet die formale Definition einer kontextfreien Grammatik aus der Vorlesung?

   Give the formal definition of *context free grammars* we provided in the lecture.

   Answer: *A tuple of* $G = (V, \Sigma, R, S)$ *consisting of set of Variables* V*, an alphabet/set of terminals* Σ*, a set of productions* $R : V \to (V \cup \Sigma)*$*, and a start variable* $S \in V$*.*

2. Geben Sie für die folgenden Sprachbeschreibungen jeweils eine passende CFG an oder erklären Sie, warum keine solche existiert. Sie können Ihre Grammatik sowohl formal als auch als EBNF angeben.

   Provide a context-free grammar for each of the following languages or explain why none exists. You can specify your grammars either formally or as an EBNF.

   a) Ein Integer-Literal welches, wenn es mit einer Null beginnt, nur aus Ziffern zwischen Null und Sieben bestehen darf, ansonsten aus Ziffern zwischen Null und Neun. Die erste Ziffer nach der Null darf keine Null sein.

      An integer literal, whose digits may only be between zero and seven if it starts with a zero, but can be between zero and nine otherwise. If it starts with a zero, the second digit must not be a zero.

      Answer:

      ```
      IntLiteral ::= '0' OctLiteral | DecLiteral
      OctLiteral ::= ('1' | ...  | '7') ('0' | ...  | '7')*
      DecLiteral ::= ('1' | ...  | '9') ('0' | ...  | '9')*
      ```

   b) Eine beliebige Folge von Ziffern zwischen Eins und Vier, gefolgt von einem Punkt und derselben Folge von Ziffern in umgekehrter Reihenfolge.

      An arbitrary sequence of digits between one and four, followed by a period and the same sequence of digits in reverse order.

      Answer:

      ```
      Thing ::= '.'  | '1' Thing '1' | '2' Thing '2' | '3' Thing '3' | '4' Thing '4'
      ```

   c) Eine beliebige Folge von Ziffern zwischen Eins und Vier, gefolgt von einem Punkt und derselben Folge von Ziffern.

      An arbitrary sequence of digits between one and four, followed by a period and the same sequence of digits.

      Answer: *Not possible because context-free grammars cannot recall sequences.*