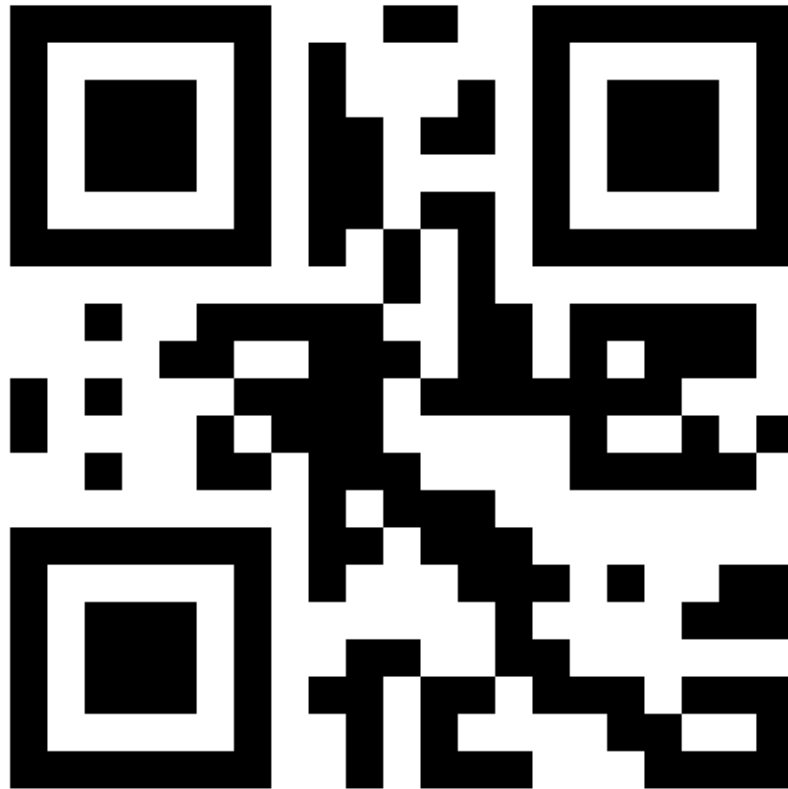


# Von der Information zum QR-Code und wieder zurück



## QR-Codes als Thema im Informatikunterricht

von  
Claudia Strödter  
Friedrich-Schiller-Universität Jena

# Inhalt

|  |           |
|--|-----------|
| <b>Von der Information zum QR-Code und wieder zurück</b> . . . . .                           | <b>3</b>  |
| <hr/>  |           |
| <b>1 Der QR-Code</b> . . . . .   | <b>4</b>  |
| <hr/>  |           |
| 1.1 Der Aufbau eines QR-Codes . . . . .  | 5         |
| 1.1.1 Das Modul . . . . .  | 5         |
| 1.1.2 Das Finder-Pattern . . . . .   | 6         |
| 1.1.3 Das Alignment-Pattern . . . . .  | 6         |
| 1.1.4 Das Timing-Pattern . . . . .   | 6         |
| 1.1.5 Die Format- und Versionsinformation . . . . .  | 7         |
| 1.1.6 Die Codewörter . . . . .   | 7         |
| 1.1.7 Der Modus und das Fehlerkorrekturniveau . . . . .                                      | 8         |
| 1.2 Von der Information zum QR-Code: Encodierung . . . . .                                   | 9         |
| 1.2.1 Grundsätzliches zur Encodierung . . . . .  | 9         |
| 1.2.2 Die Encodierung – anschaulich erklärt . . . . .  | 10        |
| 1.3 Vom QR-Code zur Information: Decodierung . . . . .                                       | 18        |
| 1.3.1 Grundsätzliches zur Decodierung . . . . .  | 18        |
| 1.3.2 Die Decodierung – anschaulich erklärt . . . . .  | 19        |
| <hr/>  |           |
| <b>2 Die Unterrichtsreihe</b>  |           |
| <b>»Von der Information zum QR-Code und wieder zurück«</b> . . . . .                         | <b>22</b> |
| <hr/>  |           |
| 2.1 Erste Stunde:<br>Die 2-D-Codes . . . . .   | 22        |
| 2.2 Zweite Stunde:<br>Die QR-Codes . . . . .   | 23        |
| 2.3 Dritte und vierte Stunde:<br>Von der Information zum QR-Code und wieder zurück . . . . . | 25        |
| 2.4 Auch KARA kann QR-Codes. . . . .   | 27        |
| <hr/>  |           |
| <b>3 Fazit</b> . . . . .   | <b>29</b> |
| <hr/>  |           |
| <b>4 Anhang</b> . . . . .  | <b>30</b> |
| <hr/>  |           |
| 4.1 Tabellen und Arbeitsblätter . . . . .  | 30        |
| 4.2 Nützliche Internetquellen . . . . .  | 37        |
| 4.3 Literatur und Internetquellen . . . . .  | 37        |

# Von der Information zum QR-Code und wieder zurück

QR-Codes als Thema im Informatikunterricht

von Claudia Strödter

Was vor 25 Jahren wohl eher als eine historische Wandmalerei (siehe Abbildung 1) oder abstrakte Kunst angesehen wurde, löst heute den Griff zum Smartphone aus.

QR-Codes sind ein fester Bestandteil des täglichen Lebens. Man findet sie in Zeitschriften, auf Plakaten und Visitenkarten, an Messeständen und Museums-Exponaten. Somit knüpft das Thema *QR-Codes* unmittelbar an die Lebenswelt von Schülerinnen und Schülern an und greift gleichzeitig wichtige informatische Inhalte auf. Im Folgenden werden die theoretischen Grundlagen der QR-Encodierung und -Decodierung erläutert und die Unterrichtsreihe *Von der Information zum QR-Code und wieder zurück* vorgestellt. Die Unterrichtsreihe ist so konzipiert, dass gezielt Kompetenzen aus den GI-Empfehlungen zu den *Bildungsstandards Informatik für die Sekundarstufe I* (AKBSI, 2008) aufgegriffen werden. Insbesondere werden die folgenden Kompetenzen gefördert:

## **Inhaltsbereiche:**

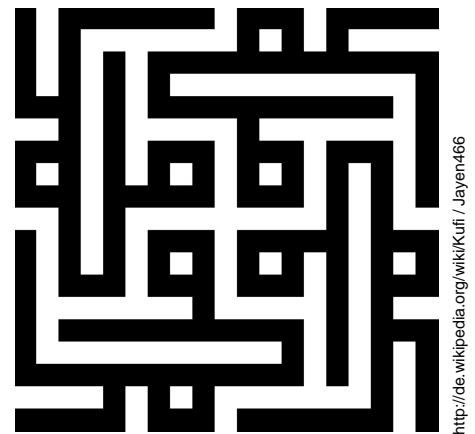
Schülerinnen und Schüler

- ▷ unterscheiden Bedeutung und Darstellungsform einer Nachricht,
- ▷ stellen Information in unterschiedlicher Form dar,
- ▷ beurteilen Vor- und Nachteile unterschiedlicher Informationsdarstellungen,
- ▷ kennen und verwenden die Datentypen Text, Zahl und Wahrheitswert,
- ▷ erläutern das Prinzip der Eingabe, Verarbeitung und Ausgabe von Daten (EVA-Prinzip) als grundlegendes Arbeitsprinzip von Informatiksystemen,
- ▷ unterscheiden Eingaben und Ausgaben (realer Automaten),
- ▷ interpretieren Handlungsvorschriften korrekt und führen sie schrittweise aus,
- ▷ lesen und verstehen Handlungsvorschriften für das Arbeiten mit Informatiksystemen,
- ▷ entwerfen Handlungsvorschriften als Text oder mit formalen Darstellungsformen,
- ▷ verwenden Variablen und Wertzuweisungen,
- ▷ entwerfen und testen einfache Algorithmen,
- ▷ entwerfen und implementieren Algorithmen,
- ▷ modifizieren und ergänzen Quelltexte von Programmen nach Vorgaben,
- ▷ überführen umgangssprachlich gegebene Handlungsvorschriften in formale Darstellungen,
- ▷ geben Problemlösungen in einer Programmiersprache an.

## **Prozessbereiche:**

Schülerinnen und Schüler

- ▷ lernen die potenziellen Gefahren bei der Nutzung digitaler Medien an Beispielen kennen,
- ▷ tauschen sich untereinander, mit Lehrkräften und anderen Personen verständlich über informatische Inhalte aus,
- ▷ stellen informatische Sachverhalte unter Benutzung von Fachbegriffen mündlich und schriftlich sachgerecht dar.



<http://de.wikipedia.org/wiki/Kufi> / Jayen466

Abbildung 1: Quadratisches Kufi aus dem 6. Jahrhundert. Ein quadratisches Kufi ist eine stark vereinfachte Variante der kufischen Schrift – eine der ältesten kalligrafischen Formen der arabischen Schrift –, die in der islamischen Architektur zum Beispiel zur Verzierung gefliester Fassaden dient. In dem abgebildeten Kufi wird der Name »Mohammed« viermal wiederholt.

# 1

## Der QR-Code

QR-Codes sind die bekanntesten und im Alltag am häufigsten verwendeten zweidimensionalen Codes. Die 2-D-Codes wurden ursprünglich zur Kennzeichnung von Bauteilen in der Industrie und Raumfahrt entwickelt. Heute ermöglichen sie unter anderem das Versenden von Briefen, das Bahnfahren und geben mit wenigen Knopfdrücken den Zugang zu verschiedenen digitalen Medien frei.

Der Begriff stammt aus dem Englischen und ist eine Abkürzung für *Quick Response*, d. h. »schnelle Antwort«. Der erste QR-Code wurde im Jahr 1994 von der Firma *DENSO WAVE* erfunden, einer Tochterfirma des japanischen Automobilzulieferers *DENSO Corporation*, die für automatische Identifikation und Datenerfassung (Auto-ID), Industrieroboter sowie Spezialelektronik für die Automation in der Fertigung zuständig ist.

Spätestens als von der Deutschen Bahn im Jahr 2002 die Online-Tickets flächendeckend eingesetzt wurden, hielten die 2-D-Codes auch Einzug in die deutschen Haushalte. Heute existieren über 40 verschiedene 2-D-Codes, die in Stapel- und Matrix-Codes (siehe Tabelle 1) unterschieden werden (vgl. Uitz/Harnisch, 2012).

Mit der zunehmenden Weiterentwicklung von Smartphones wurde die Verbreitung von QR-Codes beschleunigt. Das sogenannte Mobile-Tagging (siehe Abbildung 2) ermöglicht das einfache Scannen und Decodieren der QR-Codes.

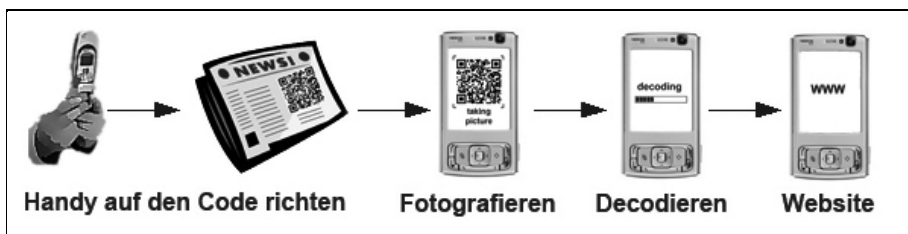
Meist wird nur eine einfache URL encodiert, die auf zusätzliche Informationen (z. B. digitale Medien) verweist, aber immer häufiger werden auch Telefonnummern, E-Mail-Adressen oder vollständige Adressdaten in Form sogenannter vCards hinterlegt (vCards enthalten alle notwendigen Adressdaten zu einer Person und können ohne weitere Zwischenschritte in die eigene Adress-Datenbank – z. B. in das Telefonbuch eines Smartphones – importiert werden). Dabei entfällt das mühevoll Eintippen von URLs oder Adressdaten.

QR-Codes sind nur wenig fehleranfällig. Es ist möglich, bis zu einem gewissen Grad verschmutzte oder beschädigte Codes zu decodieren. Eine falsche Orientierung oder Verzerrung kann ebenfalls kompensiert werden (siehe Abbildung 3).

| Stapel-Code  | Matrix-Code            | Matrix-Code  |
|--|------------------------|--|
| <p>PDF417</p>  | <p>Aztec-Code</p>      | <p>QR-Code</p>   |
| <p>Weitere Beispiele:</p> <p>Codablock<br/>Code 49</p> | <p>DataMatrix-Code</p> | <p>Weitere Beispiele:</p> <p>Micro-QR-Code<br/>Secure QR-Code<br/>MaxiCode</p> |

Tabelle 1 (oben):  
Beispiele für 2-D-Codes.

Abbildung 2 (rechts): Mobile-Tagging  
(Erfassen einer Markierung für das  
Mobiltelefon) eines QR-Codes.



[http://de.wikipedia.org/wiki/Mobile-Tagging/1\\_Mhagen](http://de.wikipedia.org/wiki/Mobile-Tagging/1_Mhagen)

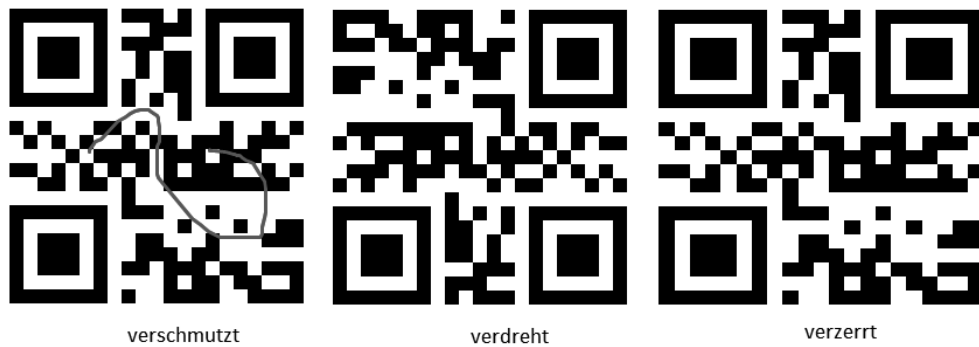
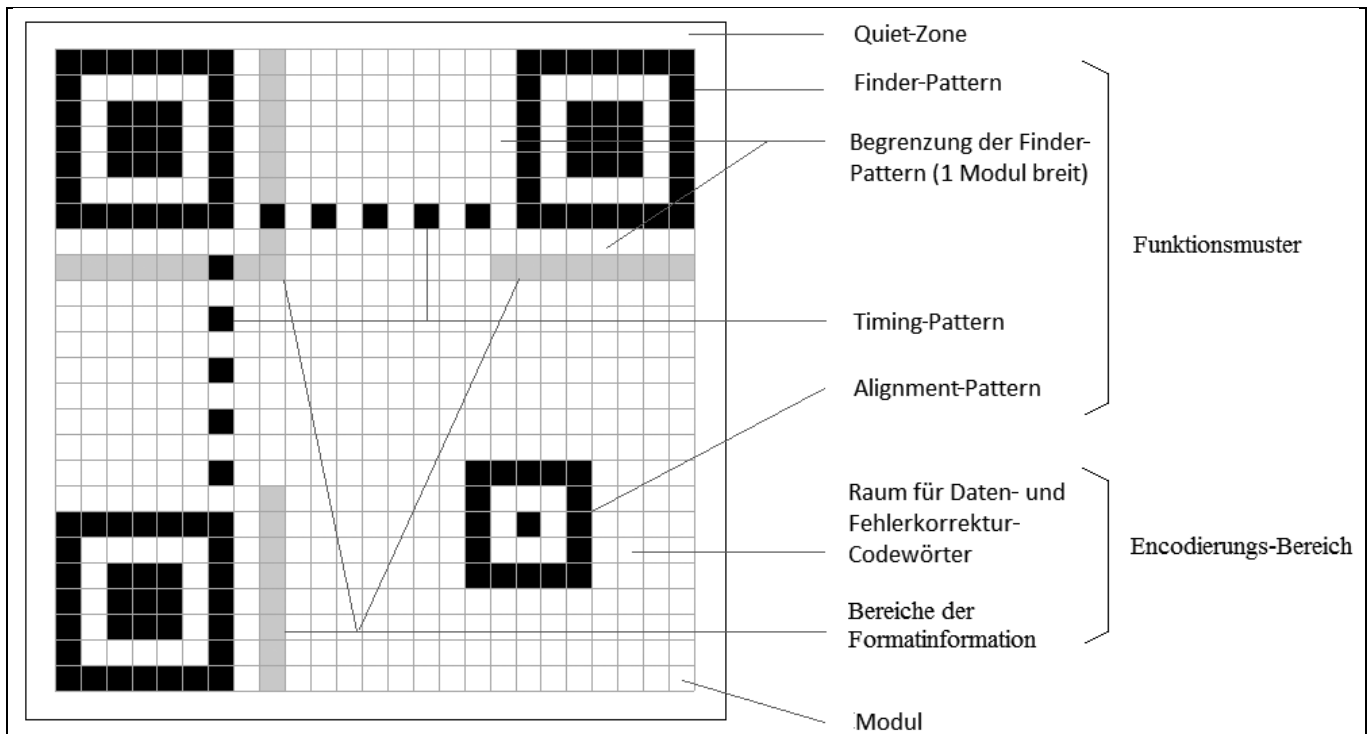


Abbildung 3 (links): Fehlerhafte, aber lesbare QR-Codes.



Der im Folgenden vorgestellte Aufbau eines QR-Codes und die beschriebenen Algorithmen folgen dem ISO-Standard von 2006 (vgl. ISO/IEC, 2000 und 2006). Zum besseren Verständnis beschränken sich die Ausführungen auf die QR-Code-Versionen 1 und 2. Höhere Versionen erfordern aufwendigere Berechnungen und bleiben daher unberücksichtigt.

### Der Aufbau eines QR-Codes

QR-Codes sind quadratische Raster. Sie bestehen aus Funktionsmustern und dem Encodierungs-Bereich (Zeichencodierungs-Bereich), der die eigentlichen Daten enthält. Das gesamte Raster ist von einer weißen Fläche (Quiet-Zone) umgeben, um die Lesbarkeit auf dunklen Hintergründen zu gewährleisten (siehe Abbildung 4).

#### Das Modul

Die kleinste Einheit bildet das Modul (siehe Abbildung 5). Es besteht aus einer quadratischen Fläche einheitlicher Größe und kann entweder hell (meist weiß) oder dunkel (meist schwarz) gefärbt sein. Dabei steht hell für die binäre 0 und



Abbildung 4: Aufbau eines QR-Codes der Version 2.

Abbildung 5: Die zwei Ausprägungen eines Moduls.

http://www.qrcode.com/en/about/version.html

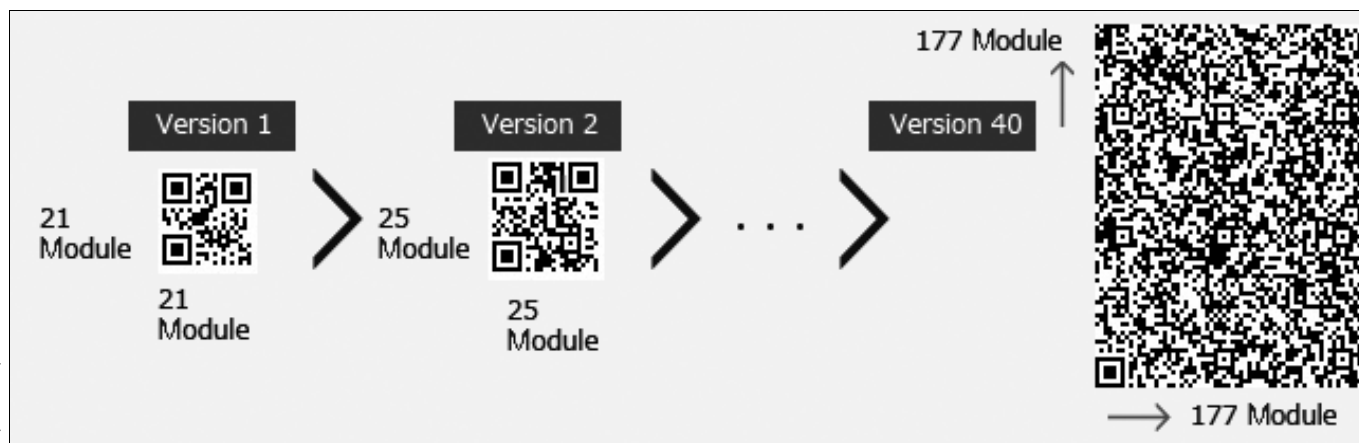


Abbildung 6 (oben):  
QR-Code-Versionen.

Abbildung 7 (unten):  
Aufbau eines Finder-Patterns  
mit Begrenzung.

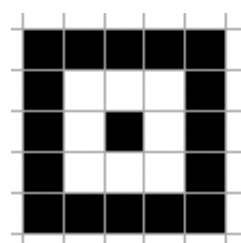
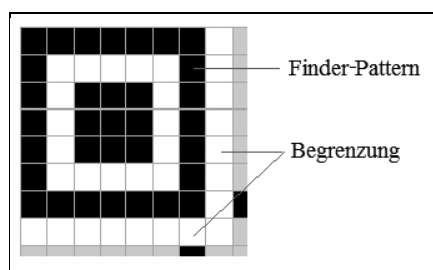


Abbildung 8:  
Aufbau eines Alignment-Patterns.

dunkel für die binäre 1. Ein Modul entspricht somit genau einem Bit. Jedes Modul hat eine eindeutige Position, die durch ihre Koordinaten bestimmt ist. Je nach Position werden sie als Funktionsmodule (Teil eines Funktionsmusters) oder Datenmodule (Teil des Encodierungs-Bereichs) bezeichnet.

Die Anzahl der verwendeten Module und somit auch die Speicherkapazität wird durch die verwendete QR-Code-Version festgelegt: Version 1 besteht aus 21×21 Modulen, Version 2 aus 25×25 Modulen. Mit zunehmender Version wächst die Kantenlänge eines Rasters um 4 Module. Die derzeit größte QR-Code-Version besteht aus 177×177 Modulen (Version 40; siehe Abbildung 6).

#### Das Finder-Pattern (Suchmuster)

Wird ein QR-Code verdreht aufgenommen oder abgedruckt, kann dies mithilfe der Finder-Pattern korrigiert werden. Alle QR-Code-Versionen enthalten genau drei Finder-Pattern (oben links, oben rechts und unten links). Diese quadratischen Funktionsmuster haben eine Kantenlänge von sieben Modulen und definieren indirekt die Modulgröße eines QR-Codes (siehe Abbildung 7). Zusätzlich ermöglichen sie das Erkennen der Ausrichtung eines QR-Codes, sodass er immer in die richtige Position gedreht werden kann.

Damit die Finder-Pattern möglichst schnell erkannt werden, sind sie innerhalb des Rasters von einer ein Modul breiten hellen Sequenz umgeben. Dieser Bereich kann weder von weiteren Funktionsmustern noch von Codewörtern belegt werden.

#### Das Alignment-Pattern (Ausrichtungsmuster)

Wird ein QR-Code nicht senkrecht aufgenommen oder auf unebenen Flächen abgedruckt, kommt es zur Verzerrung des Symbols. Mithilfe der quadratischen Alignment-Pattern kann dies korrigiert werden. Die Gestalt eines Alignment-Pattern ist mit einer Kantenlänge von fünf Modulen immer gleich (siehe Abbildung 8). Die Anzahl dieser Funktionsmuster innerhalb eines QR-Codes ist abhängig von der Größe bzw. der Version des QR-Codes: Version 1 enthält kein, Version 2 bis 6 ein, bis zur Version 40 mit 46 Alignment-Pattern. Jedes Alignment-Pattern hat eine vorgeschriebene Position. Die theoretische Position wird mit der aufgenommenen Position verglichen. Die Abweichung wird berechnet und die entstandene Verzerrung korrigiert.

#### Das Timing-Pattern (Steuerungsmuster)

Alle QR-Code-Versionen enthalten zwei Timing-Pattern, die ausgehend von Spalte 6 und Zeile 6 horizontal und vertikal verlaufen (siehe Abbildung 9, nächste Seite). Sie sind ein Modul breit und abwechselnd aus dunklen und hellen Modulen zusammengesetzt. Im Decodierungs-Prozess werden die Timing-Pattern genutzt, um die Koordinaten der einzelnen Module zu bestimmen.

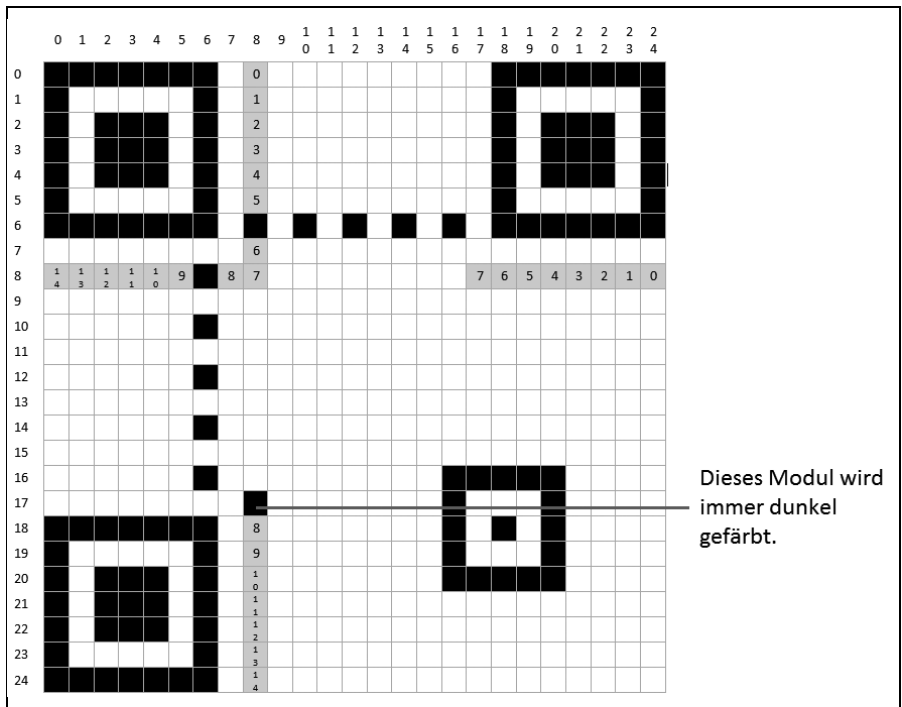


Abbildung 9: Position und Ausrichtung der Timing-Pattern und der Formatinformationen in einem QR-Code der Version 2.

**Die Format- und Versionsinformationen**

Die Formatinformationen umfassen 15 Bit und enthalten Informationen zum Fehlerkorrekturniveau und zur verwendeten Maske (vgl. Abschnitt »Von der Information zum QR-Code«, Seite 9 ff.). Ohne die Formatinformationen kann ein QR-Code nicht decodiert werden, deshalb werden diese Informationen in allen QR-Code-Versionen doppelt dargestellt (siehe in der Abbildung 9 die grau gefärbten Module). Insgesamt sind im QR-Code-Raster 31 Module für das Eintragen der Formatinformationen vorgesehen. Aus diesem Grund wird das Modul in Spalte 8 und der achten Zeile von unten immer dunkel gefärbt.

Die QR-Code-Version kann für die Versionen 1 bis 6 anhand der Rastergröße berechnet werden. Ab Version 7 werden zusätzlich Versionsinformationen im QR-Code hinterlegt.

**Die Codewörter**

Den größten Teil des Encodierungs-Bereichs nehmen die Daten-Codewörter (beinhalten die eigentlichen Daten) und Fehlerkorrektur-Codewörter (beinhalten die Daten zur Fehlerkorrektur) ein (siehe auch Abbildung 12). Ein Codewort (CW) umfasst acht Bit (z.B. 01011011) und wird meist als Block von 2x4 Modulen aufwärts oder abwärts im QR-Code dargestellt (siehe Abbildung 10).

Stößt ein Codewort an eine Grenze, wird die Richtung umgekehrt und das Codewort in die benachbarten Spalten eingetragen. Kommt es zur Behinderung durch ein Funktionsmuster, umfließt das Codewort das Muster (siehe Abbildung 11, nächste Seite).

Unter Berücksichtigung der in den Abbildungen 10 und 11 gezeigten Vorgaben werden die Codewörter – in der unteren rechten Ecke beginnend – in das

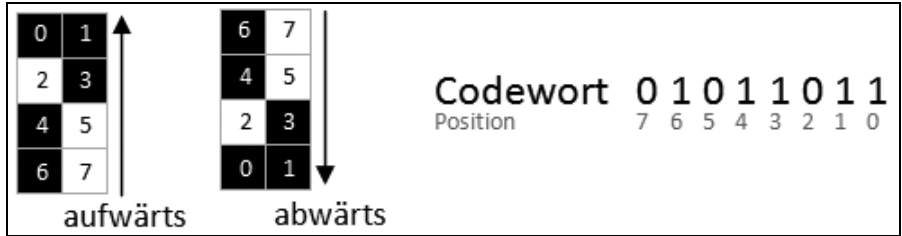


Abbildung 10: Darstellung eines Codeworts (cw); im QR-Code am Beispiel »01011011«.





| Ver-<br>sion | Fehler-<br>korrektur-<br>niveau | Anzahl<br>der Daten-<br>Code-<br>wörter | Anzahl<br>der Fehler-<br>korrektur-<br>Codewörter | Datenkapazität<br>(Anzahl der Zeichen in den verschiedenen Modi) |                   |      |       |
|--------------|---------------------------------|---|---|--|-------------------|------|-------|
|              |                                 |   |   | Numeric  | Alpha-<br>numeric | Byte | Kanji |
| 1            | L                               | 19                                      | 7   | 41   | 25                | 17   | 10    |
|              | M                               | 16                                      | 10  | 34   | 20                | 14   | 8     |
|              | Q                               | 13                                      | 13  | 27   | 16                | 11   | 7     |
|              | H                               | 9                                       | 17  | 17   | 10                | 7    | 1     |
| 2            | L                               | 34                                      | 10  | 77   | 47                | 32   | 20    |
|              | M                               | 28                                      | 16  | 63   | 38                | 26   | 16    |
|              | Q                               | 22                                      | 23  | 48   | 29                | 20   | 12    |
|              | H                               | 16                                      | 28  | 34   | 20                | 14   | 8     |
| ...          |                                 |   |   |  |                   |      |       |
| 40           | L                               | 2956                                    | 750   | 7089   | 4296              | 2953 | 1817  |
|              | M                               | 2334                                    | 1372  | 5596   | 3391              | 2331 | 1435  |
|              | Q                               | 1666                                    | 2040  | 3993   | 2420              | 1663 | 1024  |
|              | H                               | 1276                                    | 2430  | 3057   | 1852              | 1273 | 784   |

Tabelle 3:  
QR-Code-Versionen  
und ihre Kapazitäten.

- ▷ Niveau Q ermöglicht das Wiederherstellen von ca. 25 % der Daten-Codewörter und
- ▷ Niveau H ermöglicht das Wiederherstellen von ca. 30 % der Daten-Codewörter.

### Von der Information zum QR-Code: Encodierung

#### Grundsätzliches zur Encodierung

Das Erzeugen eines QR-Codes aus einer gegebenen Information wird als *Encodierung* bezeichnet. Bei der Encodierung werden folgende Schritte durchlaufen:  
(*Digitalisierung der Zeichenkette*)

1. Daten-Codewörter berechnen,
2. Fehlerkorrektur-Codewörter berechnen,
3. Ausgangsmatrix erstellen,

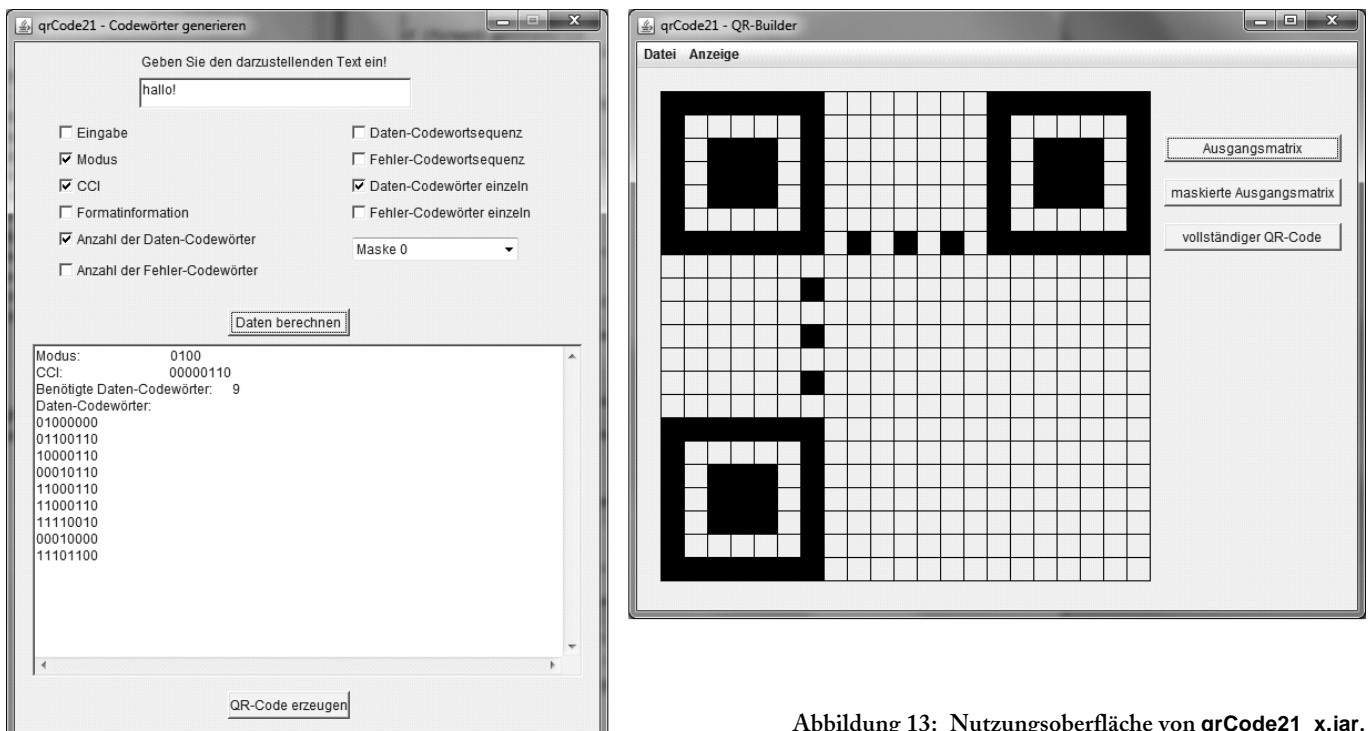


Abbildung 13: Nutzungsoberfläche von qrCode21\_x.jar.

- 4. Maskieren,
  - 5. Formatinformationen hinzufügen.
- (Generierung des QR-Codes)

Da mit zunehmender QR-Code-Version auch die Komplexität der Berechnungen ansteigt, werden nachfolgend ausschließlich QR-Codes der Version 1 betrachtet. Die Encodierungs-Schritte werden für die Modi *Numeric*, *Alphanumeric* und *Byte* an den Beispiel-Zeichenketten »01234«, »TEST« und »Hallo!« erläutert. Die Digitalisierung der Zeichenkette und die Generierung des QR-Codes werden nicht näher betrachtet.

Alle Schritte werden so erläutert, dass sie ohne Nutzung eines Informatiksystems (also per Hand) nachvollzogen werden können. Zur besseren Veranschaulichung kann zusätzlich die Anwendung qrCode21\_x.jar verwendet werden (siehe Abbildung 13, vorige Seite; die Anwendung wurde von der Autorin entwickelt und wird über den LOG-IN-Service zur Verfügung gestellt).

Die Anwendung ist für Lehrende konzipiert und soll die Vorbereitung und Durchführung der Unterrichtsreihe unterstützen. qrCode21\_x.jar ermöglicht das Erstellen von QR-Codes der Version 1 in den Modi *Numeric*, *Alphanumeric* und *Byte*. Alle im Folgenden vorgestellten Schritte können mithilfe dieser Anwendung nachvollzogen werden.

**Die Encodierung – anschaulich erklärt**

Der größte Teil eines QR-Codes wird von der Codewort-Sequenz eingenommen. Die Sequenz ist aus Daten- und Fehlerkorrektur-Codewörtern zusammengesetzt (siehe Abbildung 14).

Abbildung 14 (rechts):  
Aufbau der Codewort-Sequenz.

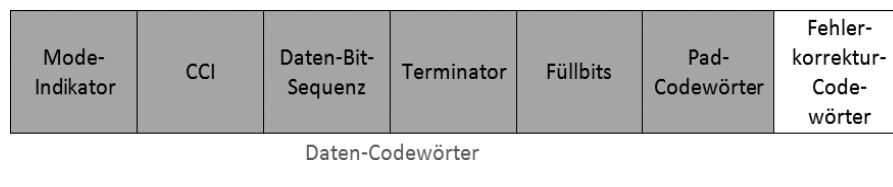


Tabelle 4 (unten):  
Vorgehen beim Berechnen von Daten-Codewörtern.

1. Schritt: Daten-Codewörter berechnen

Die einzelnen Abschnitte der Daten-Codewörter werden nach folgendem in Tabelle 4 wiedergegebenem Schema berechnet:

| Abschnitt         | Vorgehen   | Beispiel »01234«   | Beispiel »TEST«  | Beispiel »Hallo!«   |
|-------------------|--|--|--|---|
| Mode-Indikator    | <ul style="list-style-type: none"> <li>• gibt den verwendeten Modus an (vgl. Tab. 2, S.8)</li> <li>• umfasst 4 Bit</li> </ul>  | Numeric-Mode<br><b>0001</b>  | Alphanumeric-Mode<br><b>0010</b>   | Byte-Mode<br><b>0100</b>  |
| CCI               | <ul style="list-style-type: none"> <li>• gibt die Anzahl der zu encodierenden Zeichen in Binärdarstellung an</li> <li>• umfasst im Modus:<br/><b>Numeric</b> 10 Bit,<br/><b>Alphanumeric</b> 9 Bit<br/><b>Byte</b> 8 Bit</li> </ul>  | 01234<br>(5 Zeichen)<br><br><b>000000101</b>   | TEST<br>(4 Zeichen)<br><br><b>000000100</b>  | Hallo!<br>(6 Zeichen)<br><br><b>00000110</b>  |
| Daten-Bit-Sequenz | <p><b>Numeric-Mode:</b></p> <ul style="list-style-type: none"> <li>• Die Eingabedaten werden in 3er Blöcke unterteilt und in ihre 10-Bit-Binärdarstellung umgewandelt.</li> <li>• Besteht der letzte Block aus nur aus 2 bzw. 1 Zeichen, wird dieser in seine 7- bzw. 4-Bit-Binärdarstellung umgewandelt.</li> </ul> <p><b>Alphanumeric-Mode:</b></p> <ul style="list-style-type: none"> <li>• Die Eingabedaten werden in 2er Blöcke unterteilt.</li> <li>• Jedem Zeichen ist ein Wert zuzuordnen (vgl. Tab. 14, S.30).</li> <li>• Der erste Wert eines Zweierblocks wird mit 45 multipliziert.</li> <li>• Der zweite Wert des Zweierblocks wird zum Produkt addiert.</li> <li>• Das Ergebnis wird in seine 11-Bit-Binärdarstellung umgewandelt.</li> <li>• Besteht der letzte Block aus 1 Zeichen, wird der Wert des Zeichens in seine 6-Bit-Binärdarstellung umgewandelt.</li> </ul> | 012<br><br>34<br><br><br><br><br><br><br><br><br><b>0000001100</b><br><b>0100010</b> | TE 19 14<br>ST 18 19<br><br>(19*45)+14<br>=1319<br>(18*45)+19<br>=1289<br><br><b>10100100111</b><br><b>10100001001</b> | H 72<br>a 97<br>l 108<br>l 108<br>o 111<br>! 33<br><br><br><br><br><br><br><br><br><b>01001000</b><br><b>01100001</b><br><b>01101100</b><br><b>01101100</b><br><b>01101111</b><br><b>00100001</b> |

Fortsetzung nächste Seite

| Abschnitt                       | Vorgehen  | Beispiel<br>»01234«  | Beispiel<br>»TEST«   | Beispiel<br>»Hallo!«   |
|---------------------------------|---|--|--|--|
| Daten-Bit-Sequenz (Fortsetzung) | <b>Byte-Mode:</b> <ul style="list-style-type: none"> <li>Jedem Zeichen der Eingabedaten wird ein Wert (nach Zeichensatz JIS X0208, vgl. Tab. 15, S. 30) zugeordnet.</li> <li>Der Wert eines Zeichens wird in seine 8-Bit-Binärdarstellung umgewandelt.</li> </ul>                     |  |  |  |
|                                 | Die entstandenen Bit-Sequenzen werden aneinandergelängt und, sofern möglich, in 8 Bit lange Codewörter unterteilt.  | 00010000<br>00010100<br>00001100<br>0100010  | 00100000<br>00100101<br>00100111<br>10100001<br>001  | 01000000<br>01100100<br>10000110<br>00010110<br>11000110<br>11000110<br>11110010<br>0001                 |
| Terminator                      | <ul style="list-style-type: none"> <li>Die Bit-Sequenz <b>0000</b> signalisiert das Ende der Daten-Bit-Sequenz.</li> <li>Sind bis zum Ende aller Daten-Codewörter weniger als 4 Bit frei, so kann der Terminator gekürzt oder weggelassen werden.</li> </ul>                          | 00010000<br>00010100<br>00001100<br>01000100<br>000  | 00100000<br>00100101<br>00100111<br>10100001<br>0010000  | 01000000<br>01100100<br>10000110<br>00010110<br>11000110<br>11000110<br>11110010<br>00010000             |
| Füllbits                        | <ul style="list-style-type: none"> <li>Ist das bisher letzte Daten-Codewort kürzer als 8 Bit, so wird dieses Codewort mit Nullen vervollständigt.</li> </ul>  | 00010000<br>00010100<br>00001100<br>01000100<br>00000000   | 00100000<br>00100101<br>00100111<br>10100001<br>00100000   | 01000000<br>01100100<br>10000110<br>00010110<br>11000110<br>11000110<br>11110010<br>00010000             |
|                                 | Es wird eine geeignete Version und ein geeignetes Fehlerkorrekturniveau gesucht (siehe Tab. 3, S. 9). Ziel ist, einen möglichst kleinen, aber auch fehlertoleranten QR-Code zu generieren.  | Version 1H<br>9 Daten-Codewörter   | Version 1H<br>9 Daten-Codewörter   | Version 1H<br>9 Daten-Codewörter   |
| Pad-Codewörter                  | <ul style="list-style-type: none"> <li>Pad-Codewörter werden eingesetzt, wenn nicht alle zur Verfügung stehenden Daten-Codewörter verwendet werden.</li> <li>Die noch freien Daten-Codewörter werden abwechselnd mit den Pad-Codewörtern 11101100 und 00010001 aufgefüllt.</li> </ul> | 00010000<br>00010100<br>00001100<br>01000100<br>00000000<br>11101100<br>00010001<br>11101100<br>00010001 | 00100000<br>00100101<br>00100111<br>10100001<br>00100000<br>11101100<br>00010001<br>11101100<br>00010001 | 01000000<br>01100100<br>10000110<br>00010110<br>11000110<br>11000110<br>11110010<br>00010000<br>11101100 |

Mit der Anwendung qrCode21\_x.jar ist es möglich, einzelne Abschnitte (Modus, CCI) der Daten-Codewörter, aber auch die vollständigen Daten-Codewörter (siehe Abbildung 13, S. 9) zu berechnen.

2. Schritt: Fehlerkorrektur-Codewörter berechnen

Zur Berechnung der Fehlerkorrektur-Codewörter werden Reed-Solomon-Codes (RS-Codes) verwendet. Es handelt sich dabei um ein Vorwärtskorrekturverfahren (FEC), das die Korrektur fehlerhaft übertragener oder eingelesener Datenblöcke (8-Bit-Binärblöcke) ermöglicht. Dieses Verfahren wird z.B. auch bei der Datenspeicherung auf CDs und DVDs eingesetzt. Reed-Solomon-Codes wurden um 1960 von Irving S. Reed und Gustave Solomon am MIT Lincoln Laboratory, einer Forschungseinrichtung des Verteidigungsministeriums der Vereinigten Staaten, entwickelt und erstmals im Jahr 1977 beim Voyager-Programm der NASA eingesetzt.

Zur Berechnung der Fehlerkorrektur-Codewörter werden wiederum folgende Schritte durchlaufen:

- a) Anzahl der Fehlerkorrektur-Codewörter bestimmen,
- b) Bestimmen des Nachrichtenpolynoms,
- c) Bestimmen des Generatorpolynoms,

Tabelle 4 (Fortsetzung):  
Vorgehen beim Berechnen von Daten-Codewörtern.

d) eigentliche Berechnung der Fehlerkorrektur-Codewörter.

Da die Berechnung der Fehlerkorrektur-Codewörter für QR-Codes der Version 1 immer nach dem gleichen Schema erfolgt, werden die einzelnen Schritte im Folgenden nur am Beispiel »01234« erläutert (siehe Tabelle 5).

Die dargestellten Schritte sind zeitaufwendig und komplex. Die Anwendung qrCode21\_x.jar berechnet die Fehlerkorrektur-Codewörter für QR-Codes der Version 1. Für höhere QR-Code-Versionen stehen auf

<http://www.thonky.com/qr-code-tutorial/show-division-steps/>  
und

<http://www.pclviewer.com/rs2/calculator.html>

entsprechende Anwendungen zur Verfügung (siehe auch Abschnitt »Nützliche Internetquellen«, Seite 37).

3. Schritt: Ausgangsmatrix erstellen

Die erzeugten Daten- und Fehlerkorrektur-Codewörter werden anschließend nach dem in Abbildung 15 (nächste Seite) dargestellten Schema in das QR-Code-Raster eingetragen. Dabei ist die Anordnung und Ausrichtung der Codewörter sowie die Position der einzelnen Bits (siehe auch Abbildung 10, Seite 7) zu berücksichtigen.

Tabelle 5: Berechnen der Fehlerkorrektur-Codewörter.

| Abschnitt   | Vorgehen  | Beispiel »01234«  |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
|---|---|---|----------|-----|----------|-----|----------|----|----------|----|----------|-----|----------|-----|----------|----|----------|-----|----------|----|----------|-----|----------|-----|----------|-----|----------|----|----------|----|----------|----|----------|-----|----------|
| Fehlerkorrektur-Codewörter  | a) Die Anzahl der verwendeten Fehlerkorrektur-Codewörter (siehe Tab. 3, S.9) bestimmen.   | 17  |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
|   | b) Berechnung des Nachrichtenpolynoms <ul style="list-style-type: none"> <li>Die Daten-Codewörter werden in ihre Dezimaldarstellung überführt.</li> <li>Die Dezimalzahlen werden als Koeffizienten eines Polynoms der Gestalt: <math>C_1x^i + C_2x^{i-1} + \dots + C_ax^{i-a+1}</math> mit                             <ul style="list-style-type: none"> <li>c – Daten-CW(dezimal),</li> <li>a – Anzahl der Daten-CW,</li> <li>b – Anzahl Fehlerkorrektur-CW,</li> <li>i = a+b-1 verwendet.</li> </ul> </li> </ul>   | <table style="border: none;"> <tr><td>00010000</td><td>16</td></tr> <tr><td>00010100</td><td>20</td></tr> <tr><td>00001100</td><td>12</td></tr> <tr><td>01000100</td><td>68</td></tr> <tr><td>00000000</td><td>0</td></tr> <tr><td>11101100</td><td>236</td></tr> <tr><td>00010001</td><td>17</td></tr> <tr><td>11101100</td><td>236</td></tr> <tr><td>00010001</td><td>17</td></tr> </table> $16x^{25} + 20x^{24} + 12x^{23} + 68x^{22} + 0x^{21} + 236x^{20} + 17x^{19} + 236x^{18} + 17x^{17}$ | 00010000 | 16  | 00010100 | 20  | 00001100 | 12 | 01000100 | 68 | 00000000 | 0   | 11101100 | 236 | 00010001 | 17 | 11101100 | 236 | 00010001 | 17 |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
|   | 00010000  | 16  |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
|   | 00010100  | 20  |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 00001100  | 12  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 01000100  | 68  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 00000000  | 0   |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 11101100  | 236   |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 00010001  | 17  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 11101100  | 236   |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 00010001  | 17  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| c) Generatorpolynom bestimmen<br>Das Generatorpolynom wird mithilfe eines Galois-Feldes mit 256 Elementen erzeugt. Generatorpolynome unterscheiden sich je nach Anzahl der Fehlerkorrektur-CW. Sie sind im ISO-Standard (vgl. ISO/IEC, 2006, S. 71) festgeschrieben und können übernommen werden.   | $x^{13} + \alpha^{74}x^{12} + \alpha^{152}x^{11} + \alpha^{176}x^{10} + \alpha^{100}x^9 + \alpha^{86}x^8 + \alpha^{100}x^7 + \alpha^{106}x^6 + \alpha^{104}x^5 + \alpha^{130}x^4 + \alpha^{218}x^3 + \alpha^{206}x^2 + \alpha^{140}x + \alpha^{78}$   |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| d) eigentliche Berechnung der Fehlerkorrektur-Codewörter<br>Dies erfolgt durch Division des Nachrichtenpolynoms durch das Generatorpolynom. Die dafür notwendigen Schritte können im Kapitel »Error Correction Coding« unter <a href="http://www.thonky.com/qr-code-tutorial/part-2-error-correction/">http://www.thonky.com/qr-code-tutorial/part-2-error-correction/</a> des QR Code Tutorials (2012) nachvollzogen und durchgeführt werden. <ul style="list-style-type: none"> <li>In einem letzten Schritt werden die berechneten Koeffizienten in ihre 8-Bit-Binärdarstellung überführt und damit die Fehlerkorrektur-Codewörter generiert.</li> </ul> | <table style="border: none;"> <tr><td>54</td><td>00110110</td></tr> <tr><td>147</td><td>10010011</td></tr> <tr><td>168</td><td>10101000</td></tr> <tr><td>39</td><td>00100111</td></tr> <tr><td>22</td><td>00010110</td></tr> <tr><td>146</td><td>10010010</td></tr> <tr><td>249</td><td>11111001</td></tr> <tr><td>72</td><td>01001000</td></tr> <tr><td>165</td><td>10100101</td></tr> <tr><td>66</td><td>01000010</td></tr> <tr><td>123</td><td>01111011</td></tr> <tr><td>181</td><td>10110101</td></tr> <tr><td>141</td><td>10001101</td></tr> <tr><td>18</td><td>00010010</td></tr> <tr><td>18</td><td>00010010</td></tr> <tr><td>64</td><td>01000000</td></tr> <tr><td>183</td><td>10110111</td></tr> </table> | 54  | 00110110 | 147 | 10010011 | 168 | 10101000 | 39 | 00100111 | 22 | 00010110 | 146 | 10010010 | 249 | 11111001 | 72 | 01001000 | 165 | 10100101 | 66 | 01000010 | 123 | 01111011 | 181 | 10110101 | 141 | 10001101 | 18 | 00010010 | 18 | 00010010 | 64 | 01000000 | 183 | 10110111 |
| 54  | 00110110  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 147   | 10010011  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 168   | 10101000  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 39  | 00100111  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 22  | 00010110  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 146   | 10010010  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 249   | 11111001  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 72  | 01001000  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 165   | 10100101  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 66  | 01000010  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 123   | 01111011  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 181   | 10110101  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 141   | 10001101  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 18  | 00010010  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 18  | 00010010  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 64  | 01000000  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |
| 183   | 10110111  |   |          |     |          |     |          |    |          |    |          |     |          |     |          |    |          |     |          |    |          |     |          |     |          |     |          |    |          |    |          |    |          |     |          |

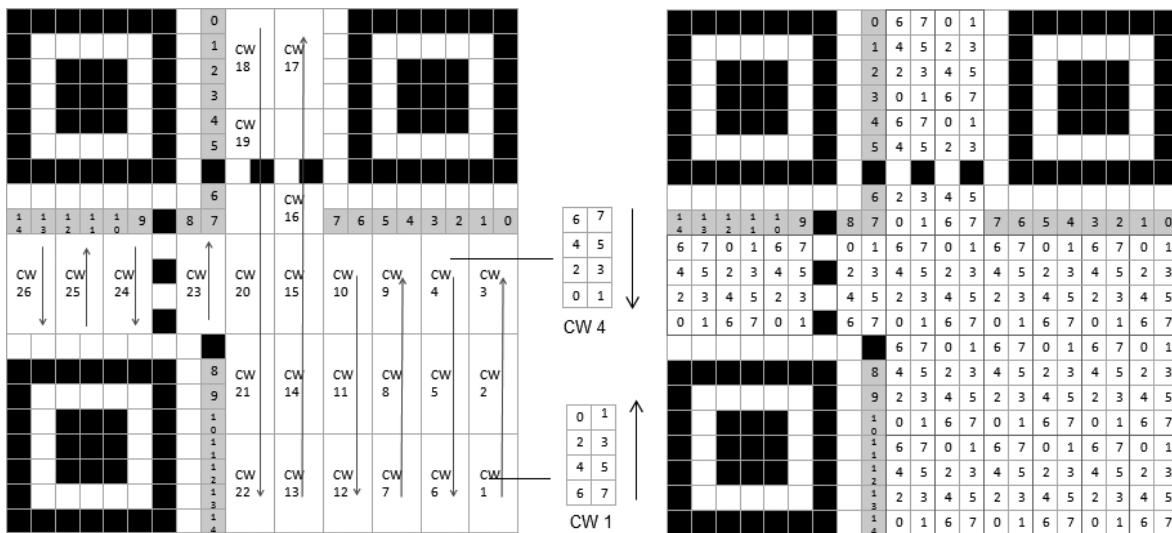


Abbildung 15: Anordnung und Ausrichtung (schmale Linien mit Richtungs-pfeilen) der Codewörter in einem QR-Code der Version 1.

| Codewort-Sequenz für das Beispiel »01234«   | Codewort-Sequenz für das Beispiel »TEST«  | Codewort-Sequenz für das Beispiel »Hallo!«  |
|---|---|---|
| Daten-Codewörter: CW 1 – CW 9<br>00010000 00010100 00001100<br>01000100 00000000 11101100<br>00010001 11101100 00010001   | Daten-Codewörter: CW 1 – CW 9<br>00100000 00100101 00100111<br>10100001 00100000 11101100<br>00010001 11101100 00010001   | Daten-Codewörter: CW 1 – CW 9<br>01000000 01100100 10000110<br>00010110 11000110 11000110<br>11110010 00010000 11101100   |
| Fehlerkorrektur-Codewörter:<br>CW 10 – CW 26<br>00110110 10010011 10101000<br>00100111 00010110 10010010<br>11111001 01001000 10100101<br>01000010 01111011 10110101<br>10001101 00010010 00010010<br>01000000 10110111 | Fehlerkorrektur-Codewörter:<br>CW 10 – CW 26<br>11010000 10010011 01111000<br>11101011 00010100 00100100<br>00001010 00101010 01001001<br>10100010 10001100 10001110<br>11011001 10100010 11001111<br>00000000 00111110 | Fehlerkorrektur-Codewörter:<br>CW 10 – CW 26<br>11101101 01110110 10100110<br>01010111 00100000 10000100<br>10100011 11101111 01001110<br>10111001 01100111 00111000<br>00011000 00100000 00111011<br>10100100 00110111 |
| Ausgangsmatrix:<br>   | Ausgangsmatrix:<br>   | Ausgangsmatrix:<br>   |

Die Anwendung qrCode21\_x.jar ermöglicht das Erzeugen solcher Ausgangsmatrizen. Ein „Klick“ auf die Schaltfläche »QR-Code erzeugen« öffnet das Fenster des QR-Builders. Die Ausgangsmatrix kann durch »Klicken« auf die einzelnen Module erzeugt oder automatisch generiert werden (siehe Abbildung 13, Seite 9).

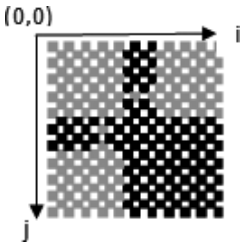

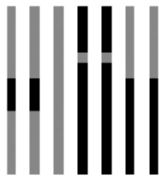
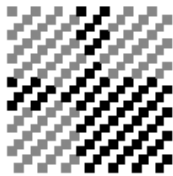
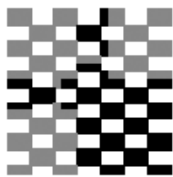
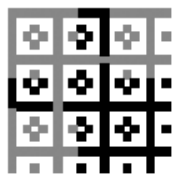


4. Schritt: Maskieren

Beim Erzeugen der Ausgangsmatrix können im Encodierungs-Bereich Modul-Sequenzen entstehen, die den Funktionsmustern ähneln. Um dies zu korrigieren und um eine gleichmäßige Verteilung von hellen und dunklen Modulen zu erhalten, schließt sich das sogenannte Maskieren an.

Beim Maskieren wird eine Maske auf alle Codewörter angewendet (Bereiche mit Finder-Pattern, Timing-Pattern oder Formatinformationen werden nicht

Tabelle 6: Eintragen der Daten- und Fehlerkorrektur-Codewörter in das QR-Code-Raster. Die Fehlerkorrektur-Codewörter der Beispiele »TEST« und »Hallo!« wurden mit qrCode21\_x.jar erstellt.

Tabelle 7:  
Masken-  
Referenzen,  
Beschreibungen  
und Darstellung.  
Die Abbildungen  
der Masken  
wurden aus  
ISO/IEC  
(2006, S.51)  
übernommen.

| Masken-Referenz | mathematische Beschreibung<br>(Das Masken-Modul mit den Koordinaten (Spalte i, Zeile j) wird dunkel gefärbt, falls der Ausdruck wahr ist.) | Masken für QR-Code<br>Version 1 (21×21)<br>(Graue Module befinden sich über den Funktionsmustern oder Formatinformationen und werden nicht maskiert.) |
|-----------------|--|---|
| 000             | $(i + j) \bmod 2 = 0$  |    |
| 001             | $j \bmod 2 = 0$  |    |
| 010             | $i \bmod 3 = 0$  |    |
| 011             | $(i + j) \bmod 3 = 0$  |    |
| 100             | $((j \text{ div } 2) + (i \text{ div } 3)) \bmod 2 = 0$  |    |
| 101             | $(i \cdot j) \bmod 2 + (i \cdot j) \bmod 3 = 0$  |    |
| 110             | $((i \cdot j) \bmod 2 + (i \cdot j) \bmod 3) \bmod 2 = 0$  |    |
| 111             | $((i \cdot j) \bmod 3 + (i + j) \bmod 2) \bmod 2 = 0$  |    |

| Modul in der Ausgangsmatrix mit den Koordinaten (i,j) | Modul in der Maske mit den Koordinaten (i,j) | Mathematische Beschreibung | Endgültiges Modul im QR-Code mit den Koordinaten (i,j) |
|---|--|----------------------------|--|
| ■   | ■  | $1 \text{ XOR } 1 = 0$     | □  |
| ■   | □  | $1 \text{ XOR } 0 = 1$     | ■  |
| □   | ■  | $0 \text{ XOR } 1 = 1$     | ■  |
| □   | □  | $0 \text{ XOR } 0 = 0$     | □  |

Tabelle 8:  
XOR-Verknüpfung beim Maskieren.

maskiert). Die mathematische Beschreibung und das Erscheinungsbild der Masken sind in der Tabelle 7 (vorige Seite) dargestellt.

Alle Module der Maske, die die Bedingungen in Spalte 2 erfüllen, werden dunkel gefärbt. Anschließend wird eine Maske auf die Ausgangsmatrix angewendet. Dazu werden die Werte (Modulausprägungen) aller Daten- und Fehlerkorrektur-Module mit den Werten der koordinatengleichen Module der Maske XOR-verknüpft (siehe Tabelle 8).

Das heißt,

- ▷ ist ein Maskenmodul dunkel, so wird die Ausprägung des koordinatengleichen Moduls in der Ausgangsmatrix invertiert,
- ▷ ist ein Maskenmodul hell, so wird die Ausprägung des koordinatengleichen Moduls in der Ausgangsmatrix beibehalten.

Die so veränderte Ausgangsmatrix enthält die endgültigen Daten- und Fehlerkorrektur-Module des QR-Codes.

Anschaulich erklärt, wird beim Maskieren die gewünschte Maske über die Ausgangsmatrix gelegt und die Ausgangsmatrix nach dem oben beschriebenen Verfahren verändert. Die Anwendung qrCode21\_x.jar veranschaulicht diese Vorgehensweise indem eine rote Maske eingeblendet wird (siehe Abbildung 16).

*Erklärung des Maskierens mit qrCode21\_x.jar:*

- ▷ Die roten Module (in der Abbildung 16 hellgrauen Module) stehen für Module, die nur in der Maske dunkel gefärbt sind.
- ▷ Die dunkelroten Module (in der Abbildung 16 dunkelgrauen Module) stehen für Module, die sowohl in der Maske als auch in der Ausgangsmatrix dunkel gefärbt sind.
- ▷ Die weißen und schwarzen Module stehen für Module, die in der Maske hell gefärbt sind.

Das heißt, nur die roten und dunkelroten Module müssen invertiert werden. Dies kann durch »Klicken« auf das jeweilige Modul erfolgen oder automatisch durchgeführt werden (siehe Tabelle 9, nächste Seite).

Tatsächlich werden nach diesem Prinzip alle acht Masken auf eine Ausgangsmatrix angewendet und die entstandenen QR-Codes bewertet. Nach den Kriterien:

Abbildung 16:  
Maskieren mit qrCode21\_x.jar  
am Beispiel »Hallo!«.

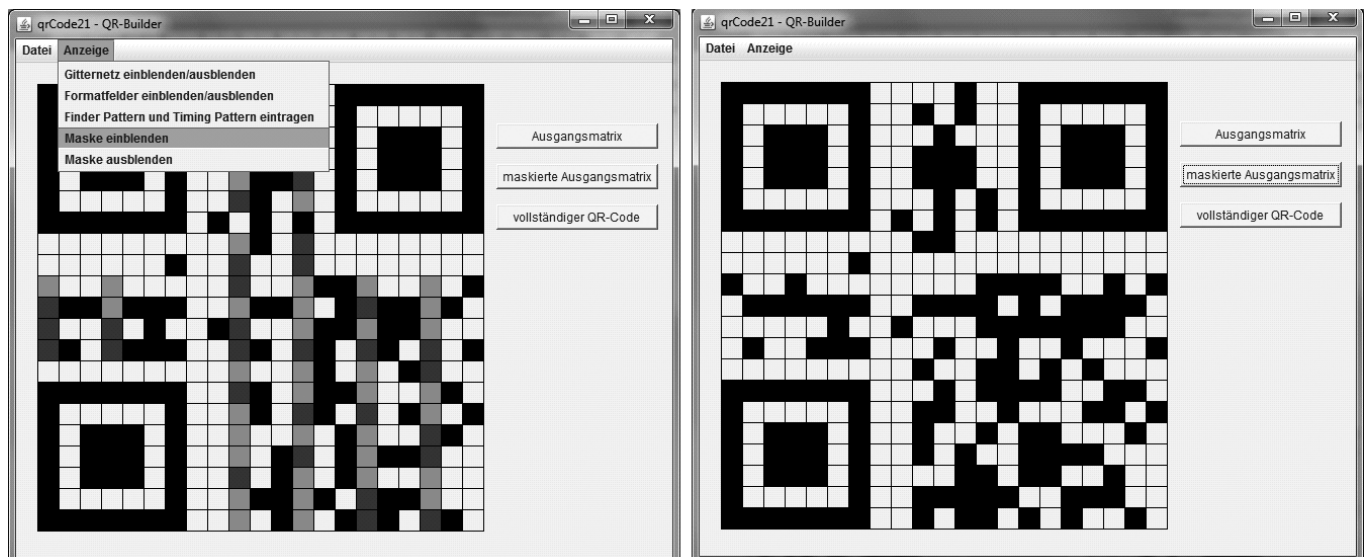


Tabelle 9: Maskieren.



- ▷ hintereinander liegende Module in Spalte/Zeile haben die gleiche Farbe,
  - ▷ Block mit Modulen in gleicher Farbe,
  - ▷ 1:1:3:1:1 (Finder-Pattern) Muster in Spalte/Zeile sowie
  - ▷ Verhältnis helle zu dunkle Module im gesamten Symbol
- werden Strafpunkte berechnet (genauere Hinweise zur Berechnung befinden sich in ISO/IEC, 2006). Die Maske mit den wenigsten Strafpunkten wird verwendet.

Auch ohne den letzten Bewertungsschritt erhält man nach dem Maskieren mit einer bestimmten Maske einen decodierbaren QR-Code. Die Anwendung qrCode21\_x.jar gibt lediglich die Möglichkeit, die maskierten QR-Codes zu vergleichen und anschließend eine Wahl zu treffen.

5. Schritt: *Formatinformationen hinzufügen*

Zuletzt müssen die Formatinformationen eingetragen werden. Sie sind in 15 Bit zusammengefasst (siehe Abbildung 17).

Das Fehlerkorrekturniveau umfasst zwei Bit (L-01, M-00, Q-11, H-10) und die Masken-Referenz drei Bit (siehe auch Tabelle 7, Seite 14). Die Fehlerkorrektur der Formatinformationen umfasst 10 Bit und wird mithilfe der BCH-Codierung berechnet. (Die Abkürzung BCH setzt sich aus den Anfangsbuchstaben der Nachnamen von R. D. Bose, D. K. Ray-Chaudhuri und A. Hocquenghem zusammen, die diese zyklisch fehlerkorrigierende Codierung Ende der 1970er-Jahre entwickelt haben). Die so zusammengesetzte Bit-Sequenz wird maskiert und anschließend in das Raster eingetragen. Die endgültigen Formatinformationen können aus Tabelle 10 (nächste Seite) abgelesen werden.

Die Formatinformationen werden an die dafür vorgesehenen Positionen (siehe Abbildung 9, Seite 7) zweimal in das QR-Code-Raster eingetragen (Die Zahlen 0 bis 14 repräsentieren die Position eines Bits innerhalb der Formatinformationen. Die 14 steht für das Bit ganz links und die 0 für das Bit ganz rechts).

Abbildung 17: Zusammensetzung der Formatinformationen.

|                       |                 |                 |
|-----------------------|-----------------|-----------------|
| Fehlerkorrekturniveau | Masken-Referenz | Fehlerkorrektur |
|-----------------------|-----------------|-----------------|



Tabelle 10: Formatinformationen.

| Fehlerkorrektur-niveau | Masken-Referenz | Formatinformationen | Fehlerkorrektur-niveau | Masken-Referenz | Formatinformationen |
|------------------------|-----------------|---------------------|------------------------|-----------------|---------------------|
| L                      | 000             | 111011111000100     | Q                      | 000             | 011010101011111     |
| L                      | 001             | 111001011110011     | Q                      | 001             | 011000001101000     |
| L                      | 010             | 111110110101010     | Q                      | 010             | 011111100110001     |
| L                      | 011             | 111100010011101     | Q                      | 011             | 011101000000110     |
| L                      | 100             | 110011000101111     | Q                      | 100             | 010010010110100     |
| L                      | 101             | 110001100011000     | Q                      | 101             | 010000110000011     |
| L                      | 110             | 110110001000001     | Q                      | 110             | 010111011011010     |
| L                      | 111             | 110100101110110     | Q                      | 111             | 010101111101101     |
| M                      | 000             | 101010000010010     | H                      | 000             | 001011010001001     |
| M                      | 001             | 101000100100101     | H                      | 001             | 001001110111110     |
| M                      | 010             | 101111001111100     | H                      | 010             | 001110011100111     |
| M                      | 011             | 101101101001011     | H                      | 011             | 001100111010000     |
| M                      | 100             | 100010111111001     | H                      | 100             | 000011101100010     |
| M                      | 101             | 100000011001110     | H                      | 101             | 000001001010101     |
| M                      | 110             | 100111110010111     | H                      | 110             | 000110100001100     |
| M                      | 111             | 100101010100000     | H                      | 111             | 000100000111011     |

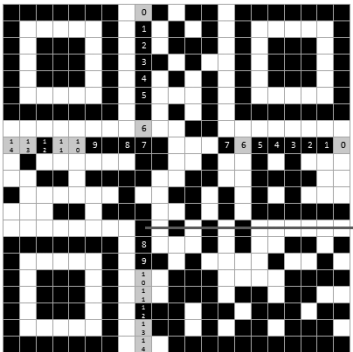
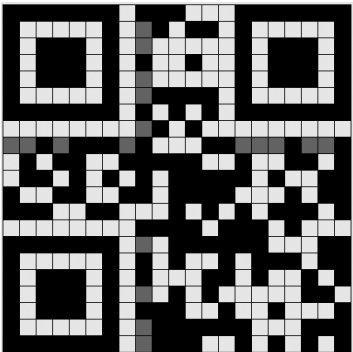

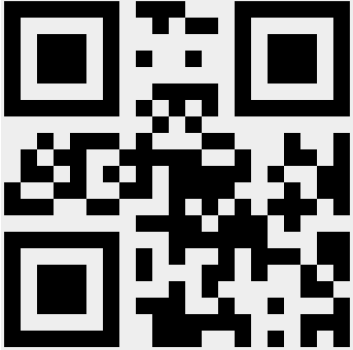
| Formatinformationen für Beispiel »01234«  | Formatinformationen für Beispiel »TEST«  |
|---|--|
| Niveau H, Maske 1(001)<br>001001110111110   | Niveau H, Maske 0(000)<br>001011010001001  |
| <p>Eintragen in das Raster (zur Veranschaulichung hier grau hinterlegt und nummeriert)</p>  <p>Dieses Modul wird immer dunkel gefärbt.</p> | <p>Eintragen in das Raster (zur Veranschaulichung hier grau hinterlegt)</p>  |
| <p>Endgültiger QR-Code</p>   | <p>Endgültiger QR-Code</p>   |

Tabelle 11: Formatinformationen für die Beispiele »01234« und »TEST«.



Die Decodierung ist im Wesentlichen die Umkehrung der Encodierung. Die Decodierungs-Schritte werden deshalb nur für QR-Codes mit dem Modus *Alphanumeric* erläutert. Die Digitalisierung der QR-Code-Grafik und die Ausgabe der Daten werden nicht näher betrachtet.

Alle Schritte werden so erläutert, dass sie ohne Nutzung eines Informatiksystems (also per Hand) nachvollzogen werden können. Zur besseren Veranschaulichung kann zusätzlich wieder die Anwendung `qrCode21_x.jar` verwendet werden.

**Die Decodierung – anschaulich erklärt**

Das Decodieren eines QR-Codes wird durch das Einzeichnen eines Gitternetzes vereinfacht. Der QR-Builder der Anwendung `qrCode21_x.jar` bietet die Möglichkeiten, ein Bild zu laden sowie Hilfsfelder und Hilfslinien einzuzeichnen. `qrCode21_x.jar` kann ausschließlich die von dieser Anwendung erzeugten QR-Codes einlesen. Für andere QR-Code-Versionen kann beispielsweise die Freeware `bcTester 4.9` verwendet werden. Der entstandene modularisierte QR-Code (siehe Abbildung 19, vorige Seite) wird als Grundlage für die folgenden Schritte genutzt.

*1. Schritt: Formatinformationen auslesen*

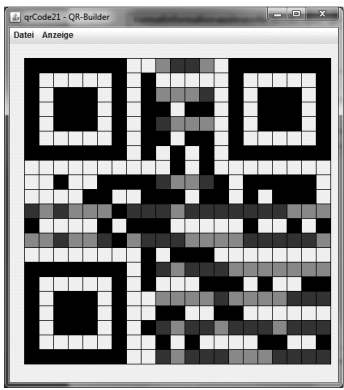
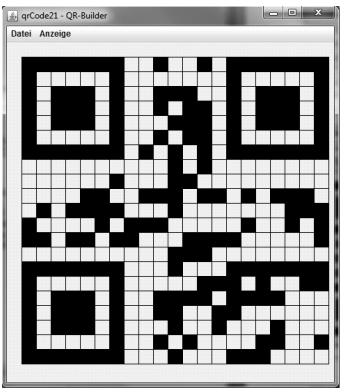
Um den verwendeten Modus und die verwendete Maske zu ermitteln, müssen zuerst die Formatinformationen ausgelesen werden.

Das Auslesen der Formatinformationen erfolgt im grau hinterlegten Bereich (die Zahlen 0 bis 14 repräsentieren die Position eines Bits innerhalb der Formatinformation. Die 14 steht für das Bit ganz links und die 0 für das Bit ganz rechts). Aus der Tabelle 10, Seite 17, kann anschließend das Fehlerkorrekturniveau und die Masken-Referenz abgelesen werden.

| Formatinformationen auslesen für den Beispiel-QR-Code |
|---|
| 001001110111110                                       |
| Fehlerkorrekturniveau H                               |
| Masken-Referenz 001                                   |

*2. Schritt: Maskieren*

Auf den modularisierten QR-Code wird die angegebene Maske angewendet, um die Ausgangsmatrix zu erzeugen.

| Maskieren für den Beispiel-QR-Code  |   |
|---|---|
| Maskieren mit Maske 1 (001)   | QR-Code nach dem Maskieren  |
|  |  |

Um mit `qrCode21_x.jar` die Ausgangsmatrix zu erzeugen, müssen im QR-Builder alle rot und dunkelrot markierten Module (beim Beispiel-QR-Code hier hellgrau und dunkelgrau dargestellt) durch »Klicken« invertiert werden.

*3. Schritt: Fehlerkorrektur-Codewörter auslesen und anwenden*

Aus dem Fehlerkorrekturniveau und der verwendeten Version kann auf die Anzahl der Fehlerkorrektur-Codewörter geschlossen werden (siehe Tabelle 3, Seite 9). Gleichzeitig werden die Anordnung und die Ausrichtung der Codewörter festgelegt (siehe Abbildung 15, Seite 13).

Die Position des ersten Fehlerkorrektur-Codewortes ist von der Version des QR-Codes und dem Fehlerkorrekturniveau abhängig.

**Fehlerkorrektur-Codewörter auslesen für den Beispiel-QR-Code**

9 Daten-Codewörter und 17 Fehlerkorrektur-Codewörter.  
Das erste Fehlerkorrektur-Codewort ist CW 10.

```
01000011 00101110 11010010 00010000 11100011 11000010 01101111
11010010 01001101 10011010 11100100 10100011 11100110 01110010
11111001 10001000 00100111
```

Um eventuelle Fehler zu erkennen und zu beheben, erfolgt nun das sogenannte RS-Decodieren (nach Reed und Salomon benannt; siehe Seite 11). In diesem Prozess werden die Codewörter mithilfe eines endlichen Körpers (engl.: *Galois field*; benannt nach Évariste Galois, einem französischen Mathematiker) in ein Gleichungssystem übertragen. Anschließend werden über zahlreiche Schritte die Koordinaten der Fehlerpositionen berechnet und die Fehler korrigiert. (Der Vorgang des RS-Decodierens ist komplex und rechenintensiv. Genauere Erläuterungen sind in ISO/IEC, 2006, zu finden.) Wurden alle fehlerhaften Module beseitigt, kann zum nächsten Schritt übergegangen werden.

*4. Schritt: Daten-Codewörter auslesen und decodieren*

Die Position des ersten Daten-Codeworts ist festgelegt (unten rechts). Die Anzahl der Daten-Codewörter ist von der Version des QR-Codes und dem Fehlerkorrekturniveau abhängig.

**Daten-Codewörter auslesen für den Beispiel-QR-Code**

9 Daten-Codewörter CW 1 bis CW 9

```
00100000 00111100 10101101 11101000 00110001 00010100 11100000
11101100 00010001
```

Unter Berücksichtigung des Aufbaus einer Codewort-Sequenz (vgl. den Abschnitt »Von der Information zum QR-Code: Encodierung«, Seite 9 ff.) werden die Daten wie in Tabelle 12 (siehe nächste Seite) decodiert.



## 2

## Die Unterrichtsreihe »Von der Information zum QR-Code und wieder zurück«

Erzähle mir, und ich vergesse.  
Zeige mir, und ich erinnere.  
Lass mich tun, und ich verstehe.  
*Konfuzius*

Konfuzius' Worten folgend, werden Schülerinnen und Schüler im Verlauf der Unterrichtsreihe zum eigenständigen Durchführen der QR-Encodierung und -Decodierung befähigt. Mit dieser Handreichung wurde Arbeitsmaterial entwickelt, das das selbstständige Erarbeiten der einzelnen Schritte ermöglicht. Die Ergebnisüberprüfung erfolgt mithilfe der Anwendung qrCode21.jar (siehe Abbildung 26, Seite 25) bzw. durch Einscannen mithilfe eines Smartphones oder einer Decodierungs-Software. Die Lehrperson kann sich in vielen Phasen der Unterrichtsreihe zurücknehmen und sich mit einzelnen Schülern oder Schülergruppen auseinandersetzen.

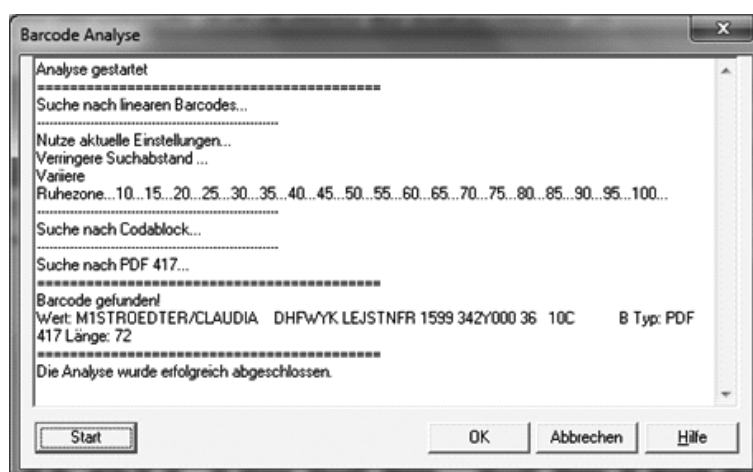
Das Arbeitsmaterial, die Anwendung qrCode21.jar und das JAVA-Package qrCode21 wurden von der Autorin entwickelt und stehen im LOG-IN-Service zum Herunterladen zur Verfügung. Die Anwendung qrCode21.jar wurde für Schülerinnen und Schüler entwickelt und stellt eine reduzierte Version der Anwendung qrCode21\_x.jar dar. Sie ermöglicht das Überprüfen von Daten-Codewörtern, das Berechnen der Fehlerkorrektur-Codewörter und stellt eine reduzierte Version des QR-Builders (ohne automatisches Eintragen) zur Verfügung.

### Erste Stunde: Die 2-D-Codes

Zu Beginn der Unterrichtseinheit decodieren die Schülerinnen und Schüler unterschiedliche 2-D-Codes. Diese werden von der Lehrkraft zur Verfügung gestellt oder von den Lernenden mitgebracht. Da Smartphones meist nur wenige 2-D-Code-Typen decodieren können, sollte zur Decodierung eine Decodierungs-Software verwendet werden (z. B. bcTester 4.9; siehe Abbildung 20).

Während dieser Unterrichtsphase sollte auffallen, dass nicht alle ausgegebenen Daten die ursprüngliche Information darstellen. So ist beispielsweise bei den Online- und Handy-Tickets der Deutschen Bahn und den Frankierungscodes der Deutschen Post das Auslesen der personenbezogenen Daten nicht möglich, und die ursprüngliche Information wird daher nur teilweise oder gar nicht ausgegeben (siehe Tabelle 16, Seite 31). Lediglich autorisierte Personen haben Zugriff auf diese Daten, sodass Aspekte des Datenschutzes ebenfalls angesprochen werden können.

Abbildung 20:  
2-D-Code-Analyse mit bcTester 4.9.



verändert nach: AKBSI, 2008, S. 23

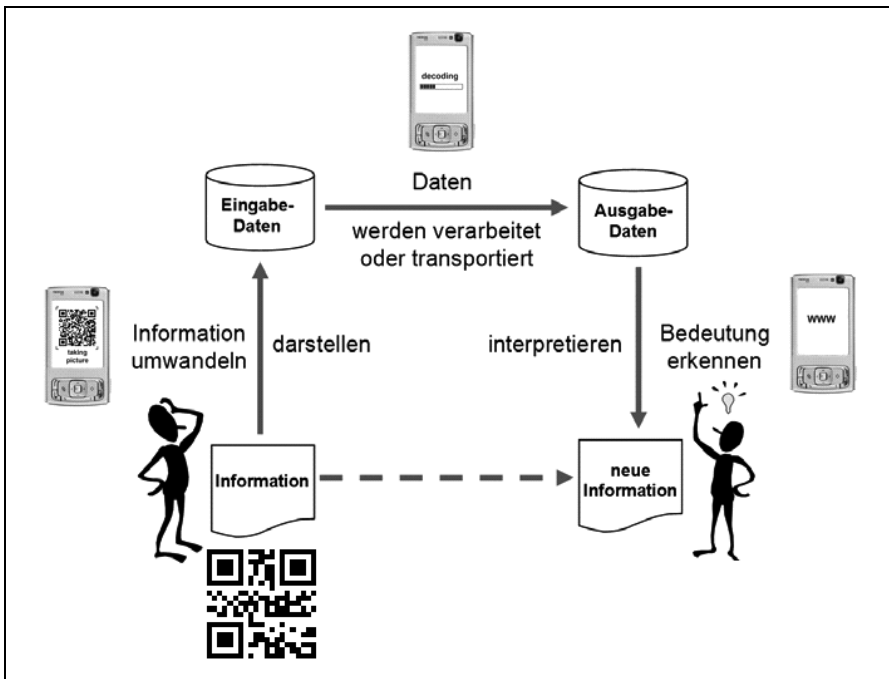
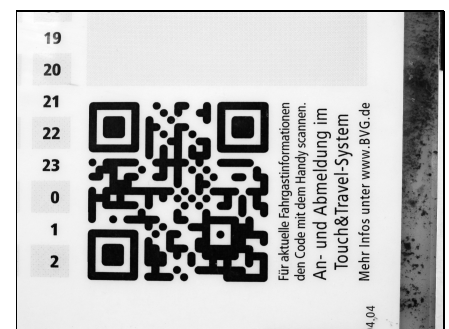


Abbildung 21 (links): Mobile-Tagging und der Zusammenhang zwischen Information und Daten.

Abbildung 22 (unten): Beispiele von QR-Codes aus dem Alltag.

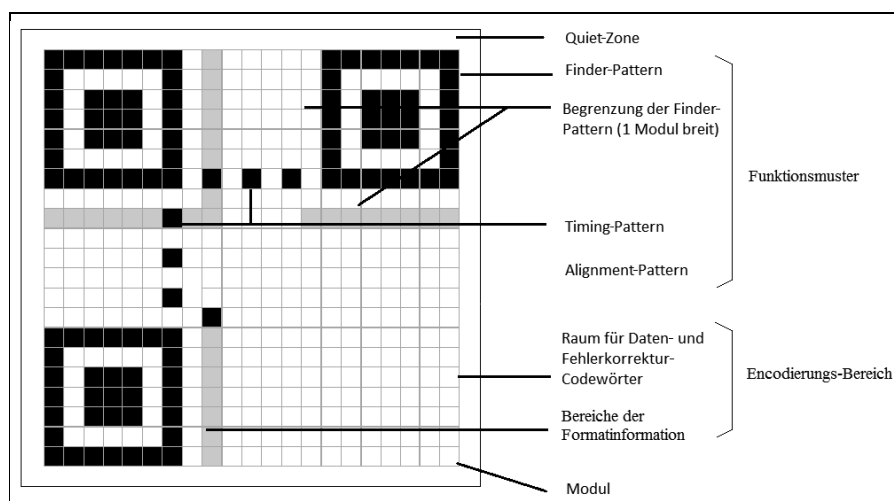


Im Rahmen dieser Unterrichtsstunde lernen die Schülerinnen und Schüler unterschiedliche 2-D-Code-Typen und deren Anwendungsbereiche kennen. In einem abschließenden Unterrichtsgespräch kann eine Systematisierung der 2-D-Codes in Stapel-Codes und Matrix-Codes erfolgen (siehe auch Tabelle 16, Seite 31). Zudem kann am Beispiel des Mobile-Tagging (siehe Abbildung 2, Seite 4) der Zusammenhang zwischen Information und Daten erarbeitet werden (siehe Abbildung 21).

### Zweite Stunde: Die QR-Codes

In der nächsten Unterrichtsphase werden die QR-Codes intensiver betrachtet. Dazu bringt jede Schülerin bzw. jeder Schüler einen QR-Code mit (siehe auch Abbildung 22). Zusätzlich sollten mehrere QR-Codes der Version 1 zur Verfügung stehen. Zur Erstellung dieser QR-Codes kann die Anwendung qrCode21\_x.jar verwendet werden.

In kleinen Gruppen wird das Erscheinungsbild der QR-Codes verglichen. Die Schülerinnen und Schüler finden selbstständig erste Funktionsmuster (Finder-Pattern, Alignment-Pattern) und erkennen die einzelnen Module im Encodierungs-Bereich. Im anschließenden Unterrichtsgespräch kann ein QR-

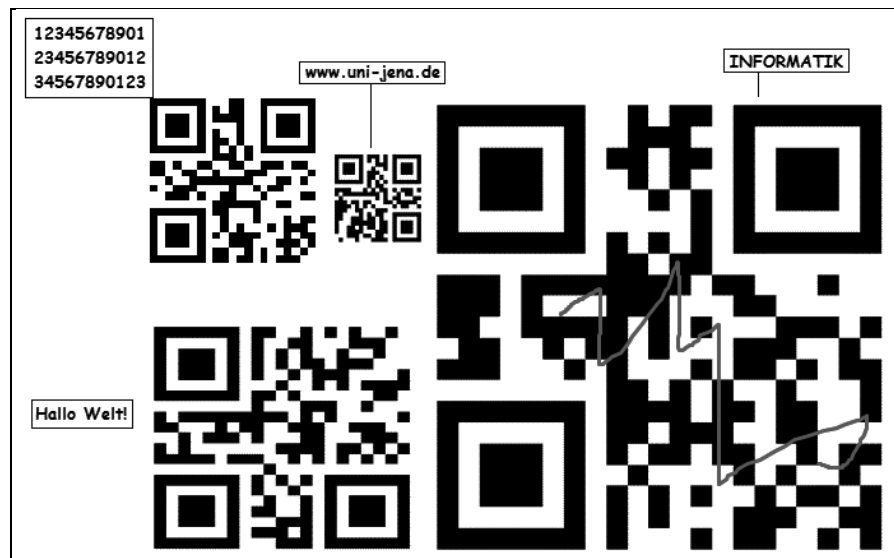


einander anzutreten.  
Fehlerfrei, so viel ist sicher, wird der Atlas des deutsch-amerikanischen Teams den Parours nicht absolvieren. Es bleibt die Hoffnung, dass möglichst viele der Konkurrenten noch schlimmer straucheln werden.  
JOHANN GROLLE

**Video: Die Rettungsroboter in Aktion**  
spiegel.de/app512013rettungsroboter oder in der App DER SPIEGEL

Abbildung 23: QR-Code-Modell für den Unterrichtseinsatz.

Abbildung 24: QR-Codes der Version 1.



Code-Modell erarbeitet und die Bedeutung der einzelnen Bestandteile geklärt werden (siehe Abbildung 23, vorige Seite).

Um den Encodierungs-Bereich genauer zu untersuchen, schätzen die Schülerinnen und Schüler die Anzahl der Module in ihrem QR-Code. Die Ergebnisse werden zusammengetragen und mit der tatsächlichen Größe der QR-Codes verglichen. Die Schülerinnen und Schüler erkennen, dass die Anzahl der Module unabhängig von der tatsächlichen Größe des QR-Codes ist. Ausgehend von dieser Erkenntnis können die QR-Code-Versionen vorgestellt und erste Überlegungen zur Datenkapazität getroffen werden.

Nun werden einige QR-Codes decodiert und die Toleranz gegenüber Rotation, Verzerrung und Zerstörung getestet (siehe Abbildung 24). Die Schülerinnen und Schüler finden Erklärungsansätze für diese Fehlerkorrekturen und erkennen die Bedeutung von Fehlerkorrektur-Sequenzen. Abschließend können die einzelnen Fehlerkorrekturniveaus und Modi vorgestellt werden (siehe Tabelle 2, Seite 8 und Tabelle 3 Version 1, Seite 9).

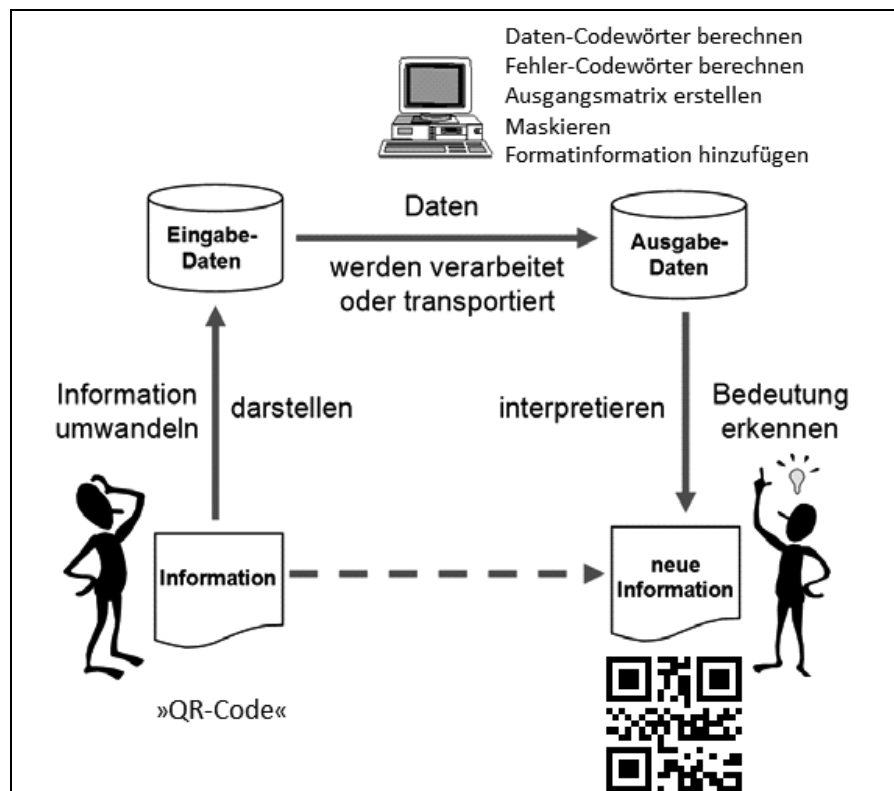


Abbildung 25: Übersicht über die Encodierungs-Schritte.

verändert nach: AKBSI, 2008, S. 23



Im täglichen Leben werden die Lernenden mit verschiedenen QR-Code-Versionen konfrontiert. Diese sollen sie in dieser Unterrichtsstunde kennen und einordnen lernen. Für die folgenden Unterrichtsstunden ist die Beschränkung auf QR-Codes der Version 1 (ohne Alignment-Pattern) empfehlenswert.

**Dritte und vierte Stunde:  
Von der Information zum QR-Code und wieder zurück**

In dieser Unterrichtsphase werden QR-Codes selbstständig erzeugt und anschließend mithilfe eines Smartphones decodiert.

Zu Beginn werden die einzelnen Schritte der Encodierung vorgestellt (siehe Abbildung 25, vorige Seite), sodass jeder Schüler einen Überblick über den gesamten Vorgang der Encodierung erhält.

Die Erarbeitung der einzelnen Encodierungs-Schritte kann mithilfe der kooperativen Lernform *Gruppenpuzzle* erfolgen. In der ersten Phase werden die Schülerinnen und Schüler entsprechend ihrer individuellen Lernvoraussetzungen auf die Expertengruppen verteilt.

- ▷ Expertengruppe 1: Daten-Codewörter berechnen.
- ▷ Expertengruppe 2: Ausgangsmatrix erstellen.
- ▷ Expertengruppe 3: Maskieren und Formatinformationen hinzufügen.

Da im schulischen Kontext keine detaillierte Betrachtung der Reed-Solomon-Codierung möglich ist, gibt es keine Expertengruppe zum zweiten Encodierungs-Schritt. Die Fehlerkorrektur-Codewörter werden mithilfe der Anwendung qrCode21.jar erzeugt.

In den Expertengruppen erarbeiten die Schülerinnen und Schüler ihren Encodierungs-Schritt. Dazu erhalten sie Arbeitsblätter, die vereinfacht die Anleitungen aus Abschnitt »Die Encodierung – anschaulich erklärt« (Beispiel »Hallo!«, Byte-Mode) beinhalten, und arbeiten mit der Anwendung qrCode21.jar (siehe Abbildung 26).

In der zweiten Phase des Gruppenpuzzles werden die Schülerinnen und Schüler auf Stammgruppen verteilt. In den Stammgruppen muss mindestens ein Mitglied aus jeder Expertengruppe sein. Jede Gruppe hat nun die Aufgabe, einen QR-Code mit dem Modus *Byte* zu erstellen. Das Prüfen der Daten-Codewörter, die Berechnung der Fehlerkorrektur-Codewörter und das Eintragen in das QR-Code-Raster erfolgt wieder mithilfe der Anwendung qrCode21.jar.

Ist der QR-Code generiert, kann er mithilfe eines Smartphones direkt vom Monitor decodiert werden. Es besteht auch die Möglichkeit, die Grafik zu speichern, sodass sie mithilfe des Computers decodiert oder ausgedruckt werden kann.

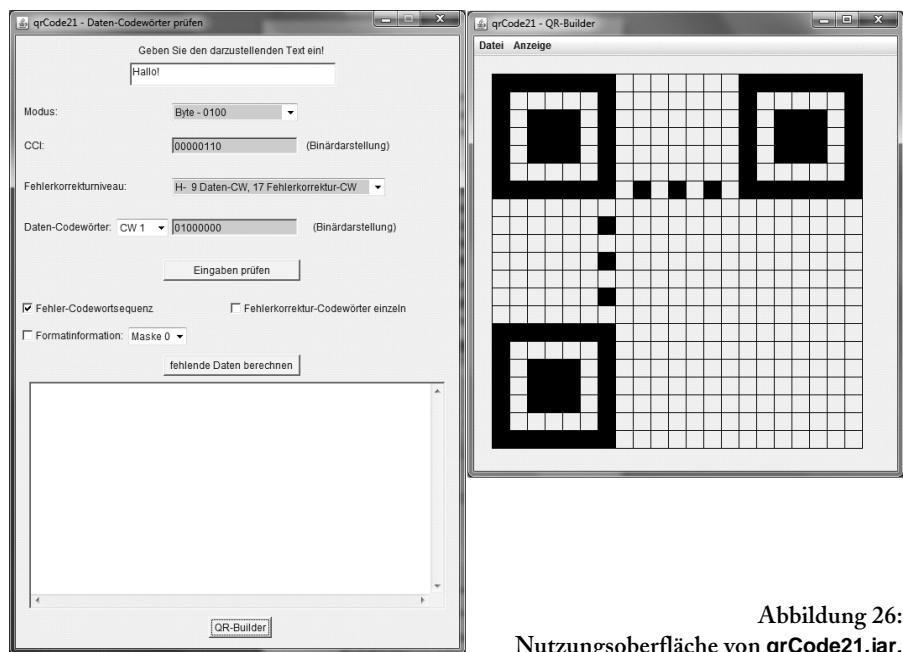
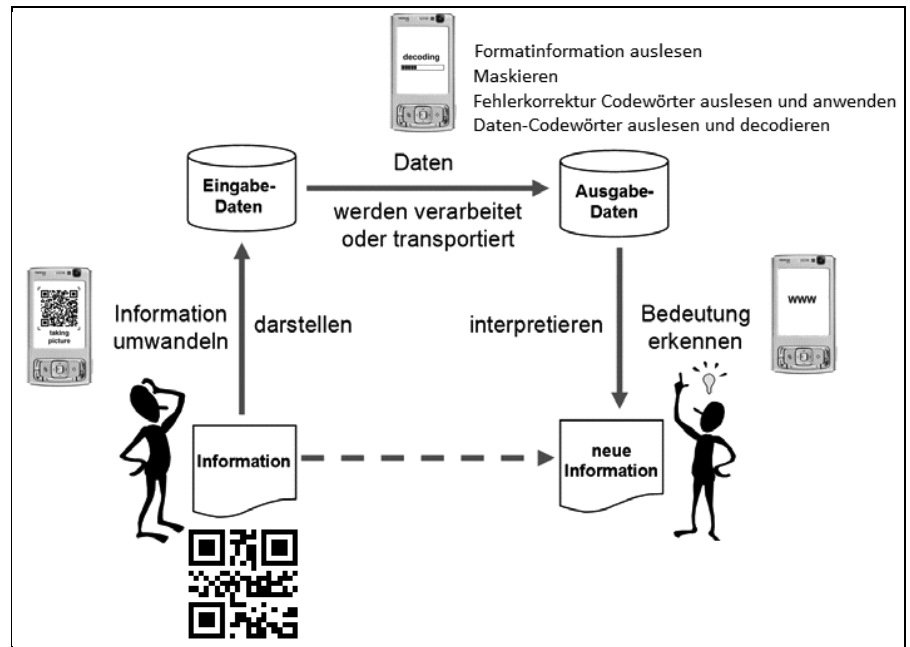


Abbildung 26:  
Nutzungsoberfläche von qrCode21.jar.

Abbildung 27: Übersicht über die Decodierungs-Schritte.



verändert nach: AKBSI, 2008, S. 23

Nach Beendigung des Gruppenpuzzles kann darauf verwiesen werden, dass alle Schritte und Berechnungen nur zum besseren Verständnis in einem QR-Code-Raster durchgeführt wurden. In einem Informatiksystem werden alle Schritte auf einem zweidimensionalen Boole'schen Feld (engl.: *Boolean array*) realisiert. Der letzte Schritt ist die Generierung des QR-Codes, bei dem das zweidimensionale Array in eine zweidimensionale Grafik umgewandelt wird.

Die Decodierung der entstandenen QR-Codes kann im Klassenverband besprochen werden (siehe Abbildung 27). Die Decodierungs-Schritte werden von den Schülerinnen und Schülern erläutert und mithilfe eines Smartphones realisiert.

Abschließend könnte der Text »QR-Code infiziert Android-Smartphones« (siehe Abbildung 32, Seite 32) gelesen werden. Somit wäre es möglich, den Schülerinnen und Schülern auch die Gefahren im Umgang mit einem Smartphone zu verdeutlichen und sie über Maßnahmen zum Schutz vor Schadsoftware diskutieren zu lassen.

Die vorgestellten Unterrichtsstunden haben einen Einblick in das Thema QR-Codes ermöglicht. Die Schülerinnen und Schüler sind in der Lage, den Aufbau eines QR-Codes zu erläutern und die Encodierungs- und Decodierungs-Schritte nachzuvollziehen. Es sind somit die Voraussetzungen dafür geschaffen, QR-Codes auch im Bereich *Algorithmen* der Bildungsstandards (vgl. AKBSI, 2008) in den Informatikunterricht einzubeziehen. Dies kann im An-

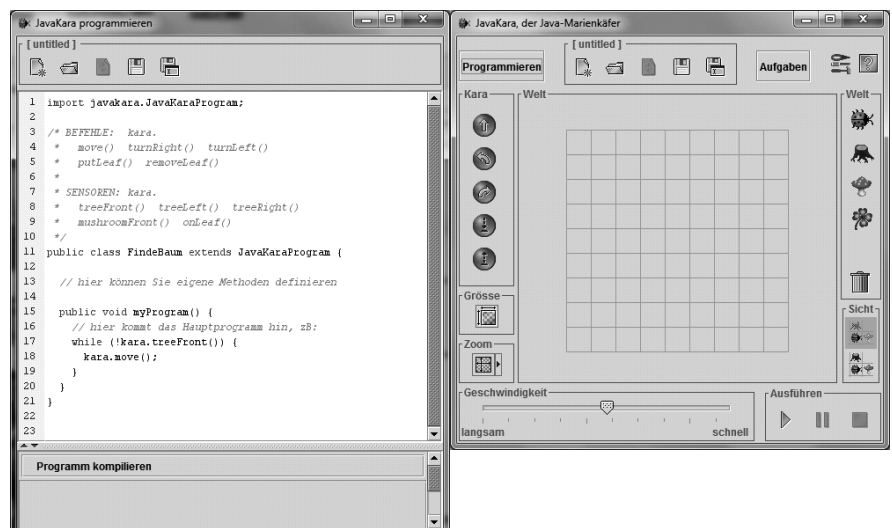
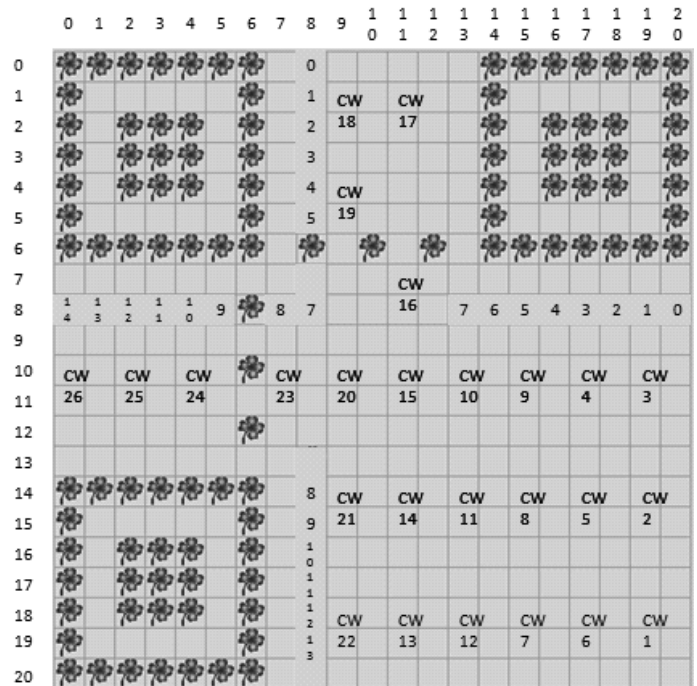
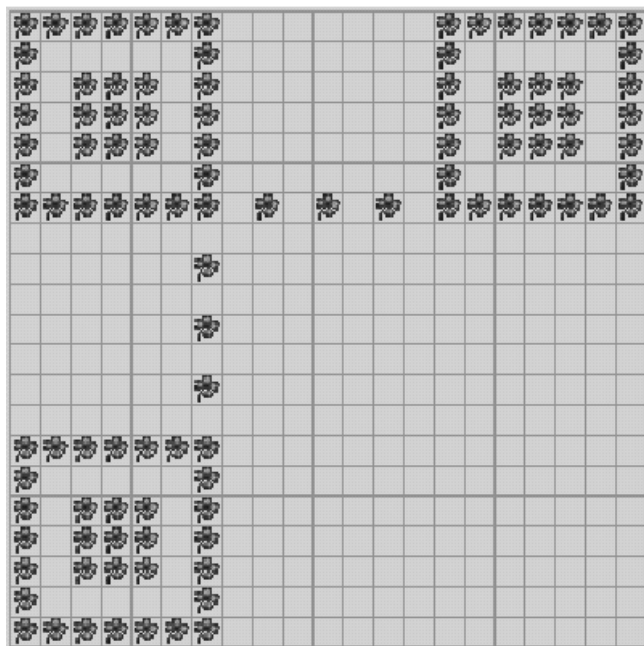


Abbildung 28: Programmierumgebung von JAVAKARA.



schluss an die bereits dargestellten Unterrichtsstunden erfolgen oder zu einem späteren Zeitpunkt wieder aufgegriffen werden.

### Auch KARA kann QR-Codes

Führt man die Prozesse *Encodierung* und *Decodierung* wie in der vorgestellten Unterrichtssequenz durch, benötigt man viel Zeit. Für den alltäglichen Gebrauch wäre dieses Vorgehen ineffizient. Deshalb werden das Encodieren und das Decodieren an Informatiksysteme delegiert und mit ihnen realisiert. Um solche Informatiksysteme zu erzeugen, müssen die jeweiligen Algorithmen verstanden und in eine Programmiersprache überführt werden. Das Programmieren von grafischen Oberflächen ist für Schülerinnen und Schüler der Sekundarstufe I nur schwer zu realisieren. Aus diesem Grund sollte eine Programmierumgebung gewählt werden, »in der sich Schülerinnen und Schüler gut aufgehoben fühlen, die sie überblicken, in der sie lernen und arbeiten und ihre Kreativität zur Entfaltung bringen können« (Fothe, 2011/2012, S. 40).

Die freie Programmierumgebung JAVAKARA (vgl. Reichert, R. u. a., 2007) erfüllt diese Forderungen und kann zur Einführung in die Programmierung, aber auch für fortgeschrittene Programmierer genutzt werden (siehe Abbildung 28, vorige Seite). Die KARA-Welt ist ein Raster aus quadratischen Flächen, die von KARA – dem programmierbaren Marienkäfer – durchlaufen werden kann. KARA kann unter anderem Kleeblätter auf einem Feld platzieren, diese einsammeln und erkennen ob auf einem Feld ein Kleeblatt liegt oder nicht.

Somit liefert die KARA-Welt alle Voraussetzungen für das Erzeugen und Lesen von QR-Codes. Ein KARA-QR-Code der Version 1 ist in der Abbildung 29 dargestellt.

Unter Verwendung des JAVA-Packages `qrcode21` ist es in JAVAKARA möglich QR-Codes der Version 1 zu generieren. Dazu muss das Package importiert und ein Objekt der Klasse `QRKara` erzeugt werden (siehe Abbildung 30).

Abbildung 29: Ein KARA-QR-Code der Version 1 und ein KARA-QR-Code-Raster.

Abbildung 30: Quelltext zum Erzeugen eines KARA-QR-Codes.

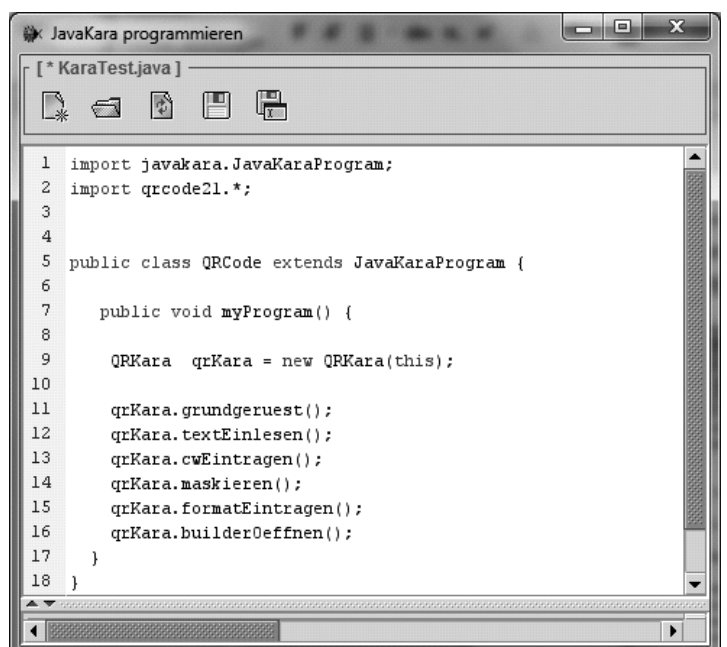


Tabelle13  
Eine Auswahl der QRKara-Methoden.

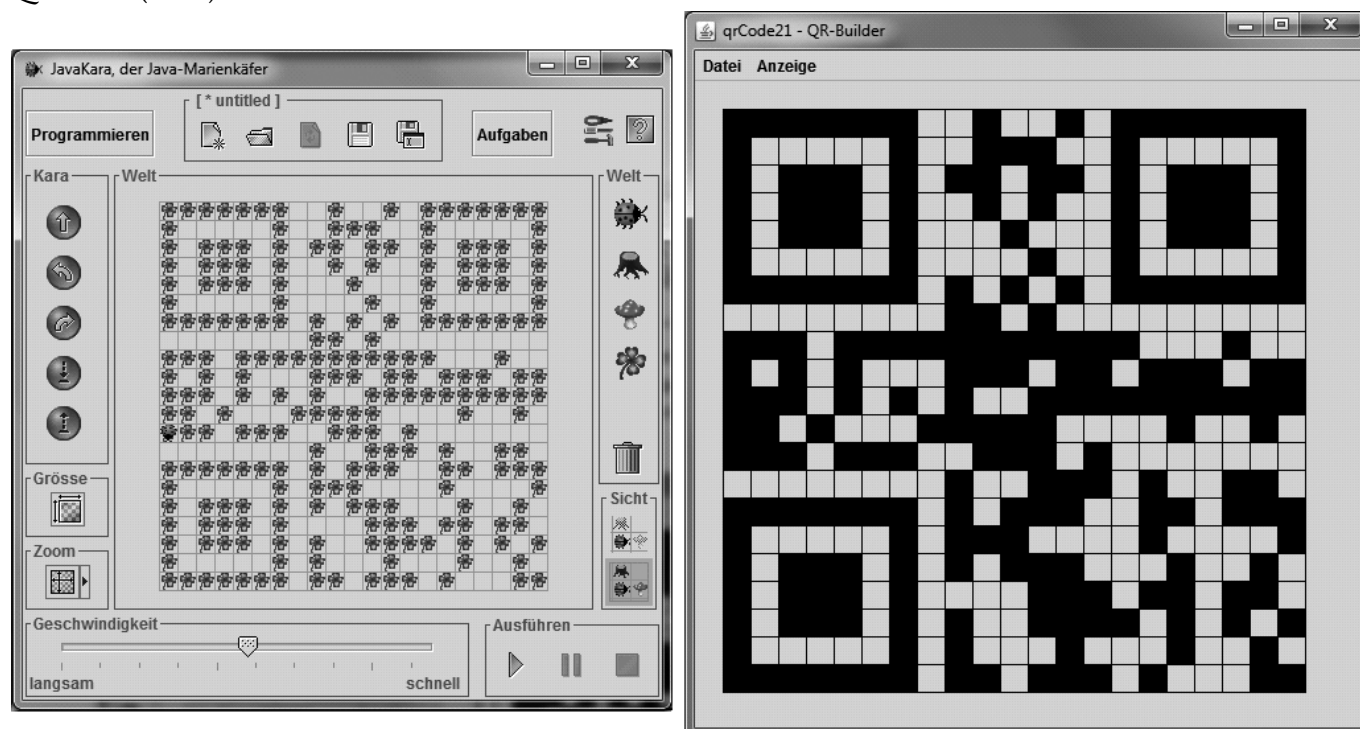
| Methode                | Beschreibung  |
|------------------------|---|
| erzeugeGrundgeruest(); | Erzeugt eine KARA-Welt der Größe 21×21 und trägt die Finder-Pattern und Timing-Pattern ein. |
| textEinlesen();        | Öffnet ein Eingabefenster zum Einlesen der darzustellenden Information.                     |
| cwEintragen();         | KARA trägt alle Codewörter ein.   |
| cwEintragen(int x);    | KARA trägt nur das x-te Codewort ein ( $x \in \{1;2;\dots;26\}$ ).                          |
| maskieren();           | KARA maskiert alle Codewörter mit der Maske 0.  |
| maskieren(int maske);  | KARA maskiert alle Codewörter mit der Maske maske ( $maske \in \{0;1;\dots;7\}$ ).          |
| formatEintragen();     | Trägt die Formatinformationen ein.  |
| builderOeffnen();      | Öffnet den QR-Builder aus qrCode21.jar und stellt die aktuelle KARA-Welt als QR-Code dar.   |

Wurden diese Schritte beachtet, können die in Tabelle 13 dargestellten Methoden mit dem Präfix qrKara. verwendet werden.

Das in Abbildung 30 (vorige Seite) dargestellte Programm ermöglicht durch Eingabe von »Hallo Karawelt!« das Erzeugen der in Abbildung 31 dargestellten KARA-Welt und das Darstellen des KARA-QR-Codes im QR-Builder.

Damit die Lernenden den Prozess des Encodierens selbstständig implementieren können, sind Kenntnisse im Umgang mit JAVAKARA und Kenntnisse zu Algorithmen und Datenstrukturen Voraussetzung. Es wurde eine Aufgabensammlung entwickelt, die sich hier im »Anhang« befindet (siehe Seite 30 ff.) und die es ermöglicht, das Encodieren in JAVAKARA mit unterschiedlichen Lernvoraussetzungen zu realisieren. Die Aufgaben wurden entsprechend einem Kompetenzmodell zum Thema *Algorithmen* in drei Kompetenzstufen eingeordnet und können zur inneren Differenzierung verwendet werden (vgl. Kohl, 2009, S.93). Darüber hinaus können die entstandenen Programme zu verschiedenen Zeitpunkten im Informatikunterricht eingesetzt und weiterentwickelt werden. So ist es möglich, die KARA-Programme als Grundlage für JAVA-Programme in der Sekundarstufe II zu verwenden. Die grafische Oberfläche des QR-Builders kann auch in diesem Zusammenhang zur Ausgabe verwendet werden. Die Beschreibung der Schnittstelle liegt dem JAVA-Package qrCode21 bei.

Abbildung 31: KARA-QR-Code in der KARA-Welt (links) und dargestellt im QR-Builder (rechts).



## Fazit

Kompetenzorientierter Informatikunterricht soll die vier Grundsätze »Altersgemäßheit beachten«, »Inhalte vernetzen«, »Mit Unterschieden klug umgehen« und »Methodenvielfalt anstreben« berücksichtigen (vgl. Fothe, 2010, S.9 ff.). Die vorliegende Unterrichtsreihe »Von der Information zum QR-Code und wieder zurück« bietet dazu zahlreiche Möglichkeiten:

- ▷ *Altersgemäßheit beachten:* Die theoretischen Vorbetrachtungen liefern eine detaillierte Beschreibung zum Aufbau der QR-Codes sowie zu den Encodierungs- und Decodierungs-Schritten. Je nach Alter können einzelne Bestandteile oder Schritte ausgelassen bzw. von der Anwendung `qrCode21.jar` übernommen und zu einem späteren Zeitpunkt vertieft besprochen werden. Die Umsetzung mit einer Programmiersprache kann in der Sekundarstufe I mit JAVA KARA begonnen und in der Sekundarstufe II in einer Programmiersprache wie JAVA fortgesetzt werden.
- ▷ *Inhalte vernetzen:* Mit dem Thema *QR-Codes* werden verschiedene informatische Grundlagen aufgegriffen. Es wird der Zusammenhang von Information und Daten, die Codierung (insbesondere die binäre Codierung) und Decodierung sowie die Digitalisierung thematisiert. Gleichzeitig kann das Thema *QR-Codes* als Ausgangspunkt für Unterrichtseinheiten zu den Themen *Algorithmen* und *Datenmodellierung* genutzt werden.
- ▷ *Mit Unterschieden klug umgehen:* Die Unterrichtsreihe berücksichtigt die individuellen Lernvoraussetzungen der Schülerinnen und Schüler auf verschiedene Weise. Die Lernenden können unterschiedliche Interessen (das Mitbringen eigener QR-Codes) in den Unterricht einbringen sowie Aufgaben unterschiedlich schnell (die Anzahl der bearbeiteten QR-Codes und Codewörter) und auf verschiedenen Niveaustufen (Arbeit in Expertengruppen, Programmierung in JAVA KARA) bearbeiten. Gleichzeitig wird durch Unterrichtsgespräche und die Arbeit in Stammgruppen ein Mindeststandard sichergestellt.
- ▷ *Methodenvielfalt anstreben:* Die Unterrichtsreihe ermöglicht ein methodisch abwechslungsreiches Unterrichtsgeschehen. Es findet ein Wechsel zwischen Unterrichtsgesprächen und Phasen der Einzel-, Partner- und Gruppenarbeit statt. Zusätzlich wird die Vielfalt durch kooperative Lernformen und das rolenspielähnliche Nachvollziehen der Algorithmen erhöht.

Die vorliegenden Ausführungen haben gezeigt, dass das Phänomen *QR-Codes* ein lohnenswertes Thema für den Informatikunterricht ist. Zum einen können viele informatische Inhalte am Beispiel eines QR-Codes vermittelt werden und zum anderen können Schülerinnen und Schüler ihre Erfahrungen aus dem realen Leben in den Informatikunterricht einbeziehen.

Entwicklung und Anwendung von QR-Codes werden stetig vorangetrieben, so werden sie beispielsweise in *Coffee Shops* beim Bezahlvorgang verwendet oder von Nahverkehrsunternehmen genutzt, um die Ankunftszeiten von Bussen an die Passagiere weiterzugeben (vgl. Uitz/Harnisch, 2012; siehe auch Abbildung 22, Seite 23). Es ist davon auszugehen, dass zukünftig noch mehr Informatiksysteme mit QR-Codes arbeiten. Damit vergrößert sich auch das Potenzial für den Informatikunterricht, sodass im Zusammenhang mit QR-Codes viele weitere informatische Inhalte (z. B. *Datenmodellierung* und *Datenschutz*) im Unterricht erarbeitet und besprochen werden können.

### Danksagung

Ich danke Fabrice Stellmacher, der im Rahmen einer von der Autorin betreuten Projektarbeit Anregungen für diese Arbeit lieferte.

### LOG-IN-Service

Im LOG-IN-Service (siehe Heft 178/179, Seite 95) stehen die in diesem Beitrag (Seite 10 und Seite 22) erwähnten Arbeitsmaterialien zum Herunterladen zur Verfügung.

# 4

## Anhang

### Tabellen und Arbeitsblätter

Tabelle 14 (rechts):  
Encodierungs-/Dekodierungstabelle  
für den Modus *Alphanumeric*.

| Char. | Value | Char. | Value | Char. | Value | Char. | Value | Char. | Value | Char. | Value | Char. | Value | Char. | Value |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 6     | 6     | C     | 12    | I     | 18    | O     | 24    | U     | 30    | SP    | 36    | .     | 42    |
| 1     | 1     | 7     | 7     | D     | 13    | J     | 19    | P     | 25    | V     | 31    | \$    | 37    | /     | 43    |
| 2     | 2     | 8     | 8     | E     | 14    | K     | 20    | Q     | 26    | W     | 32    | %     | 38    | :     | 44    |
| 3     | 3     | 9     | 9     | F     | 15    | L     | 21    | R     | 27    | X     | 33    | *     | 39    |       |       |
| 4     | 4     | A     | 10    | G     | 16    | M     | 22    | S     | 28    | Y     | 34    | +     | 40    |       |       |
| 5     | 5     | B     | 11    | H     | 17    | N     | 23    | T     | 29    | Z     | 35    | -     | 41    |       |       |

aus: ISO/IEC, 2006, S. 26

| Byte | Char. | Byte | Char. | Byte | Char. | Byte | Char. | Byte | Char. | Byte | Char. | Byte | Char. | Byte | Char. |
|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|
| 0    | NUL   | 32   | space | 64   | @     | 96   | `     | 128  |       | 160  | NBSP  | 192  | À     | 224  | à     |
| 1    | SOH   | 33   | !     | 65   | A     | 97   | a     | 129  |       | 161  | ¡     | 193  | Á     | 225  | á     |
| 2    | STX   | 34   | "     | 66   | B     | 98   | b     | 130  |       | 162  | ¢     | 194  | Â     | 226  | â     |
| 3    | ETX   | 35   | #     | 67   | C     | 99   | c     | 131  |       | 163  | £     | 195  | Ã     | 227  | ã     |
| 4    | EOT   | 36   | \$    | 68   | D     | 100  | d     | 132  |       | 164  | ¤     | 196  | Ä     | 228  | ä     |
| 5    | ENQ   | 37   | %     | 69   | E     | 101  | e     | 133  |       | 165  | ¥     | 197  | Å     | 229  | å     |
| 6    | ACK   | 38   | &     | 70   | F     | 102  | f     | 134  |       | 166  | ¦     | 198  | Æ     | 230  | æ     |
| 7    | BEL   | 39   | '     | 71   | G     | 103  | g     | 135  |       | 167  | §     | 199  | Ç     | 231  | ç     |
| 8    | BS    | 40   | (     | 72   | H     | 104  | h     | 136  |       | 168  | ¨     | 200  | È     | 232  | è     |
| 9    | HT    | 41   | )     | 73   | I     | 105  | i     | 137  |       | 169  | ©     | 201  | É     | 233  | é     |
| 10   | LF    | 42   | *     | 74   | J     | 106  | j     | 138  |       | 170  | ª     | 202  | Ê     | 234  | ê     |
| 11   | VT    | 43   | +     | 75   | K     | 107  | k     | 139  |       | 171  | «     | 203  | Ë     | 235  | ë     |
| 12   | FF    | 44   | ,     | 76   | L     | 108  | l     | 140  |       | 172  | ¬     | 204  | Ì     | 236  | ì     |
| 13   | CR    | 45   | -     | 77   | M     | 109  | m     | 141  |       | 173  | SHY   | 205  | Í     | 237  | í     |
| 14   | SO    | 46   | .     | 78   | N     | 110  | n     | 142  |       | 174  | ®     | 206  | Î     | 238  | î     |
| 15   | SI    | 47   | /     | 79   | O     | 111  | o     | 143  |       | 175  | ¯     | 207  | Ï     | 239  | ï     |
| 16   | DLE   | 48   | 0     | 80   | P     | 112  | p     | 144  |       | 176  | °     | 208  | Ð     | 240  | ð     |
| 17   | DC1   | 49   | 1     | 81   | Q     | 113  | q     | 145  |       | 177  | ±     | 209  | Ñ     | 241  | ñ     |
| 18   | DC2   | 50   | 2     | 82   | R     | 114  | r     | 146  |       | 178  | ²     | 210  | Ò     | 242  | ò     |
| 19   | DC3   | 51   | 3     | 83   | S     | 115  | s     | 147  |       | 179  | ³     | 211  | Ó     | 243  | ó     |
| 20   | DC4   | 52   | 4     | 84   | T     | 116  | t     | 148  |       | 180  | ´     | 212  | Ô     | 244  | ô     |
| 21   | NAK   | 53   | 5     | 85   | U     | 117  | u     | 149  |       | 181  | µ     | 213  | Õ     | 245  | õ     |
| 22   | SYN   | 54   | 6     | 86   | V     | 118  | v     | 150  |       | 182  | ¶     | 214  | Ö     | 246  | ö     |
| 23   | ETB   | 55   | 7     | 87   | W     | 119  | w     | 151  |       | 183  | ·     | 215  | ×     | 247  | ÷     |
| 24   | CAN   | 56   | 8     | 88   | X     | 120  | x     | 152  |       | 184  | ¸     | 216  | Ø     | 248  | ø     |
| 25   | EM    | 57   | 9     | 89   | Y     | 121  | y     | 153  |       | 185  | ¹     | 217  | Ù     | 249  | ù     |
| 26   | SUB   | 58   | :     | 90   | Z     | 122  | z     | 154  |       | 186  | º     | 218  | Ú     | 250  | ú     |
| 27   | ESC   | 59   | ;     | 91   | [     | 123  | {     | 155  |       | 187  | »     | 219  | Û     | 251  | û     |
| 28   | FS    | 60   | <     | 92   | \     | 124  |       | 156  |       | 188  | ¼     | 220  | Ü     | 252  | ü     |
| 29   | GS    | 61   | =     | 93   | ]     | 125  | }     | 157  |       | 189  | ½     | 221  | Ý     | 253  | ý     |
| 30   | RS    | 62   | >     | 94   | ^     | 126  | ~     | 158  |       | 190  | ¾     | 222  | Þ     | 254  | þ     |
| 31   | US    | 63   | ?     | 95   | _     | 127  | DEL   | 159  |       | 191  | ¿     | 223  | ß     | 255  | ÿ     |

Tabelle 15 :  
Encodierungs-/  
Dekodierungs-  
tabelle für den  
Modus *Byte*  
(Zeichensatz  
ISO/IEC  
8859-1).

aus: ISO/IEC, 2006, S. 28


| 2-D-Code-Version                             | Beispiel  | Ausgabe-Daten  |
|--|---|--|
| <p><b>Stapel-Codes</b></p> <p>PDF 417</p>    |    | <p>M1STROEDTER/CLAUDIA<br/>DHFWYK LEJSTNFR 1599<br/>342Y000 36 10C<br/>B Typ: PDF 417 Länge: 72</p>  |
| <p><b>Matrix-Codes</b></p> <p>Aztec-Code</p> |    | <p>Kontonr. des Begünstigten:<br/>01008000010-<br/>Name. des Begünstigten:<br/>ö d t w x y z<br/>Bank des Begünstigten:<br/>Ä N Ä s q z ö j ç n ä ö " ö R Ö %<br/>Name des Auftragsgebers:<br/>Verwendungszweck:<br/>Währung:<br/>Betrag:<br/>Form-Code:<br/>Spesioption:<br/>Meldecode:</p> |
| <p>Aztec-Code</p>                            |  | <p>www.EVAG-Erfurt.de</p>  |
| <p>Data-Matrix-Code</p>                      |  | <p>Nur die von der Deutschen Post<br/>offengelegten Frankierarten 2, 8,<br/>9, 18, 25, 26 und 28 werden<br/>unterstützt. Die vorliegende<br/>Frankierart kann nicht interpretiert<br/>werden: Frankierart 05</p>   |
| <p>QR-Code</p>                               |  | <p>http://www.thuringen-kiosk.de</p>   |

Tabelle 16:  
Beispielhaftes  
Arbeitsergebnis  
der ersten Stunde.  
Die Ausgabe-Daten  
wurden mit  
bc-Tester 4.9 erzeugt.

**QR-Code infiziert Android-Smartphones** 03.10.2011

Wie Kaspersky in seinem Blog beschreibt, wurde nun erstmals ein Smartphone über einen QR-Code infiziert. Dabei wird ein Trojaner installiert, der teure SMS an einen russischen Premium-Dienst versendet. Das bei Smartphones verbreitete Verfahren, Software via QR-Code zu installieren, nutzen Angreifer für die Installation eines Trojaners aus. Die Schadsoftware sendet SMS an einen teuren Premium-Dienst.

QR-Codes sind quadratische Matrix-Barcodes, die den eindimensionalen Barcodes von Auszeichnung auf Waren ähneln. Die QR-Codes können über die Kamera eines Smartphones eingelesen werden und dabei Links auf Webseiten oder Download-URLs übertragen. Das erspart mühsame Tipp-Arbeit. Darum werden sie im Mobile-Bereich immer häufiger genutzt. Wie Kasperksy in seinem Securelist-Blog beschreibt, hat es mit einem QR-Code den ersten Angriff auf Android-Handys gegeben. Dabei versucht der eingelesene Code, den Trojaner „Trojan-SMS.AndroidOS.Jifake.f“ auf dem Smartphone zu installieren. Mit dem sogenannten Attagging wird ein Link auf den Trojaner eingeschleust. Da man vor dem Scannen des QR-Codes den Link nicht sehen kann und die Anwender in diesem Bereich nur wenig Sicherheitsbewusstsein besitzen, ist ein Angriff dabei problemlos möglich.

Laut dem Antivirenhersteller lädt der QR-Code normalerweise das Paket jimm.apk herunter. Allerdings ist das eigentliche Ziel eine andere Website. Die auf den Code verweisende URL lädt für das Smartphone die Anroid-Software ICQ-Clients Jimm herunter. Diese Version von Jimm, dem freien ICQ-Client für Java-Handys, ist mit dem Trojaner Trojan-SMS.AndroidOS.Jifake.f infiziert. Er versendet mehrere jeweils 6 US-Dollar teure SMS an einen russischen Premium-Dienst. Nach Berichten von Kaspersky bieten inzwischen auch andere Websites Trojaner an, die in Java geschrieben wurden und über QR-Code verbreitet werden.

**Abbildung 32:**

Artikel zu Gefahren im Umgang mit QR-Codes von Thorsten Eggeling (03.10.2011).

<http://www.com-magazin.de/news/sicherheit/qr-code-infiziert-android-smartphones-5818.html>



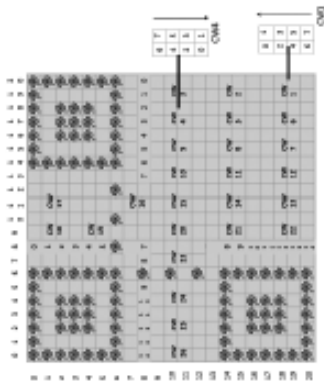
**Kompetenzstufe I:**

**Kara erstellt QR-Codes**

Ein QR-Code der Version 1 besteht aus 21x21 Modulen. Jedes der 441 Module kann dunkel (repräsentiert die binäre 1) oder hell (repräsentiert die binäre 0) gefärbt sein.

Um in Karas Welt einen QR-Code zu erzeugen, benötigt man eine Karawelt mit 21x21 Feldern, die die Module eines QR-Codes repräsentieren.

Ein Feld mit einem Kleeblatt entspricht einem dunklen Modul. Ein leeres Feld entspricht einem hellen Modul.



**Aufgabe 1: Grundgerüst erzeugen**

- a) Öffne in JavaKara das Programmieren-Fenster. Benenne die Klasse mit QRCode und ergänze den angegebenen Quelltext. Speichere dein Kara-Programm als QRCode.java.

```

import javax.swing.JOptionPane;
import org.qrcode21.*;

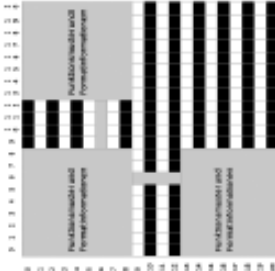
public class QRCode extends JavaKaraProgram {
    public void myProgram() {
        QRKara qrKara = new QRKara(this);
    }
}
    
```

- b) Erzeuge eine Karawelt mit 21x21 Feldern. Entwirf und implementiere in der Klasse QRCode ein KaraProgramm, das das Finder Pattern in der oberen linken Ecke einträgt.
- c) Erweitere dein Programm aus Aufgabe b) so, dass alle drei Finder Pattern eingetragen werden.
- d) Erweitere dein Programm aus Aufgabe c) so, dass zusätzlich die beiden Timing Pattern eingetragen werden.

Abbildung 33: JAVAKARA-Material Kompetenzstufe I.

**Aufgabe 2: Maske erzeugen**

Die Maske 1 hat das folgende Erscheinungsbild:



- a) Öffne eine neue Karawelt mit 21x21 Feldern und trage die Maske 1 in deine Karawelt ein. Beachte, dass die Bereiche mit Finder Pattern, Timing Pattern und Formatinformationen nicht maskiert werden. Speichere deine Karawelt als maske1.world.
- b) Führe anschließend dein Programm aus Aufgabe 1 auf der Karawelt maske1.world aus.

**Aufgabe 3: Codewörter eintragen**

Die Methode qrKara.cwEintragen(String eingabe) berechnet die Daten und Fehlerkorrektur-Codewörter zu der übergebenen Zeichenkette und trägt sie in die Karawelt ein. Wurde bereits eine Maske erzeugt, erfolgt gleichzeitig das Maskieren.

- a) Erweitere dein Kara-Programm aus Aufgabe 1 um den Methodenaufruf qrKara.cwEintragen(„Deine Eingabe“); Führe anschließend dein Programm auf der Karawelt maske1.world aus.

**Aufgabe 4: QR-Code erzeugen**

Die Methode qrKara.formatEintragen(String eingabe, int maske) berechnet zu der übergebenen Zeichenkette und Maske die Formatinformationen und trägt sie in die Karawelt ein. Die Methode qrKara.builderOffnen() stellt dir deine aktuelle Karawelt im QR-Builder dar.

- a) Erweitere dein Kara-Programm aus Aufgabe 3 um die Methodenaufrufe qrKara.formatEintragen(„Deine Eingabe“, 1); qrKara.builderOffnen(); Führe anschließend dein Programm auf der Karawelt maske1.world aus.
- b) Scanne deinen QR-Code ein und prüfe dein Ergebnis.
- c) Teste dein Kara-Programm mit den Eingaben „Hallo!“, „01234“ und „Test“. Überprüfe deine Ergebnisse, indem du die QR-Codes einscannst.

Abbildung 34:  
JAVAKARA-  
Material Kompe-  
tenzstufe II.

**Kompetenzstufe II:**

**Kara erstellt QR-Codes**

Ein QR-Code der Version 1 besteht aus 21x21 Modulen. Jedes der 441 Module kann dunkel (repräsentiert die binäre 1) oder hell (repräsentiert die binäre 0) gefärbt sein.

Um in Karas Welt einen QR-Code zu erzeugen, benötigt man eine Karawelt mit 21x21 Feldern, die die Module eines QR-Codes repräsentieren.

Ein Feld mit einem Klebblatt entspricht einem dunklen Modul. Ein leeres Feld entspricht einem hellen Modul.

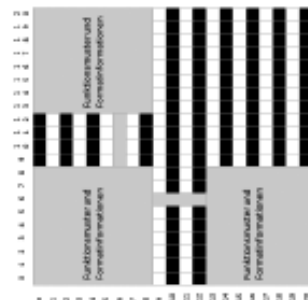
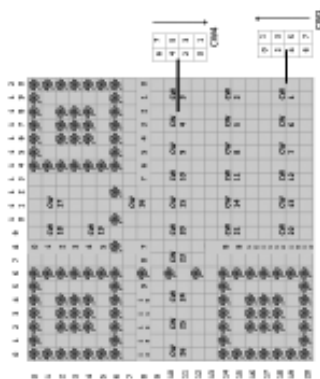
**Aufgabe 1: Grundgerüst erzeugen**

- a) Erzeuge eine Karawelt mit 21x21 Feldern. Entwirf und implementiere ein Kara-Programm Grundgeruest.java, das das Finder Pattern in der oberen linken Ecke einträgt.
- b) Erweitere dein Kara-Programm aus Aufgabe a) so, dass alle drei Finder Pattern eingetragen werden.
- c) Erweitere dein Programm aus Aufgabe b) so, dass zusätzlich die beiden Timing Pattern eingetragen werden.

**Aufgabe 2: Maske erzeugen**

Die Maske 1 hat das folgende Erscheinungsbild:

- a) Erzeuge eine Karawelt mit 21x21 Feldern. Entwirf und implementiere ein Kara-Programm Maske.java, das die Maske 1 einträgt. Beachte, dass die Bereiche mit Finder Pattern, Timing Pattern und Formatinformationen nicht maskiert werden.
- b) Führe anschließend dein Kara-Programm aus Aufgabe 1 aus. Speichere die Karawelt als maske1.world.



**Aufgabe 3: Codewörter eintragen**

Die Methode qrKara.cwEintragen(String eingabe) berechnet die Daten und Fehlerkorrektur-Codewörter zu der übergebenen Zeichenkette und trägt sie in die Karawelt ein. Wurde bereits eine Maske erzeugt, erfolgt gleichzeitig das Maskieren.

```
import java.util.Scanner;
import qrCode31.*;

public class qrCode extends JavaKaraSystem {
    public void qrEintragen() {
        qrKara qrKara = new qrKara(this);
    }
}
```

- a) Öffne in JavaKara das Programmieren-Fenster. Benenne die Klasse mit QRCode und ergänze den angegebenen Quelltext. Speichere dein Kara-Programm als QRCode.java.

**b) Ergänze in diesem Kara-Programm die Methode:**

qrKara.cwEintragen(„Deine Eingabe“);

Führe anschließend dein Programm auf der Karawelt maske.world aus und speichere die neue Welt als maskiert.world.

**Aufgabe 4: Formatinformationen eintragen**

Die Methode qrKara.formatErzeugen(String Eingabe, int maske) berechnet zu der übergebenen Zeichenkette und Maske die Formatinformationen und gibt sie als Zeichenkette (String) zurück. Die Methode qrKara.builderOffnen() stellt dir deine aktuelle Karawelt im QR-Buildler dar.

- a) Erweitere dein Kara-Programm aus Aufgabe 3 so, dass die Formatinformationen in die Karawelt maskiert.world eintragen wird.
- b) Erweitere dein Kara-Programm aus Aufgabe 4a) um den Methodenaufruf qrKara.builderOffnen(); Führe anschließend dein Programm auf der Karawelt maskiert.world aus.
- c) Scanne deinen QR-Code ein und prüfe dein Ergebnis.
- d) Teste deine Kara-Programme Grundgeruest.java, Maske.java und QRCode.java mit den Eingaben „Hallo!“, „01234“ und „Test“. Überprüfe deine Ergebnisse, indem du die QR-Codes einscannst.

Kompetenzstufe III:

Kara erstellt QR-Codes

Ein QR-Code der Version 1 besteht aus 21x21 Modulen. Jedes der 441 Module kann dunkel (repräsentiert die binäre 1) oder hell (repräsentiert die binäre 0) gefärbt sein.

Um in Kara Welt einen QR-Code zu erzeugen, benötigt man eine Karawelt mit 21x21 Feldern, die die Module eines QR-Codes repräsentieren.

Ein Feld mit einem Klebblatt entspricht einem dunklen Modul. Ein leeres Feld entspricht einem hellen Modul.

Aufgabe 1: Grundgerüst erzeugen

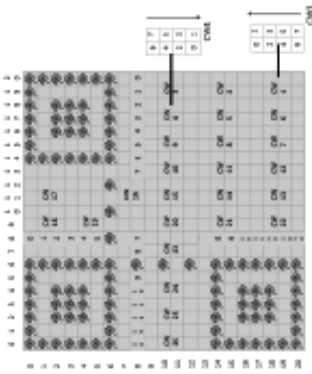
- a) Öffne in JavaKara das Programmierfenster. Benenne die Klasse mit QRCode und ergänze den angegebenen Quelltext.
- Speichere dein Kara-Programm als QRCode.java.

```

import java.awt.event.ActionEvent;
import javax.swing.*;

public class QRCode extends JavaKaraProgram {
    public void myProgram() {
        QRCode qrKara = new QRCode(12345);
    }
}
    
```

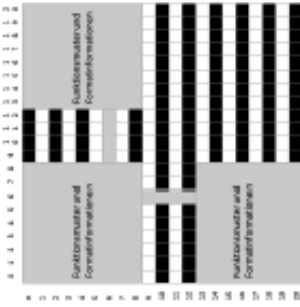
- b) Entwirf und implementiere für die Klasse QRCode eine Methode grundgeruestErzeugen(), die eine Karawelt mit 21x21 Feldern erzeugt und die das Finder Pattern in der oberen linken Ecke einträgt.
- c) Erweitere deine Methode aus Aufgabe b) so, dass alle drei Finder Pattern eingetragen werden.
- d) Erweitere deine Methode aus Aufgabe c) so, dass zusätzlich die beiden Timing Pattern eingetragen werden.



Aufgabe 2: Maske erzeugen

Die Maske 1 hat das folgende Erscheinungsbild:

- a) Entwirf und implementiere eine Methode maskeEintragen(), die die Maske 1 einträgt. Beachte, dass die Bereiche mit Finder Pattern, Timing Pattern und Formatinformationen nicht maskiert werden.



Zusatzaufgabe:

- b) Entwirf und implementiere eine Methode maskeEintragen(int maske), die die Maske maske (0, 1, ..., 7) erzeugt.

Aufgabe 3: Codewörter eintragen

Die Methode qrKara.cwErzeugen(String eingabe) berechnet die Daten- und Fehlerkorrektur-Codewörter zu der übergebenen Zeichenkette und gibt sie als eine Zeichenkette von Nullen und Einsen (String) zurück.

- a) Entwirf und implementiere eine Methode cwEintragen(String eingabe), die die Codewörter in die Karawelt einträgt.

Aufgabe 4: Formatinformationen eintragen

Die Methode String qrKara.formatErzeugen(String Eingabe, int maske) berechnet zu der übergebenen Zeichenkette und Maske die Formatinformationen und gibt sie als Zeichenkette (String) zurück.

- a) Entwirf und implementiere eine Methode formatEintragen(String eingabe, int maske), die die Formatinformationen in die Karawelt einträgt.

Die Methode qrKara.builderOeffnen() stellt dir deine aktuelle Karawelt im QR-Builder dar.

- b) Erweitere dein Hauptprogramm um den Methodenaufruf qrKara.builderOeffnen(); und führe anschließend alle Methoden nacheinander aus.
- c) Scanne deinen QR-Code ein und prüfe dein Ergebnis.
- d) Teste dein Kara-Programm mit verschiedenen Eingaben. Überprüfe deine Ergebnisse, indem du die QR-Codes einscannst.

Abbildung 35: JAVAKARA-Material Kompetenzstufe III.



## Nützliche Internetquellen

Zum Decodieren von QR-Codes – bcTester 4.9:  
<http://www.bctester.de/>

Zum Erstellen von QR-Codes – Portabler QR-Code Generator v1.9.0:  
<http://www.heise.de/download/qr-code-generator-1185046.html>

Zum Generieren der Fehlerkorrektur-Codewörter:  
<http://www.thonky.com/qr-code-tutorial/show-division-steps/>  
oder  
<http://www.pclviewer.com/rs2/calculator.html>

Weitere nützliche Links:  
<http://barcode.tec-it.com/barcode-generator.aspx?LANG=en>  
und  
<http://goqr.me/de/>

## Literatur und Internetquellen

AKBSI – Arbeitskreis „Bildungsstandards“ der Gesellschaft für Informatik (Hrsg.): Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I. Empfehlungen der Gesellschaft für Informatik e.V. vom 24. Januar 2008. In: LOG IN, 28. Jg. (2008), Nr. 150/151, Beilage.

Eggeling, Th.: QR-Code infiziert Android-Smartphones. 03.10.2011.  
<http://www.com-magazin.de/news/sicherheit/qr-code-infiziert-android-smartphones-5818.html>

Fothe, M.: Kunterbunte Schulinformatik – Ideen für einen kompetenzorientierten Unterricht in den Sekundarstufen I und II. Berlin: LOG IN Verlag, 2010.

Fothe, M.: Lasst uns kleine Welten schaffen! KARA, PUCK und die optische Telegrafie. In LOG IN, 31. Jg. (2011/2012), Heft 172/173, S.40–44.

Hartzt, W.: Basiswissen QR-Code. 03.03.2008.  
<http://qrcode.wilkohartz.de/>

ISO/IEC 18004:2000(E): Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code. First Edition: 2000-06-15.  
<http://www.codeplex.com/Download?ProjectName=qrcodenet&DownloadId=284291>

ISO/IEC 18004:2006(E): Information technology — Automatic identification and data capture techniques — QR Code 2005 bar code symbology specification. Second Edition: 2006-09-01.  
<http://www.codeplex.com/Download?ProjectName=qrcodenet&DownloadId=321409>

Kohl, L.: Kompetenzorientierter Informatikunterricht in der Sekundarstufe I unter Verwendung der visuellen Programmiersprache Puck. Jena: Fakultät für Mathematik und Informatik der Friedrich-Schiller-Universität Jena, 2009 (Dissertation).  
<http://www.db-thueringen.de/servlets/DerivateServlet/Derivate-17565/Dissertation.pdf>

Law, C.; So, S.: QR Codes in Education. In: JETDE – Journal of Educational Technology Development and Exchange, 3. Jg. (2010), Heft 1, S. 85–100.  
<http://www.sicet.org/journals/jetde/jetde10/7-So.pdf>

QR Code Tutorial. 2012.  
<http://www.thonky.com/qr-code-tutorial/>

Reichert, R.; Nievergelt, J.; Hartmann, W.: Programmieren mit Kara – Ein spielerischer Zugang zur Informatik. Reihe »eXamen.press«. Berlin u. a.: Springer, 2005.

Reichert, R. u. a.: Programmieren lernen mit Kara – Kara: Lernumgebungen rund ums Programmieren. 2007-08-03.  
<http://www.swisseduc.ch/informatik/karatojava/>

Uitz, I.; Harnisch, M.: Der QR-Code – aktuelle Entwicklungen und Anwendungsbereiche. In: Informatik Spektrum, 35. Jg. (2012), Heft 5, S.339–347.

Wie ein QR-Code codiert wird Turtorial / QR-Code encoding tutorial! 2012.  
<http://blauerbildschirm.wordpress.com/tag/qr-code-tutorial/>

Alle Internetquellen wurden zuletzt am 15. Juli 2014 geprüft und können aus dem Service-Bereich des LOG IN Verlags (<http://www.log-in-verlag.de>) heruntergeladen werden.



Beilage zu LOG IN, 34 Jg. (2014), Heft Nr. 178/179

Claudia Strödter  
Friedrich-Schiller-Universität Jena  
Fakultät für Mathematik und Informatik  
Ernst-Abbe-Platz 2  
07743 Jena  
E-Mail: [claudia.stroedter@uni-jena.de](mailto:claudia.stroedter@uni-jena.de)

LOG IN Verlag GmbH  
Redaktion LOG IN  
Friedrichshaller Straße 41  
14199 Berlin  
E-Mail: [redaktionspost@log-in-verlag.de](mailto:redaktionspost@log-in-verlag.de)  
URL: <http://www.log-in-verlag.de/>

---

**L O G I N**  
V E R L A G

---