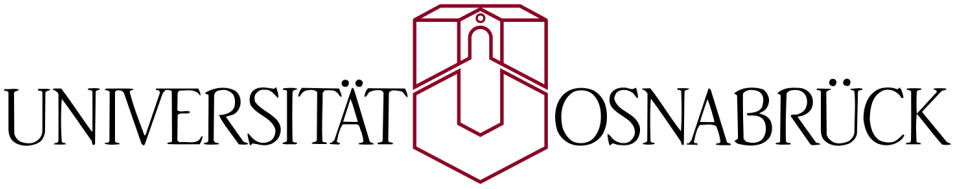


Ein Classroom-Quiz

mit Hilfe von VBA-Skripten in Powerpoint und Java-MIDlets in Bluetooth-fähigen Mobiltelefonen

Bachelorarbeit
von
Sergiy Krutykov

Gutachter:
Prof. Dr. Oliver Vornberger
Jun.Prof. Dr.-Ing Elke Pulvermüller

Fachbereich Mathematik/Informatik
Universität Osnabrück

29.09.2008

Danksagung

Ich möchte mich bei allen bedanken, die mich beim Erstellen dieser Arbeit unterstützt haben, besonders bei:

- Herrn Prof. Dr. Oliver Vornberger für die Betreuung der Arbeit und dafür, dass er mir immer wieder neue Hinweise und Anregungen gegeben hat.
- Patrick Fox für das Korrekturlesen dieser Arbeit und dafür, dass er immer zu einer fachlichen Diskussion bereit war und mich immer softwaretechnisch unterstützt hat.
- Friedhelm Hofmeyer für die technische Unterstützung.
- Dorothee Langfeld für die Mitwirkung an besonderes kritischen Stellen.

Darüber hinaus danke ich auch allen, die voller Geduld an den zahlreichen Testen teilgenommen haben.

Warenzeichen

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

1 Einleitung

Während einer Vorlesung oder einer Unterrichtsstunde hat die Dozentin oder der Dozent bzw. eine Lehrerin oder ein Lehrer (im Weiteren der Vortagende) häufig den Wunsch, ein bisschen Feedback von den zuhörenden Studenten bzw. Schülern (im weiteren die Zuhörer) zu bekommen, ob diese wirklich zuhören, und ob sie das Thema verstehen. Deshalb stellen die Vortragenden öfters eine didaktisch konzipierte Frage ins Publikum, wonach manchmal sehr lange Pause eintritt, bis sich jemand traut, sich zu melden. Dabei gibt es unter den Zuhörern nur eine kleine „aktive“ Untergruppe, die sich meldet und der Rest bleibt unbeteiligt, so dass der Vortragende ein Gefühl bekommt, dass er ins Leere spricht.
Um diese Situation zu verbessern, werden zum Beispiel an einigen Universitäten Amerikas alle Studenten dazu gezwungen, auf solche Fragen zu antworten. Das ganze sieht dann so aus, wie der Publikum-Joker in der Show „Wer wird Millionäre“: Der Dozent stellt eine Frage in seiner Präsentation mit vier alternativen Antworten 'A', 'B', 'C' und 'D' und die Studenten müssen mittels speziellen Fernbedienungen darauf antworten.
Diesen Vorgang bezeichnet man als *Classroom-Quiz*, welcher in dieser Arbeit implementiert werden musste.

1.1 Aufgabenstellung

Das Ziel dieser Arbeit ist, die oben genannten Fernbedienungen durch Bluetooth-fähige Mobiltelefone zu ersetzen, die die Antworten 'A'-'D' an den Rechner des Vortragenden senden können, wobei die Häufigkeiten der einzelnen Antworten prozentual in einer Powerpoint-Präsentation visualisiert werden sollen. Damit ist die Aufgabe klar eingegrenzt: Es muss ein Programm für Mobiltelefone geschrieben, (möglichst) automatische Manipulation von Powerpoint-Folien ermöglicht und eine Schnittstelle zwischen Powerpoint und Mobiltelefonen erschaffen werden.

1.2 Aufbau der Arbeit

Diese Arbeit ist im Wesentlichen in drei Teile gegliedert.
Der erste Teil wird im Abschnitt 2 behandelt. Es wird zuerst ein kleiner Einblick in die Microsoft Skript-Sprache *Visual Basic for Applications* gegeben. Dann wird anhand der Classroom-Quiz-Folien selbst erklärt, wie die Folien automatisch erzeugt, die sogenannten *Add-Ins* benutzt und die selbstauslösenden Makros geschrieben werden.

Im zweiten Teil, der im Abschnitt 3 behandelt wird, werden zuerst die allgemeinen Konzepte der J2ME (Java 2 Micro Edition) und die Arbeitsweise der MIDlets selbst beschrieben. Danach folgt die Beschreibung, wie ein MIDlet für ein Classroom-Quiz aussehen kann.
Das dritte Teil (Abschnitt 4) liefert einige rein technische Details für die Bluetooth-Technologie sowie eine kleine Übersicht über das allgemeine Netzwerk-Konzept, nach welchen die konkrete Implementation der Kommunikation zwischen Mobiltelefonen und Powerpoint folgt.

2 Classroom-Quiz in Powerpoint

Die bei Powerpoint ankommenden Häufigkeiten der einzelnen Antworten auf die Quiz-Frage müssen visualisiert werden und der Empfang von diesen Häufigkeiten muss auch gesteuert werden, wobei beides möglichst automatisch geschehen sollte.
Im Wesentlichen gibt es nur zwei Möglichkeiten, eine Powerpoint-Präsentation zu automatisieren: mit einer der Sprachen von Visual Studio, oder mit *VBA* (Visual Basic for Applications) [1], das in MS Office standardmäßig dabei ist. In dieser Arbeit wurde VBA verwendet.

Hinweis: Wenn von einer *Präsentation* die Rede ist, weiß man nicht genau, ob eine Präsentation im Vollbildmodus, oder ein ppt-Dokument gemeint ist. Um die Verwirrungen zu vermeiden, wird im Weiteren das erste einfach als eine Präsentation und das zweite als ein Präsentationsdokument bezeichnet.

2.1 Makros in Powerpoint

Visual Basic for Applications ist eine Programmierumgebung und gleichzeitig eine Sprache für Erstellung von Makros [1][2] unter anderem für MS Office Anwendungen, d.h., von kleinen Programmen, die die Fähigkeiten dieser Anwendungen erweitern.

2.1.1 Makros in VBA

In VBA gibt es Funktionen und Prozeduren, die unterschiedliche Parameterlisten erhalten können, wobei Funktionen zusätzlich einen Rückgabewert definieren. Die Modifikatoren Private oder Public können ebenfalls benutzt werden. Als ein Makro kann nur eine Prozedur ohne Modifikator und mit einer leeren Parameter-Liste, wie zum Beispiel die folgende Prozedur, gelten:

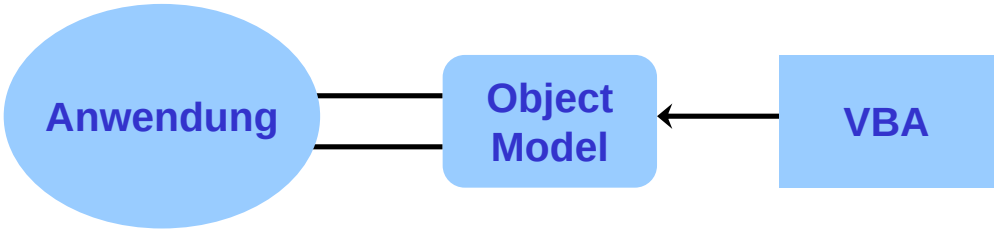
```
Sub test()
    MsgBox "Hello, Wolrd!"
End Sub
```

Die Prozedur MsgBox öffnet ein Mitteilungsfenster mit dem übergebenen String:

Dieser Code kann zum Beispiel währen der Präsentation (also im Vollbildmodus) auf einer beliebigen Folie ausgeführt werden. Dazu müssen auf dieser Folie irgendeine Form (Formen oder Shapes sind unter anderem alle auf der Folie sichtbare Objekte, solche wie Textboxes, Kreise, Linien, ..., die von dem Benutzer auf eine Folie platziert werden können) mit Makros versehen werden [3]. Wenn man jetzt die Präsentation startet und dann auf dieser Form klickt, muss das Mitteilungsfenster von oben (in der laufenden Präsentation) erscheinen.

2.1.2 Manipulation von MS Powerpoint Präsentationen: Object Model

Da in dieser Arbeit die Absicht bestand, die Ergebnisse der Classroom-Quiz-Abstimmung auf einer Folie automatisch zu visualisieren, musste es ermöglicht werden, dass die Folien durch VBA manipuliert werden.
VBA selbst kann die Objekte auf der Folie eigentlich nicht manipulieren, denn es hat keinen speziellen Zugang zu dem Inneren von Applikationen. Dafür stellt die Applikation selbst ihre Fähigkeiten im Rahmen eines sogenannten *Object Models* [1][2] zur Verfügung, während VBA zu dem Object Model „sprechen“ kann:



Im Object Model wird jeder Teil der Applikation zu einem Objekt, das eigene Liste von Funktionen besitzt. Jede Applikation hat ein eigenes spezifisches Object Model und VBA manipuliert lediglich diese Objekte, soweit die (Host-)Applikation es erlaubt. Die Manipulation sieht meistens so aus, dass einem Objekt eine Eigenschaft zugewiesen wird. VBA gilt dadurch als Objekt-orientiert.
In der Abbildung 1 ist ein kleiner Ausschnitt aus der Objekte-Hierarchie des Object Models von MS Powerpoint schematisch dargestellt. Das oberste Objekt in dieser Hierarchie ist die **Application**, also die Anwendung selbst (z.B. Powerpoint) in welcher VBA ausgeführt wird. Dieses Object hat unter anderem die Eigenschaft **Name** - einen gegen Veränderungen geschützten String, der den offiziellen Namen der Anwendung enthält (bei Powerpoint ist natürlicherweise „MS Powerpoint“ in ihm gespeichert). MS

Powerpoint kann mehrere laufende Präsentationen (**SlideShowWindows**) sowie mehrere geöffnete Präsentationsdokumente (**Presentations**) haben, wobei jedes geöffnete Präsentationsdokument mehrere Folien (**Slides**) haben kann. In jeder Folie ist es unter anderem möglich mit Hilfe von VBA Kopf- und Fußzeilen (**HeadersFooters**) zu manipulieren, sowie auf einzelne Formen zuzugreifen. Viele, aber nicht alle Formen haben einen Text-Rahmen (**TextFrame**), der zum Beispiel die Orientierung des Textes manipulieren kann, sowie das Objekt **TextRange** enthält, das unter anderem den Text der dazugehörigen Form enthält. In der Abbildung ist der Weg, welchen man in dieser Hierarchie gehen muss, damit man einer Form einen Text zuweisen kann, mit den Pfeilen markiert. Wenn man also in dem zuerst geöffneten Präsentationsdokument auf der fünften Folie der ersten Form den Text „Hallo“ zuweisen will, muss man folgenden VBA-Code ausführen:

```
Application.Presentations(1).Slides(5).Shapes(1).TextFrame.TextRange.Text _
    = "Hallo"
```

Der Unterstrich ist nötig in VBA, wenn man einen Zeilenumbruch macht.
An dieser Stelle ist zu bemerken, dass dies nur ein Beispiel ist, und man normalerweise keine konstanten Nummern benutzt, um ein Objekt zu identifizieren, sondern „bequemere“ Unterobjekte von dem Objekt Application, solche wie ActivePresentation, das als Platzhalter für das aktuelle Präsentationsdokument (anstatt von Presentations(1)) steht.

2.2 Classroom-Quiz-Folien

Bei einem Classroom-Quiz muss auf der aktuellen Folie sichtbar sein, wann dieses Quiz startet, welche Frage zu beantworten ist und welche Tipps gegeben werden. Zu besseren Erkennung wird für jedes Quiz in einem Präsentationsdokument eine extra Folie, eine standardisierte *Quiz-Folie* vorgesehen, die durch ein Makro automatisch erzeugt werden soll.
Die Abbildung 2 zeigt, wie diese aussehen soll. Sie besteht aus dem Titel, einem Textfeld für die Quiz-Frage, vier Bezeichnungen ’A’, ’B’, ’C’, ’D’, vier Textfeldern für die Tipps, einem Diagramm, bestehend aus vier (blauen) Balken und vier Textfeldern für Prozentzahlen und einem (grünen) Kreis, der die richtige Antwort markiert. Dieser Kreis kann auf die richtige Position (zu der richtigen Antwort) verschoben werden.
Im Folgenden werden die einzelnen Schritte in dem Makro für die Erzeugung von Folien erläutert, wobei jedes Mal am Anfang das visuelle Ergebnis eines solchen Schrittes, gefolgt von dem entsprechendem Code und von den Erläuterungen, geschildert wird.

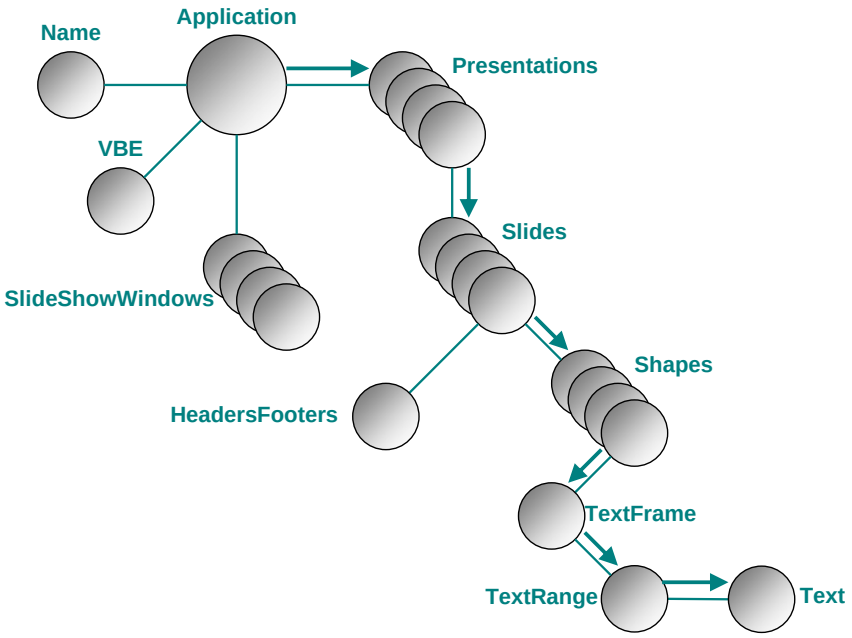


Abbildung 1: Einige Objekte aus dem Object Model von Powerpoint.

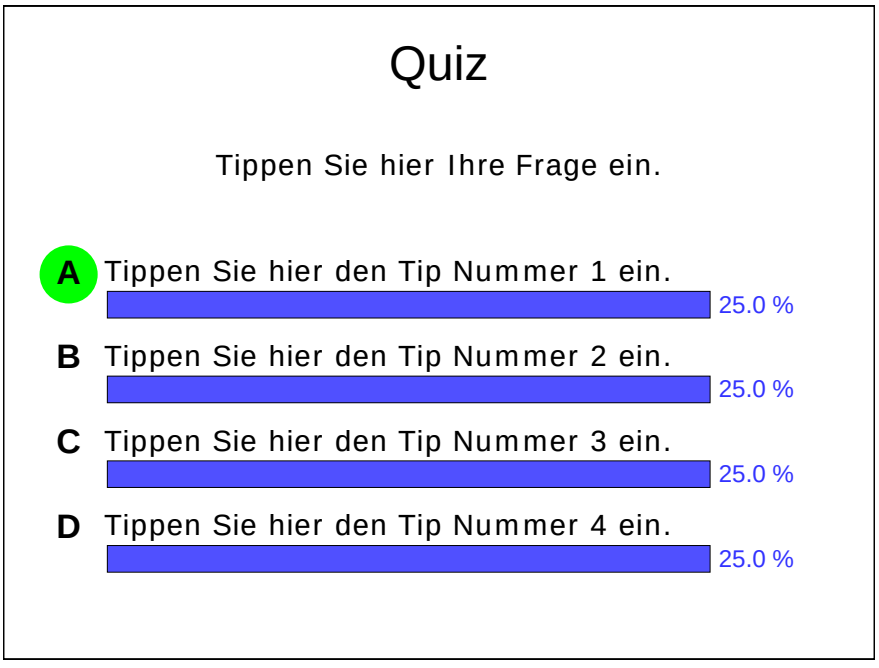
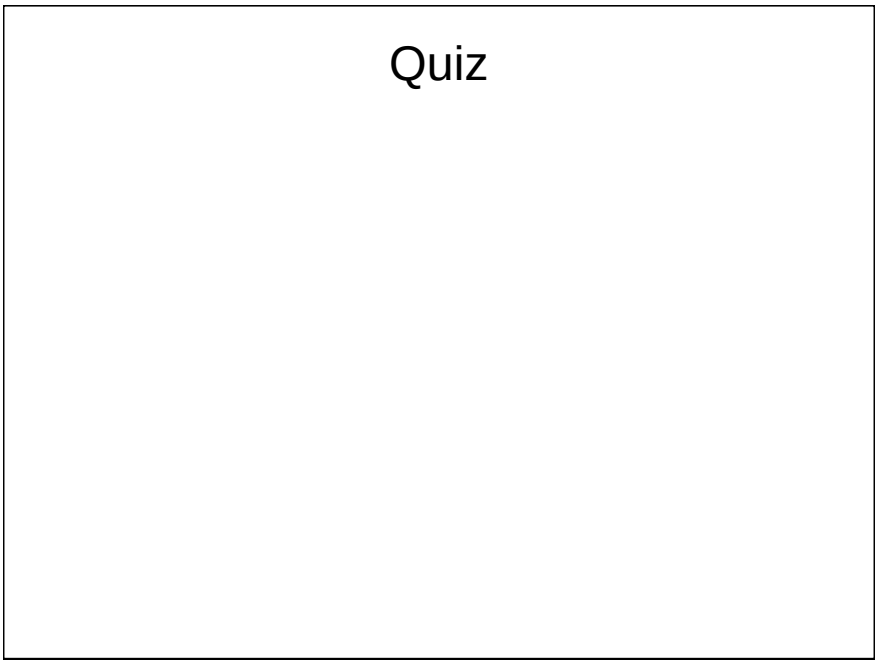


Abbildung 2: Automatisch erzeugte Standard-Classroom-Quiz-Folie.

Automatisches Erzeugen einer neuen Folie hinter der aktuellen mit dem Titel „Quiz“



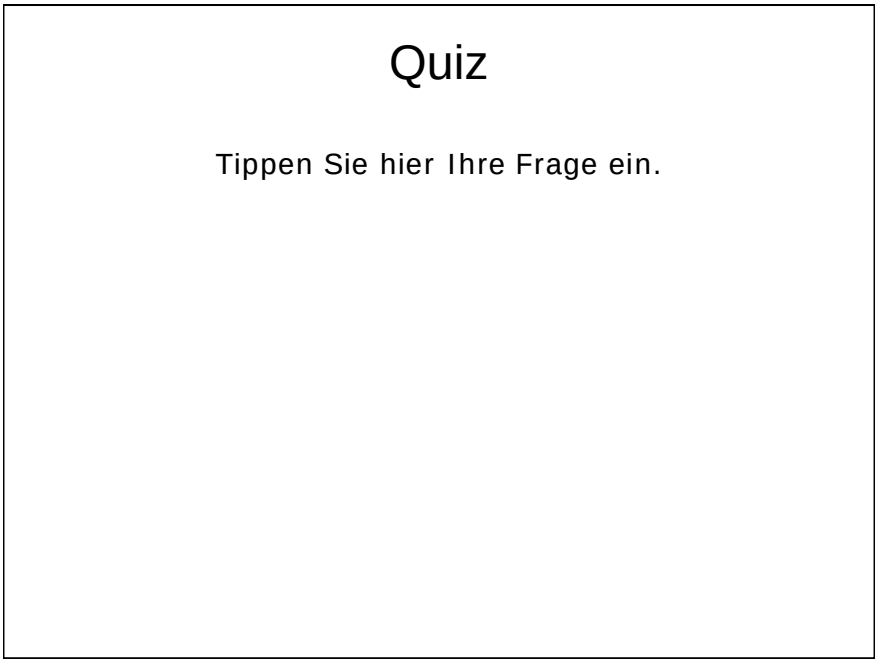

```
Dim newQuizSlide As Slide
Set newQuizSlide = ActivePresentation.Slides.Add(ActiveWindow.View.Slide. _
    SlideIndex + 1, ppLayoutTitleOnly)

newQuizSlide.Shapes(1).TextFrame.TextRange.Text = "Quiz"
newQuizSlide.Shapes(1).Name = "quizSlideTitleShape"
```

Es wird zuerst die Variable newQuizSlide vom Typ Slide deklariert. Die Set-Anweisung erzeugt diese Folie als ein Powerpoint Objekt. Da es mehrere Präsentation gleichzeitig geöffnet werden können und man nicht weiß in welcher man sich gerade befindet, hilft die Variable ActivePresentation, die aktuelle Präsentation zu identifizieren, und die Variable ActiveWindow identifiziert das zur Zeit geöffnete Bearbeitungs-Fenster (das Fenster wo die Folien entwickelt werden). Das Object ActivePresentation.Slides ist ein Array von allen Folien der aktuellen Präsentation. Mit Hilfe von ActivePresentation.Slides.Add(x, y) kann man zu dieser aktuellen Präsentation eine Folie an der Position x mit dem Layout y hinzufügen. Hier wurde für x die Position der aktuellen Folie (`ActiveWindow.View.Slide.SlideIndex`) plus eins gewählt, damit die neue Quiz-Folie direkt hinter der aktuellen auftaucht und mit dem Parameter ppLayoutTitleOnly wird eine Folie nur mit dem Titel erzeugt. Dies vereinfacht den Zugriff auf den Titel,

denn man ist sicher, dass es sich nur eine Form (Shape) auf der Folie befindet und sie ist gerade der Titel der Folie. Somit kann man über die Eigenschaft newQuizSlide.Shapes(1).TextFrame.TextRange.Text diesen Titel setzen.
Es ist immer empfehlenswert den Namen einer automatisch erzeugten Form festzulegen, sonst bleibt der „automatische“ Name wie etwa „Shape1“ erhalten und man kann diese Form nur schlecht identifizieren. In weiteren Erläuterungen werden allerdings diese „Namengebungen“ weggelassen.

Hinzufügen einer Form für die Quiz-Frage

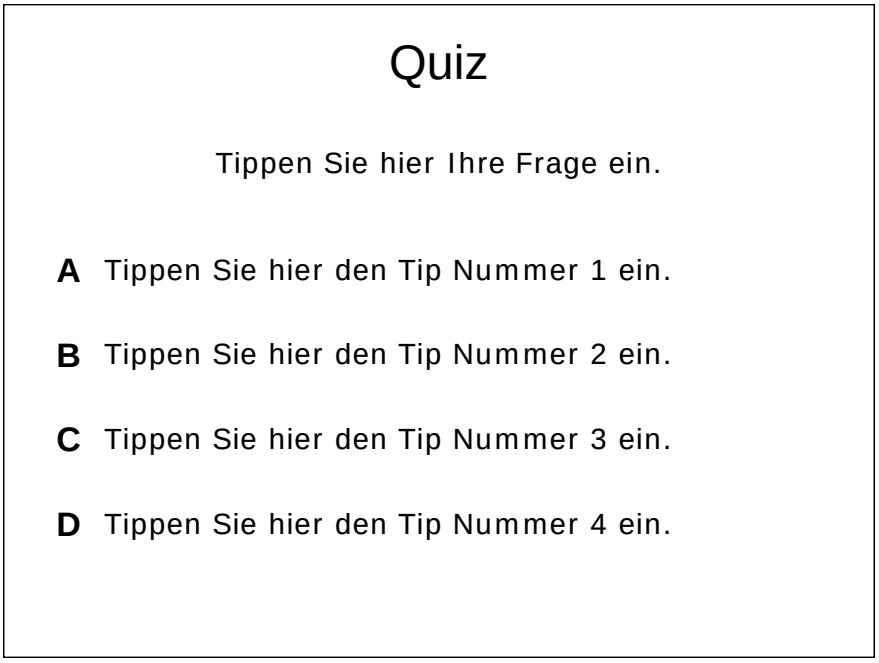



```
Dim queryTextBox As Shape
Set queryTextBox = newQuizSlide.Shapes.AddTextbox( _
    msoTextOrientationHorizontal, x0, y0, w0, h0)
    queryTextBox.TextFrame.TextRange.Text = "Tippen Sie hier Ihre Frage" _
       & "ein."
```

Für die Form für die Quiz-Frage wird ein Textfeld (TextBox) verwendet. Diese Formen sind die gleichen wie der Titel der Folie. newQuizSlide.Shapes ist ein Array für alle Formen der aktuellen Folie. Der Parameter msoTextOrientationHorizontal ist selbsterklärend, die Parameter x0, y0 sind die absoluten Koordinaten von der linken oberen Ecke der Form und w0, h0 sind ihre Breite und Höhe. Hier und im Weiteren werden keine konkreten Zahlen und Ausdrücke angegeben, sondern nur die abstrakten Variablen, wie etwa x0,

x1, ... Wie in dem Abschnitt 2.1.2 kann man mit .TextFrame.TextRange.Text den Inhalt vom TextBox ändern, was hier auch gemacht wird, wodurch auf der neuerzeugten Folie in der Frage-TextBox sofort die Aufforderung „Tippen Sie hier Ihre Frage ein.“ steht. Über .TextFrame.TextRange lässt sich nicht nur den Text sondern auch die Schrift, so wie weitere Formatierungen manipulieren.

Hinzufügen von Formen für die Bezeichnungen und Tipps

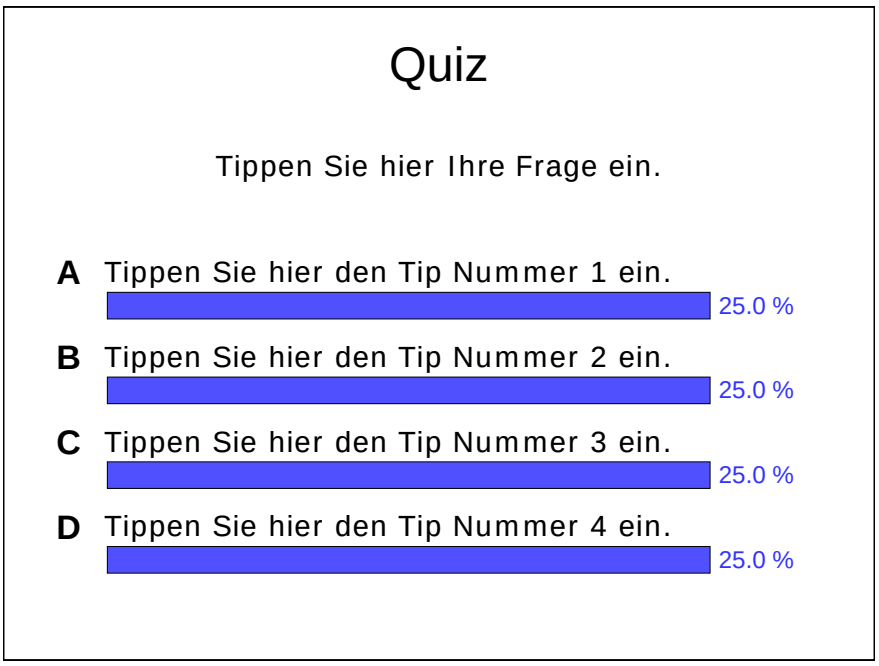



```
Dim label(4) As Shape
Dim tip(4) As Shape
For i = 0 To 3
    Set tip(i) = newQuizSlide.Shapes.AddTextbox( _
        msoTextOrientationHorizontal, x1, y1 + 70 * i, w1, h1)
    Set label(i) = newQuizSlide.Shapes.AddTextbox( _
        msoTextOrientationHorizontal, x2, y2 + 70 * i, w2, h2)
    label(i).TextFrame.TextRange.Text = Chr(65 + i)
    tip(i).TextFrame.TextRange.Text = "Tippen Sie hier den Tip Nummer " & _
        (i + 1) & " ein."
Next i
```

Bequemlichkeitshalber wurden hier für die Bezeichnungen und Tipps zwei Arrays aus jeweils 4 Elementen (label(i) und tip(i)) angelegt und in einer For-Schleife initialisiert. Analog zu dem Textfeld für die Quiz-Frage werden alle acht als TextBoxes erzeugt mit

einer entsprechenden Verschiebung nach unten. Die Text-Attribute werden analog zum vorherigen Schritt gesetzt.

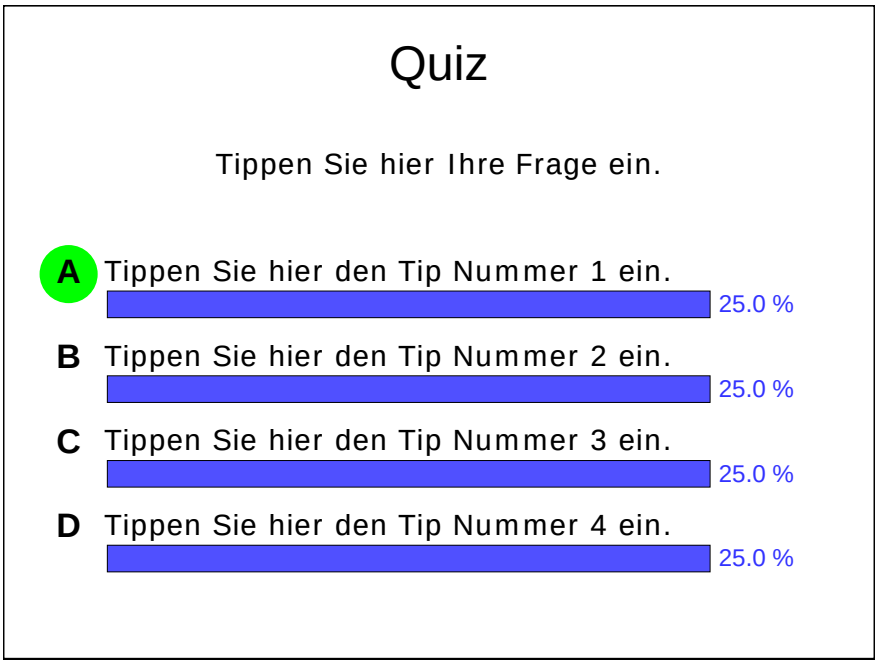
Hinzufügen von Formen für das Diagramm




```
Dim bar(4) As Shape
Dim percentage(4) As Shape
For i = 0 To 3
    Set bar(i) = newQuizSlide.Shapes.AddShape(msoShapeRectangle, x1, y1 + _
        70 * i, w1, h1)
    Set percentage(i) = newQuizSlide.Shapes.AddTextbox( _
       msoTextOrientationHorizontal, x2, y2 + 70 * i, w2, h2)
    bar(i).Fill.ForeColor.RGB = RGB(80, 80, 255)
    percentage(i).TextFrame.TextRange.Text = "25.0 %"
    ⋮
Next i
```

Analog zum vorherigen Schritt werden hier zwei Arrays für die Balken und Prozentzahlen (bar(i) und percentage(i)) angelegt und initialisiert. Die Funktion AddShape(msoShapeRectangle, x, y, w, h) unterscheidet sich von AddTextbox(msoTextOrientationHorizontal, x, y, w, h) dadurch, das hier ein Rechteck erzeugt wird, dass auch gefärbt werden kann, was mit der Anweisung (bar(i).Fill.ForeColor.RGB = RGB(80, 80, 255) geschieht.

Hinzufügen von einem Kreis




```
Dim crcl As Shape
Set crcl = newQuizSlide.Shapes.AddShape(msoShapeOval, x3, y3, w3, h3)
    crcl.Line.Visible = msoFalse
    crcl.Fill.ForeColor.RGB = RGB(0, 255, 0)
    crcl.ZOrder msoSendToBack
```

Einen Kreis kann man analog zu den Rechtecken erzeugen, indem man den Parameter msoShapeRectangle durch msoShapeOval ersetzt. Darüber hinaus kann man durch crcl.Line.Visible = msoFalse erreichen, dass keine schwarze Linie am Rande des Kreises erscheint. Die ZOrder Anweisung mit dem Parameter msoSendToBack platziert den Kreis auf die unterste Ebene, so dass er die Buchstaben 'A', 'B', 'C' oder 'D' nicht verdeckt, sondern unterhalb von ihnen liegt.

Hinzufügen von Animationsschritten

Es ist offensichtlich, dass der Kreis im Vollbildmodus nicht sofort sichtbar sein darf, denn sonst die Lösung des Quizes verraten wird. Außerdem ist es wünschenswert, bei einem Quiz, um mehr Spannung zu erzeugen, die einzelnen Formen langsam nacheinander zu zeigen. Dazu benutzt man in Powerpoint Animationsschritte, die mit Hilfe von VBA den einzelnen Formen automatisch zugeordnet werden können:

```
queryTextBox.AnimationSettings.EntryEffect = ppEffectAppear
For i = 0 To 3
label(i).AnimationSettings.EntryEffect = ppEffectWipeDown
Next i
label(0).AnimationSettings.EntryEffect = ppEffectWipeDown
For i = 1 To 3
    label(i).AnimationSettings.EntryEffect = ppEffectWipeDown
    newQuizSlide.TimeLine.MainSequence(newQuizSlide.TimeLine.MainSequence _
        .Count).Timing.TriggerType = msoAnimTriggerWithPrevious
Next i
For i = 0 To 3
    tip(i).AnimationSettings.EntryEffect = ppEffectWipeDown
Next i
crcl.AnimationSettings.EntryEffect = ppEffectAppear
```

Diese Zuordnung passiert mit dem Attribut .AnimationSettings.EntryEffect, das unter anderem auf „Erscheinen“ (ppEffectAppear) oder „Wischen“ mit der Zusatz-Option „Von oben“ (ppEffectWipeDown) gesetzt werden kann. Dadurch wird erreicht, dass im Vollbildmodus zuerst die Folie mit nur dem Titel „Quiz“ zu sehen ist, dann, nach einem Klick die Frage. Weiter tauchen die Buchstaben ’A’, ’B’, ’C’ und ’D’ nacheinander automatisch (also ohne zusätzlich Klicks) mit einer Verzögerung von einer halben Sekunde auf. Das passiert mit Hilfe von der Konstruktion .TimeLine.MainSequence(i).Timing.TriggerType = msoAnimTriggerWithPrevious, wobei die i-te Animation nach einer halber Sekunde automatisch nach der vorherigen erscheint. Man müsste also für die betroffenen Formen feststellen, welche Animationsnummer sie haben. Da sie aber gerade (eine Zeile davor) hinzugefügt werden, ist ihre Nummern immer gleich der Anzahl von Animationen in dieser Folie: .TimeLine.MainSequence.Count. Die erste Bezeichnung (label(0) = der Buchstabe ’A’) ist von dieser Prozedur ausgeschlossen, weil ’A’ gerade nach einem Klick erscheinen muss und nur ’B’, ’C’ und ’D’ danach automatisch aufgeblendet werden. Danach erscheinen nach je einem Klick die Tipps und danach durch einen weiteren Klick die Lösung in der Form eines Kreises.

2.3 Classroom-Quiz-Add-In

Damit das oben beschriebene automatisches Erzeugen von einer Folie in einem Präsentationsdokument funktioniert, muss der Code in dem Dokument vorhanden sein. D.h., jeder der diesen Code benutzen will, muss ihn in sein Präsentationsdokument integrieren. Es gibt Möglichkeit einen VBA-Modul mit relativ wenig Aufwand in den Visual Basic

Editor zu importieren, wodurch der Code in diesem Präsentationsdokument ausführbar wird. Aber der Code muss erstens in einer Textdatei unverschlüsselt geliefert werden, was für einen Programmierer nicht immer wünschenswert ist und zweitens muss der Benutzer das Importieren selbst per Hand durchführen, was das Programm nicht benutzerfreundlich macht.
Eine Lösung sind *Add-Ins* [1], die man mit Visual Studio aber auch einfach mit PowerPoint erstellen kann. In PowerPoint kann man jedes Präsentationsdokument, wenn es zumindest ein (kompilierbares) Makro enthält, als Add-In speichern. Dadurch entsteht eine Datei, die keine Folien mehr enthält, dafür aber den VBA-Code. Ein solches Add-In kann man in PowerPoint installieren, worauf alle in der Zukunft geöffneten Präsentationen es benutzen können.
Bei der Benutzung der Add-Ins tritt allerdings bei PowerPoint (im Gegensatz zu etwa MS Excel) das Problem auf, dass man auf die sich in einem installierten Add-In befindenden Makros von anderen VBA-Modulen, oder einfach von PowerPoint nicht zugreifen kann. Bei diesem Problem schaffen die Symbolleisten und Menüs Abhilfe, die auch mit Hilfe von VBA erzeugt werden können, denn sie sind einerseits für jeden zugänglich und andererseits, wenn sie zu einem Add-In gehören, haben sie Zugriff auf seinen Code: Die Symbolleisten und Menüs dienen praktisch als eine Schnittstelle zwischen dem Benutzer von einem Add-In und dem Add-In selbst, dessen Makros auf andere Weise sonst für den Benutzer nicht zugänglich wären. Darüber hinaus verleihen die Symbolleisten und Menüs einem Add-In die gewünschte Benutzerfreundlichkeit.
Die Symbolleisten und Menüs müssen also in einem Makro erzeugt werden, das sich in dem Add-In selbst befindet. Dadurch entsteht die Frage, wie dieses Makro selbst aufzurufen ist. Dafür kann man die entsprechende Prozedur mit einem in VBA vordefinierten Prozedur-Namen benennen, wodurch diese bei einem bestimmten PowerPoint-Ereignis (Event) automatisch aufgerufen wird. Einer dieser Namen ist Auto_Open(): Eine Prozedur mit diesem Namen wird bei jedem Start von PowerPoint automatisch aufgerufen. D.h., wenn ein Add-In mit der implementierten Auto_Open()-Prozedur installiert ist, wird diese Prozedur bei jedem starten von PowerPoint ausgeführt. Man muss also die Symbolleiste und Menü erzeugende Prozedur einfach Auto_Open() nennen.
Jetzt kann man den Grund verstehen, warum die Symbolleiste und das Menü temporär sein müssen, denn ansonsten würde bei jedem Starten von PowerPoint ein neues Menü und eine neue Symbolleiste dazukommen. Außerdem besteht dabei der Vorteil, dass nach der Deinstallation vom Add-In nach dem Neustart von PowerPoint die Symbolleiste und das Menü verschwinden.

Die Abbildung 3 zeigt, wie die Symbollesite und das Menü des Classroom-Quiz-Add-Ins aussehen.

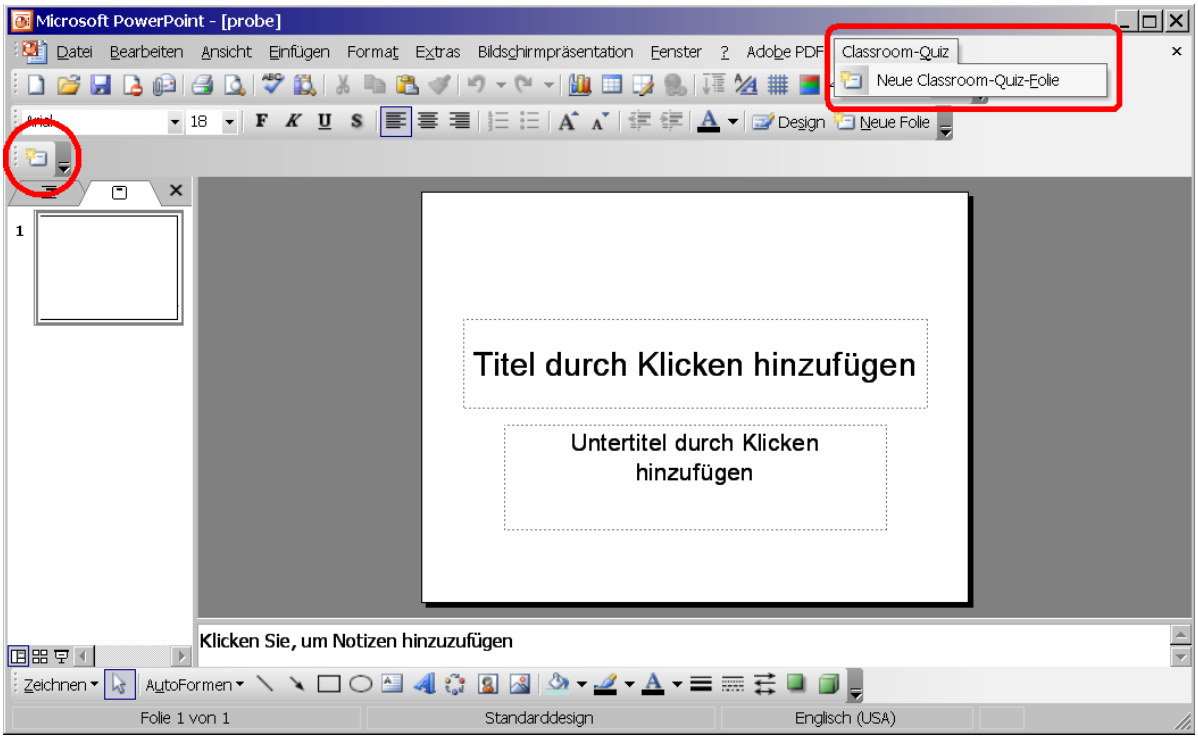


Abbildung 3: Symbol- und Menuleiste des Classromm-Quiz-Add-Ins.

Das Klicken auf der Schaltfläche in der Symbolleiste oder auf dem Menü-Eintrag soll eine neue Quiz-Folie erzeugen. Im folgenden wird beschreiben, wie das programmtechnisch gelöst wird.
Die Symbolleiste wird durch den folgenden Code erzeugt:

```
Dim classroomqiuzBar As CommandBar
Dim newQuizButton As CommandBarButton

Set classroomqiuzBar = Application.CommandBars.Add(Temporary:=True)
Set newQuizButton = classroomqiuzBar.Controls.Add(Type:=msoControlButton)
    newQuizButton.TooltipText = "erzeugt eine neue Classroom-Quiz-Folie"
    newQuizButton.Style = msoButtonIcon
    newQuizButton.FaceId = 680
    newQuizButton.OnAction = "makeNewQuizSlide"

    classroomqiuzBar.Position = msoBarTop
    classroomqiuzBar.Visible = True
```

Zuerst werden die Symbolleiste (CommandBar) und die einzige Schaltfläche (CommandBarButton) auf ihr deklariert. Powerpoint stellt seine Symbolleisten in dem Objekt CommandBars zur Verfügung, wobei durch die Prozedur Add eine neue hinzugefügt wird. In diesem Fall ist sie temporär, d.h., sie ist nur für die aktuelle Sitzung von PowerPoint da und beim Neustart von PowerPoint verschwindet sie wieder. (Warum das sein muss, wird später erklärt.) Jede Symbollesite ihrerseits stellt ihre Steuerelemente in dem Objekt Controls zur Verfügung. Hier wird mit der Prozedur Add zu der Symbolleiste eine Schaltfläche hinzugefügt, die ein Kurzinfo (TooltipText) hat und nur aus einem Icon (also ohne Text) besteht (Style = msoButtonIcon). Mit dem Attribut FaceId kann man ein Icon aus der MS Office Icons-Datenbank nehmen: Unter der Nummer 680 steht das Symbol für eine neue Folie. Das Attribut OnAction enthält (als String) den Namen von Makro-Prozedur, die ausgeführt werden muss, sobald man auf diese Schaltfläche klickt. In diesem Fall wird das Makro „makeNewQuizSlide“ ausgeführt, soweit es existiert. Am Ende wird die Symbolleiste oben (dort wo die Symbolleisten sich normalerweise auch befinden) platziert (Position = msoBarTop) und sichtbar gemacht.

Das Menü wird durch den folgenden Code erzeugt:

```
Dim ClasssroomQuizMenu As Variant
Dim NewQuizMenuItem As Variant

Set ClasssroomQuizMenu = CommandBars.ActiveMenuBar.Controls.Add(Type:= _
    msoControlPopup, Temporary:=True)
    ClasssroomQuizMenu.Caption = "Classroom-&Quiz"
Set NewQuizMenuItem = ClasssroomQuizMenu.Controls.Add(Type:= _
    msoControlButton)
    NewQuizMenuItem.TooltipText = "erzeugt eine neue Classroom-Quiz-Folie _
        "
    NewQuizMenuItem.Style = msoButtonIconAndCaption
    NewQuizMenuItem.Caption = "Neue Classroom-Quiz-&Folie"
    NewQuizMenuItem.FaceId = 680
    NewQuizMenuItem.OnAction = "makeNewQuizSlide"
```

Im Unterschied zu Symbolleisten gibt es in VBA für Powerpoint keine Variablen-Typen für Menüs, deshalb werden hier das Menü sowie der Menüeintrag als Variant deklariert, der für ein beliebiges Objekt steht. Bei Menüs muss man sich entscheiden, in welcher Menüleiste es sich befinden soll. Es ist möglich mit Hilfe von VBA eine eigene zu erzeugen, aber normalerweise nimmt man die Standard-Leiste, auf welcher bereits „Datei“, „Bearbeiten“ usw. steht. Sie ist unter dem Objekt CommandBars.ActiveMenuBar

verborgen. Mit Controls.Add kann man ein Steuerelement hinzufügen, wobei der erste Parameter hier (msoControlPopup) bedeutet, dass das ein Popup-Menü sein muss. Das Symbol „&“ in dem Attribut Caption erlaubt es, die Menü-Shortcuts zu benutzen: Dieses Menü ist beispielsweise über Alt-Q ansprechbar. Ein Menü-Eintrag ist offensichtlich das gleiche Objekt wie die Schaltfläche von der Symbolleiste, allerdings wird hier der Stil msoButtonIconAndCaption benutzt, der diesen Eintrag sowohl mit einem Icon, als auch mit einer Aufschrift verseht, die Ebenfalls Shortcuts zulässt, in diesem Falle Alt-F. Ganz analog zu der Schaltfläche der Symbolleiste kann man ein Makro diesem Eintrag zuordnen, was hier auch geschieht.

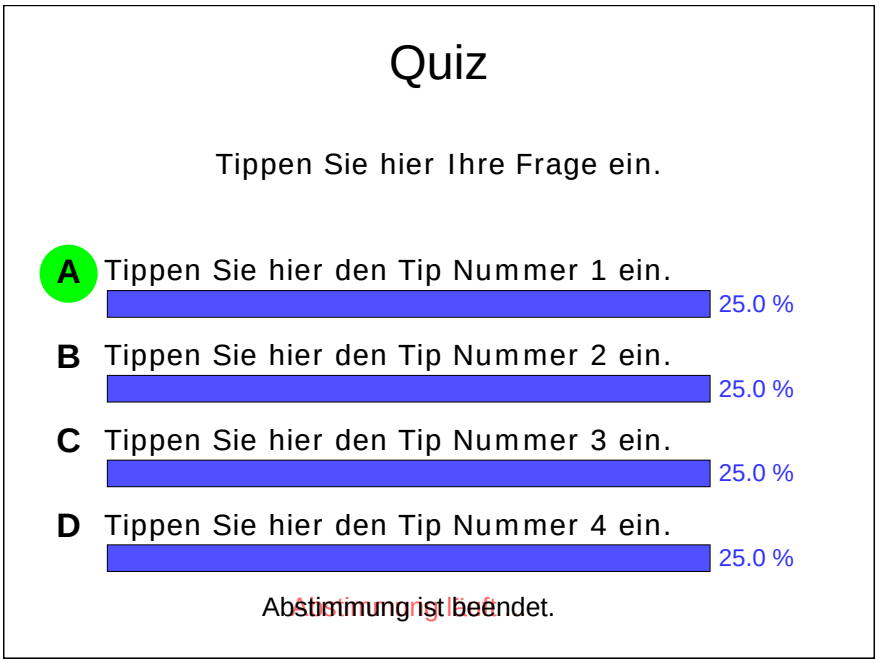
2.4 Classroom-Quiz: Arbeitsweise

Während eines Classroom-Quizes müssen einige Aktionen ausgelöst werden, zum Beispiel eine Aktion, die zur Visualisierung führt. Eine weitere Aktion sollte das Starten des Quizes kennzeichnen und mitteilen, dass die Abstimmung neu anfängt und die Häufigkeiten von null auf gezählt werden müssen.
Damit man aus dem PowerPoint ein Quiz starten und seine Ergebnisse in PowerPoint empfangen kann, muss VBA mit einem externen Programm, dass als Bluetooth-Server (siehe Abschnitt 4) dient, kommunizieren. In dieser Arbeit wurde für diesen Server die Programmiersprache „Java„ gewählt. Damit muss man von der Seite von VBA noch entscheiden, wie VBA mit einem (laufenden) Java-Programm kommuniziert.
Zuletzt müssen die Ergebnisse der Abstimmung in eine Folie „integriert“ werden, so dass man sie auf der Folie sehen kann.

2.4.1 Auslösen von Quiz-Aktionen während einer Präsentation

Damit eine Aktion ausgelöst wird, könnte man wie es im Abschnitt 2.1 eine Schaltfläche auf die Folie (beim Erzeugen, eventuell in einem Animationsschritt) platzieren und das benötigte Makro mit dieser verknüpfen. Aber das in dem vorherigen Abschnitt beschriebene Problem tritt wieder auf: Wenn das entsprechende Makro in einem Add-In „versteckt“ ist, kann diese Schaltfläche darauf nicht zugreifen.
Stattdessen kann man eine weitere (zusätzlich zu Auto_Open()) automatische Prozedur OnSlideShowNextBuild(), die beim „Durchklicken“ durch die Animationsschritte im Vollbildmodus bei jedem Klick ausgelöst wird, benutzen. Die Idee dabei ist, selbst abzuzählen, wie viele Klicks vergangen sind, um bei einem bestimmten Klick das entsprechende Makro auszuführen.

Zuerst werden noch zwei weitere Formen zu der Quiz-Folien erzeugenden Prozedur hinzugefügt, die sowohl für die bessere Erkennung vom Zeitpunkt des Starts bzw. des Stopps von der Abstimmung als auch zum automatischen Auslösen von Makro dienen:



Zuerst werden zwei Formen startQuizShape und stopQuizShape, wie vorhin erzeugt. In den Animationsschritten wird vor der Animation für den Kreis der folgende Code eingefügt:

```
startQuizShape.AnimationSettings.EntryEffect = ppEffectAppear
startQuizShape.AnimationSettings.AfterEffect = ppAfterEffectHideOnClick
stopQuizShape.AnimationSettings.EntryEffect = ppEffectAppear
stopQuizShape.AnimationSettings.AfterEffect = ppAfterEffectHideOnClick
```

Die Beduetung vom Attribut EntryEffect wurde bereits erklärt und mit AfterEffect = ppAfterEffectHideOnClick wird erreicht, dass die Form, nachdem sie nach einem Klick erscheinen ist, nach einem weiteren Klick verschwindet. Dadurch sieht man die Überlappung von beiden Formen in dem Vollbildmodus nicht.
Zusätzlich wird im Folgenden benötigt, dass eine Quiz-Folie als eine solche identifiziert und von den anderen unterschieden werden kann. Dafür kann man das Attribut Name der Folie verwenden. Dazu wird dieselbe Prozedur um die folgende Zeile ergänzt:

```
newQuizSlide.Name = newQuizSlide.Name & "_ThisIsAClassroomQuizSlide"
```

Durch das Symbol „&“ werden Strings in VBA konkateniert. D.h., das Attribut Name wird um den String „_ThisIsAClassroomQuizSlide“ ergänzt. Man kann nicht die Folie

einfach mit „ThisIsAClassroomQuizSlide“ benennen, weil dann wegen der Namenkollision nur eine einzige Quiz-Folie erzeugt werden könnte. So kann man von einer Folie durch ihren Namen erkennen, ob sie eine Quiz-Folie ist.
Jetzt kann man die automatische Prozedur OnSlideShowNextBuild() verwenden, in welcher die global deklarierte Variable step zum Zählen von Schritten benutzt wird:

```
Sub OnSlideShowNextBuild()
    With Application.ActivePresentation.SlideShowWindow.View.Slide

        If Right(.Name, 25) <> "ThisIsAClassroomQuizSlide" Then Exit Sub

        If step = 7 + .PrintSteps - 14 Then
            startQuiz
        ElseIf step = 9 + .PrintSteps - 14 Then
            evaluateQuiz
        End If

        step = step + 1

    End With
End Sub
```

Der Block With ... End With in VBA dient lediglich zur kürzerer Schreibweise: Alles was hier innerhalb von diesem Block mit dem Punkt anfängt bezieht sich auf das Objekt Application.ActivePresentation.SlideShowWindow.View.Slide, also auf die sich aktuell im Vollbildmodus befindende Folie. In der zweiten Zeile wird die Prozedur beendet, sobald die aktuelle Folie keine Quiz-Folie ist. Weiter wird in Abhängigkeit von dem aktuellen Animationsschritt des Quizes durch das Makro startQuiz gestartet, oder seine Ergebnisse durch das Makro evaluateQuiz in die Folie integriert. Wie genau das geschieht, wird im folgenden Abschnitt beschrieben.
Bei jedem Klick wird die Anzahl der Schritte eins hochgezählt. Der Ausdruck 7 + .PrintSteps - 14 wird benutzt, damit der Benutzer in der erzeugten Folie noch weitere Animationen einbauen kann. Damit insbesondere die Animationen, die zusätzliche Klicks in der Folie verursachen, die Anzahl von Schritten nicht durcheinanderbringen, wird nicht einfach die Konstante 7, die ohne solche zusätzlichen Klicks richtig wäre, benutzt, sondern wird sie um die Anzahl von zusätzlichen Klicks erhöht (gesamte Anzahl von Klicks in der Standard-Folie ist gleich 14, in der aktuellen gleich .PrintSteps). Somit erkennt das Makro, um wieviel Klicks die aktuelle Folie verglichen zum Standard mehr Klicks benötigt und verschiebt das Aufrufen von den Makros startQuiz und evaluateQuiz um

diese Anzahl nach vorne. D.h., es ist nur erlaubt die zusätzliche Klicks verursachende Animationen vor dem Textfeld „Abstimmung läuft“ zu benutzen: Nach diesem Textfeld dürfen keine weiteren zusätzlichen Klicks passieren.
Damit man mehrere Quiz-Folien in einer Präsentation benutzen kann, muss die globale Variable step bei jeder Quiz-Folie zurückgesetzt werden. Dafür kann man noch eine automatische Prozedur OnSlideShowPageChange() benutzen, die bei jedem Folienwechsel aufgerufen wird:

```
Sub OnSlideShowPageChange()
    step = 1
End Sub
```

2.4.2 Kommunikation zwischen VBA und Java

Die Prozeduren startQuiz und evaluateQuiz müssen in der Lage sein, zu einem laufenden Java-Programm ein Signal über das Starten von einem Quiz (mittels startQuiz) zu senden und das Ergebnis aus dem Java-Programm (mittels evaluateQuiz) zu lesen. Dafür stehen mehrere Methoden zur Verfügung.
Als erstes kann man ein weiteres externes Java-Client-Programm schreiben, das von VBA über Shell gestartet werden kann und das beim Starten abhängig vom dem Kommandozeilenparameter eine Start-Methode in dem Java-Server, der die Antworten von Mobiltelefonen verwaltet, aufruft oder eine Stopp-Methode, die die Ergebnisse in einer externen Datei speichert. Danach müsste man nur mit VBA diese Datei auslesen. Das Problem an dieser Methode ist, dass man dieses Speichern von einer Datei und Auslesen von derselben nicht synchronisieren kann, so dass man eventuell einen zusätzlichen Animationsschritt für die Prozedur, die das Speichern von einer Datei auslösen soll, einbauen muss und nur hoffen kann, dass der Benutzer beim „Durchklicken“ durch die Animationen nicht zu schnell klickt, damit der Java-Server genügend Zeit hat, die Ergebnisse abzuspeichern.
Eine weitere Möglichkeit ist es, das Active-X Objekt „WinSock“ zu benutzen, das über einen Port mittels TCP mit Java kommunizieren kann. Da hierfür aber eine Installation von Visual Studio benötigt wird, wurde von dieser Möglichkeit abgesehen.
Die Variante, die in dieser Arbeit verwendet wurde, ist die Kommunikation zwischen VBA und Java mittels eines Webservices. Dafür kann man in VBA das Objekt MS SOAP benutzen. Dieses Objekt gehört standardmäßig zwar nicht zu VBA, ist aber ein Bestandteil von „SOAP Toolkit 3.0“ [6], dass unter frei herunterzuladen ist.

Damit VBA auf einen Java-Webservice [6] zugreifen kann, muss dieser in einem Java-Programm gestartet werden. Der Webservice in Java ist eine einfache Klasse (hier heißt sie CQServer = Classroom-Quiz-Server), die sich auf jeden Fall in einem nicht-Default-Paket befinden und zwei spezielle Annotationen enthalten muss: @WebService markiert die Klasse als ein Webservice, während @SOAPBinding(style=Style.RPC) angibt, dass diese Klasse über SOAP-Protokoll gebunden werden soll, wobei die Kommunikation prozedurorientiert (RPC = Remote Procedure Call) ist. Der Quellcode 1 zeigt den Aufbau dieser Klasse.

Quellcode 1: Classroom-Quiz-Webservice.

```
package service;

import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style=Style.RPC)

public class CQServer {

    public void start() {

    ...

    }

    public String stop() {
        int A = 0;
        int B = 0;
        int C = 0;
        int D = 0;

        ...

        return A + ":" + B + ":" + C + ":" + D;
    }
}
```

An dieser Stelle ist noch nicht klar, was man beim Starten von einem Quiz machen muss, aber es ist offensichtlich, dass man beim Stoppen die Ergebnisse an VBA liefern muss.

Die einfachste (und in Bezug auf VBA die einzig funktionierende) Methode ist, alle Ergebnisse in einen String zu „verpacken“. Da das gesamte Ergebnis einer Abstimmung im Prinzip nur aus vier ganzen Zahlen besteht (Häufigkeiten von 'A', 'B', 'C' und 'D'), kann man diese Zahlen zu einem String über ein fest definiertes Trennsybmol (hier ':') konkatenieren, damit VBA später diesen String an diesem Trennsymbol aufsplitten kann, um diese Zahlen wieder auszulesen.
Damit diese Klasse als Webservice läuft, muss man aus einem anderen Paket (zum Beispiel aus dem Default-Paket) die folgende Zeile ausführen:

```
javax.xml.ws.Endpoint.publish("http://localhost:8080/classroomquiz", new
    service.CQServer());
```

Der Name „classroomquiz“ ist hier willkürlich gewählt. In dieser Zeile wird der Server von oben erzeugt und unter der angegebenen URL publiziert. An dieser Stelle hängt das Programm und man kann unbeschränkt „von außen“ über das SOAP-Protokoll die Methoden start() und stop() ausführen. Damit die obige Java-Methode start() ausgeführt wird, muss die Prozedur startQuiz() in VBA wie folgt aussehen:

```
Sub startQuiz()
    Dim client As MSSOAPLib30.SoapClient30
    Set client = New MSSOAPLib30.SoapClient30

    Call client.MSSoapInit(par_WSDLFile:="http://localhost:8080/" & _
                "classroomquiz?wsdl")
    Call client.start()

    Set client = Nothing
End Sub
```

Die Variable client ist ein SOAP-Client-Objekt (Version 3.0), das einfach durch die Anweisung New erzeugt wird. Mit der Funktion MSSoapInit wird das Objekt initialisiert, indem die nötigen Informationen aus der durch den CQServer publizierten WSDL(Web Services Description Language)-Datei bezogen werden (dafür wird der ursprünglich in Java benutzte String „http://localhost:8080/classroomquiz“ um „?wsdl“ ergänzt). Diese Funktion wird über „Call“ aufgerufen, um den Rückgabewert dieser Funktion zu verwerfen. Ohne dieses „Call“ würde ein Fehler auftreten, weil der Rückgabewert von niemandem angenommen wird. Mit Call client.start() wird die Java-Methode start() vom CQServer aufgerufen. Da dies eine void-Methode ist, muss man aus den gleichen Gründen wie vorhin ein „Call“ davorstellen, um einen Fehler zu vermeiden. Das Setzen auf Nothing (VBA-Analogon zu „null“ von Java) ist wünschenswert, damit der Port freigegeben wird.

Um die Methode stop() vom CQServer aufzurufen, ist nur sehr kleine Abänderung des obigen Codes nötig:

```
Sub evaluateQuiz()
    Dim str As String

    Dim client As MSSOAPLib30.SoapClient30
    Set client = New MSSOAPLib30.SoapClient30
    Call client.MSSoapInit(par_WSDLFile:="http://localhost:8080/" & _
                "classroomquiz?wsdl")
    str = client.stop()
    Set client = Nothing
End Sub
```

Man benötigt eine String-Variable, um den Rückgabewert der stop()-Methode zu speichern. Da es nicht mehr gewünscht ist, dass der Rückgabewert der Funktion verworfen wird, wird bei dem Aufruf der Methode stop() kein „Call“ davorgestellt, sondern diese Methode einfach der entsprechenden Variablen zugewiesen.

2.4.3 Visualisierung der Ergebnisse

Damit auf der Quiz-Folie beim Beenden des Quizes die Ergebnisse visualisiert werden, muss das Diagramm (also die vier Balken und Prozentzahlen) in der Prozedur evaluateQuiz() an diese Ergebnisse „angepasst“ werden.
Als erstes wird der String str aufgesplittet:

```
Dim results() As String
    results = Split(str, ":")
```

Das Array results enthält die Häufigkeiten von 'A', 'B', 'C' und 'D' als String, die jetzt zu Integers konvertiert werden können (hier und im Folgenden wird der Teil des Codes nur für 'A' dargestellt):

```
Dim A, B, C, D As Double

A = CInt(results(0))
⋮
```

Die Variablen sind als Doubles deklariert, damit die Häufigkeiten durch die entsprechenden Prozentzahlen ersetzt werden können:

```
Dim sum As Double
sum = A + B + C + D
If sum = 0 Then sum = 1        ' protects against devision by 0

A = 100 * A / sum
⋮
```

Damit der Unterschied zwischen der einzelnen Häufigkeiten deutlich wird, wird der größte Balken maximal lang gemacht und die anderen an ihn angepasst. Dafür muss zuerst das Maximum max von allen vier Variablen bestimmt werden (wobei wenn max = 0 ist, muss max = 1 gesetzt werden, um die Division durch null zu verhindern), und alle Balken müssen entsprechend vergrößert oder verkleinert werden:

```
With ActivePresentation.SlideShowWindow.View.Slide
    .Shapes("chartBarA").Width = (A / max) * .Shapes(" _
        initialBarParameters").Width
    ⋮
End With
```

In dem vorherigen Unterabschnitten wurde gezeigt, wie man eine Form über ihre Nummer anspricht (ActivePresentation.SlideShowWindow.View.Slide.Shapes(1)). Man kann sie aber auch über ihren Namen ansprechen, wofür auch die Vergabe von benutzerdefinierten Namen an die Formen vorteilhaft ist. Wenn man zum Beispiel beim Erstellen der Folie die Balken für 'A', 'B', 'C' und 'D' entsprechend „chartBarA“, „chartBarB“, ... genannt hat, kann man sie über ActivePresentation.SlideShowWindow.View.Slide.Shapes("chartBarA") usw. ansprechen. Über das Attribut Width wird die Länge des Balken angepasst. Seine Länge soll auf einen bestimmten Bruchteil von der maximalen Länge gesetzt werden. Das Problem ist, dass man die maximale Länge eventuell nicht kennt: Man kann keine konstanten Zahlen dafür verwenden, weil diese Zahlen von einem Rechner zu anderem sich unterscheiden können, weil die Breiten der Folien sich unterscheiden. Wenn die Quiz-Folie „frisch erzeugt“ ist, dann kann man sich die Länge von den Balken merken, bevor sie verändert werden, aber wenn die Quiz-Folie zum zweiten Mal benutzt wird und die Balken eventuell bereits verändert worden sind, hat man keine zuverlässige Information über die Längen. Man braucht also eine Möglichkeit, sich beim Erzeugen der Quiz-Folie irgendwo die Länge von einem der Balken (diese Parameter sind ja bei allen Balken am Anfang gleich) zu merken. Was hier benutzt wurde, ist ein zusätzlicher Balken (hier „initialBarParameters“), der in dem Attribut Width die maximale Länge enthält und durch Visible = false beim Erzeugen der Quiz-Folie unsichtbar gemacht wird. Dadurch dient

diese Form als ein „Parameter-Träger“ und wird hier als solcher benutzt: Mit (ActivePresentation.SlideShowWindow.View.Slide.Shapes("initialBarParameters").Width bekommt man die maximale Länge und durch ihre eventuelle Verkürzung die passende Länge für jeden Balken.
Die Textfelder mit Prozentzahlen müssen noch so verschoben werden, dass sie sich unmittelbar rechts von dem rechten Ende von jeweiligem Balken befinden:

```
With ActivePresentation.SlideShowWindow.View.Slide
    .Shapes("percentageOfA").Left = .Shapes("initialBarParameters").Left _
       + .Shapes("chartBarA").Width
    .Shapes("percentageOfA").TextFrame.TextRange.Text = Format(A, "#0.0") _
        & " %"
    ⋮
End With
```

Hier sind diese Textfelder als „percentageOfA“, „percentageOfB“, . . . genannt und mit der Überschreibung des Attributs Left an die gewünschte Position horizontal verschoben. Die gewünschte horizontale Position ist gleich der Länge des entsprechenden Balken, vergrößert um die horizontale Position von der oberen linken Ecke desselben Balkens. Analog zum vorherigen Schritt wird hier das Attribut Left von der „Parameter-Träger“-Form beim Erzeugen der Quiz-Folie auf die passende Zahl gesetzt und hier ausgelesen. Der Text in diesen Textfeldern muss natürlich auch angepasst werden: Über die Format-Funktion mit dem zweiten Parameter "#0.0" wird gesichert, dass die Fließkommazahlen auf die erste Dezimalstelle gerundet werden und bei Zahlen kleiner eins immer eine Null vor dem Komma steht, das Ergebnis der Formatierung wird dann mit dem Zeichen „%“ konkateniert.
Das Ergebnis dieser Visualisierung für den Fall, dass das Java Programm den String „1:2:3:4“ zurück liefert (was der Prozentverteilung 10%, 20%, 30%, 40% entspricht), wird in der Abbildung 4 gezeigt.

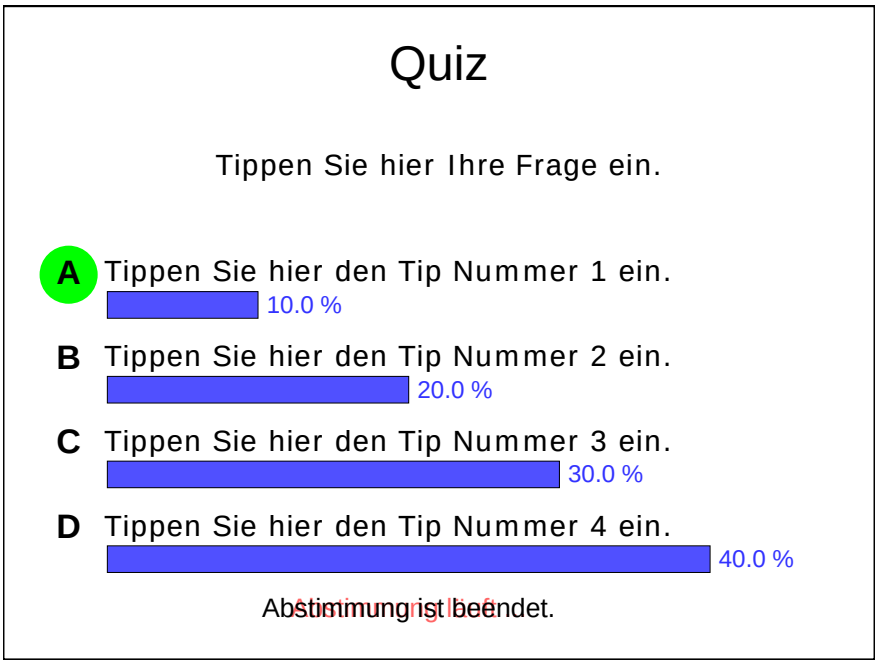


Abbildung 4: Das Ergenis einer Abstimmung auf einer Classroom-Quiz-Folie.

3 Classroom-Quiz auf Mobiltelefonen

Damit man von einem Mobiltelefon eine Nachricht (im vorliegenden Fall einfach einen Buchstaben 'A', 'B', 'C' oder 'D') an einen Rechner versenden kann, muss erstmal ein entsprechendes Programm auf dem Mobiltelefon vorhanden sein. Bei der Mobilgeräte-Programmierung muss man sich im Wesentlichen für eine der drei Sprachen entscheiden: C, C++ oder Java. Allerdings benötigen die Mobiltelefone einige Voraussetzungen, damit ein C-, oder C++-Programm auf ihnen läuft, etwa das Betriebssystem Symbian OS, welches C-Programmierung erlaubt. Da das Ziel der Arbeit war, möglichst vielen Mobiltelefonen den Einsatz zu ermöglichen, wurde Java gewählt, das auf den meisten Mobiltelefonen vorhanden ist und zusätzlich einen einfachen Umgang mit Bluetooth ermöglicht.

3.1 Java-Programme für Mobilgeräte

Die Java-Programme für die Mobiletelefone unterscheiden sich allein wegen einer anderen grafischen Oberfläche von den Java-Programmen für die Rechner. Sie unterscheiden sich nicht nur im Aussehen, sie laufen sogar auf unterschiedlichen virtuellen Maschinen und haben zum Teil unterschiedliche Klassen zur Verfügung. Die Java-Programmen haben außerdem eine besondere Arbeitsweise.

3.1.1 Java 2 Micro Edition

Trotz der Unterschiede zwischen den Java-Programmen für PCs und denen für Mobilgeräte, gehören beide Klassen von Programmen zu einer Java-Familie. Somit ist Java für die Mobiltelefone, die so genannte *J2ME* (Java 2 Micro Edition) [9][11], ein Bestandteil der Java-Plattform, deren Gliederung in der Abbildung 5 dargestellt ist.
Diese Bestandteile unterscheiden sich in erster Linie durch die Endgeräte, für die sie konzipiert sind: Während *J2EE* (Java 2 Enterprise Edition) für große Server gedacht ist und solche Technologien wie Enterprise Java Beans oder Servlets beinhaltet, ist *J2SE* (Java 2 Standard Edition) eine Umgebung für traditionelle Anwendungen auf den privaten Rechnern, die aber auch das Programmieren von einfachen Server- und Client-Anwendungen sowie Anwendungen für Web Browser erlaubt. *J2ME* (Java 2 Micro Edition) wird in den sogenannten High-End-Geräten (zum Beispiel Navigationssysteme in Autos) und Low-End-Geräten (zum Beispiel Mobiltelefone und PDAs) benutzt. Wie bereits erwähnt, wird in Mobiltelefonen anstatt von der üblichen *JVM* (Java Virtual Machine) eine andere, und zwar die *KVM* (Kilobyte Virtual Machine) benutzt. Der Grund dafür ist, dass die

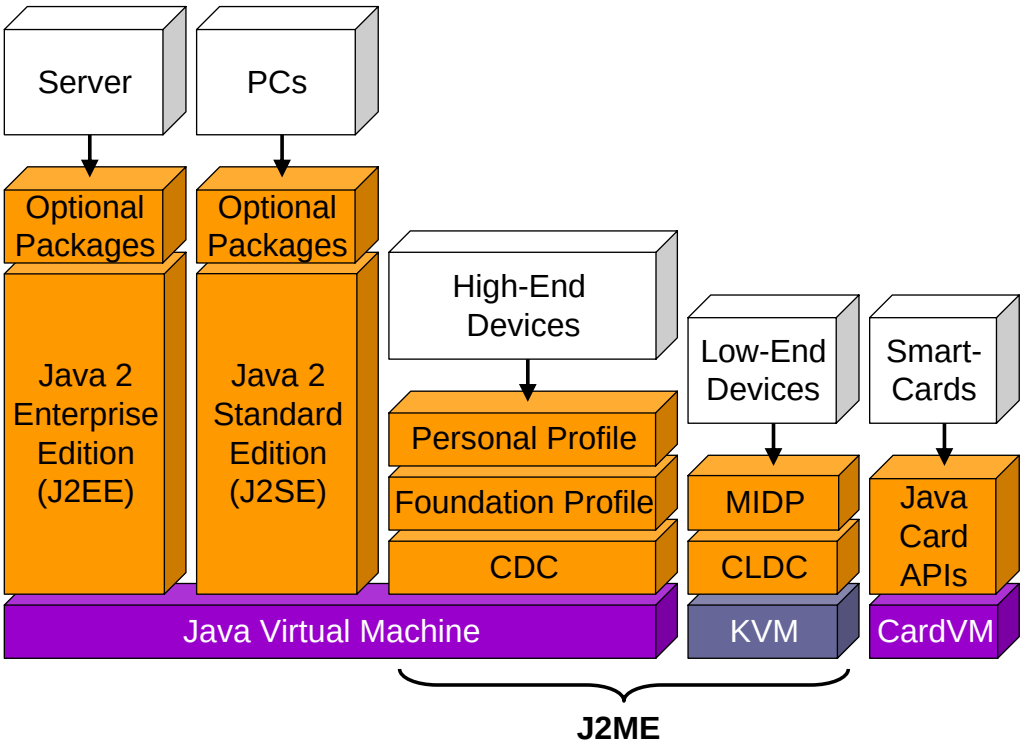


Abbildung 5: Java Editionen.

JVM einfach zu groß für solche kleinen Geräte ist, während KVM nur 128kB Speicher verlangt, darüber hinaus von Grund auf in C entwickelt und wegen dem Verzicht auf JIT (Just-In-Time-Compiler) bis zu drei mal schneller ist. Die meisten Einschränkungen von KVM werden durch die *CLDC* (Connected Limited Device Configuration) diktiert, die im Grunde ist eine (ziemlich starke) Reduktion von J2SE ist, d.h., alle Klassen von J2SE, die bei Mobiltelefonen für überflüssig, oder zu Speicher-intensiv gehalten werden, werden aus CLDC ausgeschlossen. Außer AWT fehlen in CLDC noch Java Native Interface (JNI), Thread-Gruppen and Daemon-Threads, Finalization, manche Exceptions und einiges mehr. Darüber hinaus werden (zumindest in der ersten Version von CLDC) Fließkommazahlen nicht unterstützt.
Damit ein Java-fähiges Gerät ein Java-Programm ausführen kann, muss außer CLDC zusätzlich ein *Profile*[8] in java auf dem Gerät integriert sein. Ein Profile liefert alle nötigen Klassen, die in J2SE und daher auch in CLDC nicht vorhanden sind, wie etwa Gerät-spezifische GUI-Klassen.
Java-fähige Mobiltelefone enthalten das Profile *MIDP* (Mobile Information Device Profile). Alle Java-Programme die auf MIDP basieren, werden als *MIDlets* bezeichnet.

3.1.2 Lebenszyklen von MIDlets

Wenn man die Struktur von MIDlets betrachtet, merkt man dass sie den Applets sehr ähnlich sind. Die Midlets werden auch von einer speziellen Klasse abgeleitet, der Klasse MIdlet. Ein MIDlet muss drei Methoden implementieren startApp(), pauseApp() und destroyApp(boolean) und darf einen Default-Konstruktor haben. Ein MIDlet kann in einem der drei Zustände existieren: „aktiv“, „pausiert und „beendet“ [10][11]. Die Abbildung 6 stellt den Zusammenhang zwischen diesen Methoden und Zuständen dar.
Wenn ein MIDlet erzeugt wird, geht es in den zustand „pausiert“ und wenn in diesem Zustand eine Exception ausgelöst wird, dann geht es sofort in den Zustand „beendet“, sonst wird die Methode startApp() aufgerufen und das MIDlet geht in den Zustand „aktiv“. Von diesem Zustand kann das MIDlet zu dem Zustand „pausiert“ entweder automatisch durch das System des Mobiltelefons oder durch einen impliziten Aufruf der Methode notifyPaused() zurückkehren, wobei in beiden Fällen die Methode pauseApp() automatisch aufgerufen wird. Es kann aber auch zu dem Zustand „beendet“ wieder durch das System (etwa beim Drücken auf dem Mobilgerät auf die Taste „End“) oder durch einen impliziten Aufruf von der Methode notifyDestroyed() wechseln, wobei in beiden Fällen die Methode destroyApp(boolean) ebenfalls automatisch aufgerufen wird. Das Boolean-Argument in der Methode destroyApp(boolean) hat folgende Bedeutung:

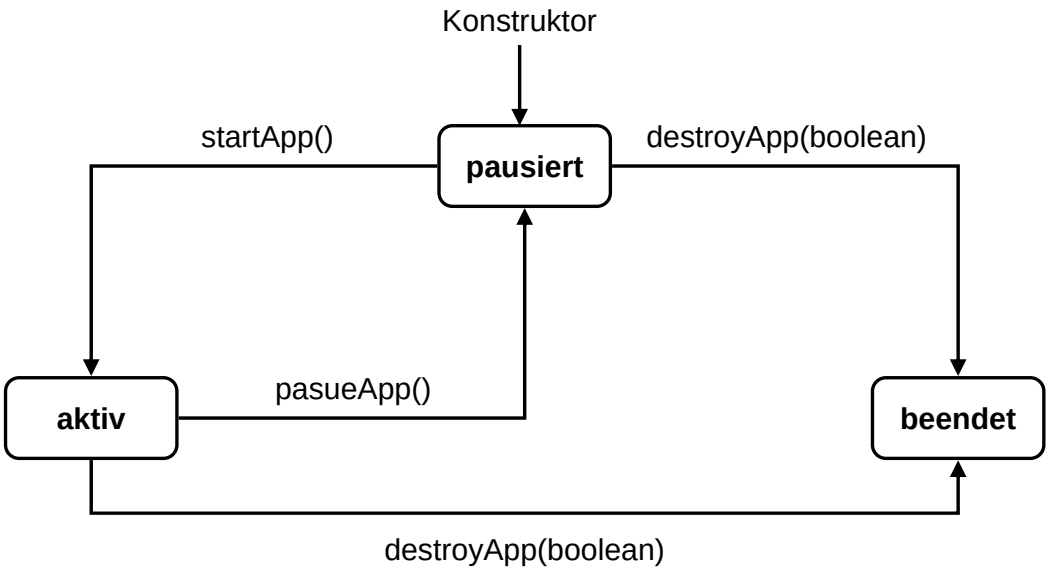


Abbildung 6: Labenszyklen eines MIDlets.

Wenn er true ist, dann wird das MIDlet keine andere Wahl haben, als alle Ressourcen zu befreien und sich selbst zu beenden, wenn er jedoch false ist, kann das MIDlet eine MIDletStateChangeException werfen, um die Beendung zu verhindern und die Ausführung fortzusetzen.
Die Methode startApp() unterscheidet sich von dem Konstruktor dadurch, dass sie mehrmals im Lebenszyklus des MIDlets aufgerufen werden kann (jedes Mal nach der Aktivierung nach dem pausierten Zustand) und man kann die beiden Methoden pauseApp() und startApp() nutzen, indem man in der ersten die Ressourcen freigibt und in der zweiten sie dann wieder beschafft.

3.1.3 MIDP GUI: LCDUI

Wie bereits erwähnt, unterscheidet sich das GUI der Mobiltelefone sehr stark von dem GUI der J2SE-Anwendungen. Dabei geht es nicht nur um unterschiedliches Aussehen, sondern tragen die geringe Bildschirmgröße der Mobiltelefone und deshalb auch sehr eingeschränkte Funktionalität der grafischen Oberfläche, sowie das Fehlen der für die J2SE üblichen Maus-Events zu der Unterscheidung bei. Diese Gründe, aber auch die Tatsache, dass AWT zu viele für Mobiltelefone überflüssige Funktionalitäten (wie zum Beispiel das Vergrößern und Verkleinern von grafischen Fenstern) beinhaltet, hat die *MIDP Expert Group* davon abgebracht, das GUI für Mobiltelefone aus AWT abzuleiten,

so dass das MIDP GUI API, das in Java-Jargon als *LCDUI*[9][11] bezeichnet wird und in dem Paket javax.microedition.lcdui enthalten ist, extra für Mobiltelefone entwickelt werden musste.
LCDUI wird in zwei weitere APIs unterteilt: in das sogenannte *High-Level API* und *Low-Level API*. Das High-Level API bietet eine (sehr begrenzte) Anzahl an vorgefertigten (grafischen) Steuerelementen (wie etwa Schaltflächen, Optionsschaltflächen, Auswahllisten) und verleiht einem MIDlet (besser gesagt, seinem Aussehen) eine große Portabilität, indem es zum Beispiel das GUI an unterschiedlich große Bildschirme anpasst. Der Preis für diese Portabilität ist eine hohe Abstraktion, die dem Programmierer viele Möglichkeiten zum Manipulieren des Aussehens des Programms nimmt (je nach dem Alter eines Mobiltelefons und entsprechend der Version des MIDP gibt es manchmal keine Möglichkeit die Form, Farbe oder Schrift eines grafischen Elements zu ändern). Darüber hinaus werden viele Events automatisch vom System bearbeitet, und das Programm erfährt nichts von ihnen. Das Low-Level API hat eine geringere Abstraktion, was dem Programmierer erlaubt die grafischen Komponenten selbst „pixelweise“ zu entwickeln und alle Events selbst zu bearbeiten. Dadurch büßt dieses API natürlich die Portabilität ein, denn die Komponenten werden nicht an den Bildschirm angepasst und die eventuell benutzten Geräte-spezifischen Mechanismen auf einem Mobiltelefon können nicht auf alle anderen Mobilgeräte übertragen werden.
Seine Einfachheit im Umgang (die Komponenten existieren bereits) und seine Portabilität waren die triftigsten Gründen dafür, in dieser Arbeit das High-Level API zu benutzen.

Die wichtigste Klasse von LCDUI ist die Klasse Display, die den Bildschirm als einen Container repräsentiert, der selbst keinen Aussehen hat, dafür aber unterschiedliche GUI-Elemente enthalten kann. Alle Elemente die auf dem Bildschirm platziert werden dürfen, müssen von der Klasse Displayable abgeleitet werden. In der Abbildung 7 wird die Klassen-Hierarchie von allen Unterklassen von Displayable dargestellt, wobei nur die Klasse Canvas für das Low-Level API steht. Die Klasse Displayable selbst ist abstrakt und hat nur wenige implementierte Methoden, eine erlaubt das Setzen des Titels. Die Klasse Screen ist ebenfalls abstrakt und stellt überhaupt keine Methoden zur Verfügung, sondern dient als eine Verallgemeinerung von allen High-Level API Klassen: Alert für unterschiedliche Mitteilungsfenster, List für Auswahllisten, TextBox für Eingabe von Texten und Form für das Integrieren von GUI-Formularen. Jede von diesen vier Element-Arten beansprucht den ganzen Bildschirm, wobei nur die Form als ein Container dienen darf

und zu welcher GUI-Elemente hinzugefügt werden können, wie Textfelder (die im Gegensatz zu TextBox nicht den ganzen Bildschirm einnehmen), Schaltflächen, Bilder usw. Darüber hinaus können zu einer Form einfache Strings hinzugefügt werden.

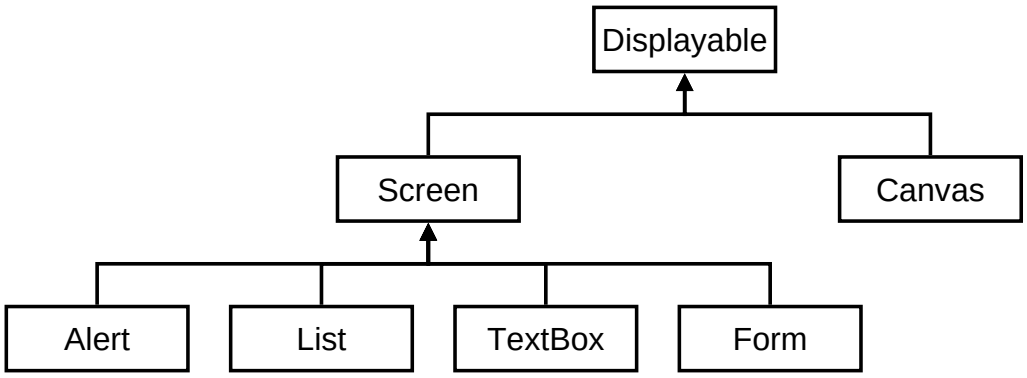


Abbildung 7: Hierarchie von „Displayables“.

In dieser Arbeit wurden nur die drei Klassen List, TextBox und Form benutzt, wobei Formen nur Strings enthalten.

3.1.4 Ein Beispiel-MIDlet

Bei der Entwicklung von Programmen für Mobiltelefone hilft das „Sun Java Wireless Toolkit for CLDC“ [12] MIDlets zu kompilieren und dank dem integrierten Emulator auch ohne Mobiltelefone zu testen.
Der folgende Code beschreibt ein kleines Beispiel, das im weiteren Erläutert wird, und die Abbildung 8 ist ein Screenshot der Emulation dieses MIDlets.

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;

public class exemplaryMIDlet extends MIDlet {

    Display display;
    Form form;

    public exemplaryMIDlet() {
        form = new Form("\"Hello, World!\" - Form");
    }

    protected void startApp() {
        if(display != null) return;

        display = Display.getDisplay(this);

        form.append("Hello, World!");

        display.setCurrent(form);
    }

    protected void pauseApp() { }

    protected void destroyApp(boolean unconditional){ }
}
```

Wie bereits erwähnt, ist die Klasse Display die wichtigste Klasse, die in jedem MIDlet, das eine GUI-Oberfläche hat, enthalten sein muss. Diese Klasse wird in der Methode startApp() instanziiert. Diese Instanziierung geschieht statisch, weil es immer nur einen Bildschirm geben kann, wobei die Methode getDisplay() ein MIDlet als Parameter erwartet, so dass ihr Parameter meistens das Schlüsselwort this ist. Die Form kann mit nur einem String als Parameter erzeugt werden, wobei dieser String zu der Überschrift der Form wird. Mit der Methode append kann man einen String zu der Form hinzufügen. Damit die Form wirklich auf dem Bildschirm auftaucht, muss sie als Parameter der Methode setCurrent des Display-Objekts übergeben werden. Die erste Zeile der Methode startApp() ist dafür da, damit diese Methode nur einmal ausgeführt wird (d.h., wenn die Anwendung eventuell aus der Pause wieder „geweckt“ wird, soll nichts passieren) - man wollte an dieser Stelle keine zusätzliche Boolean-Variable verwenden. Obwohl die-

Abbildung 8: Emulation von einem MIDlet mit Hilfe von „Sun Java Wireless Toolkit“.

se Methode nur einmal am Anfang ausgeführt werden soll, soll ihr Inhalt nicht in den Konstruktor verschoben werden. Der Grund dafür ist, dass es nicht garantiert ist, dass jedes Mobiltelefon imstande ist, die Methode Display.getDisplay(...), sowie alle anderen Display-Methoden im Konstruktor aufzurufen.

3.2 Classroom-Quiz-MIDlet

Um die Auswahl des Antwort-Buchstabens so übersichtlich wie möglich zu gestalten, wurde die Klasse List gewählt, bei welcher auf den meisten Mobiltelefonen die Selektierung ganz eindeutig ist. Wie der Bildschirm des MIDlets für das Classroom-Quiz mit

dieser Kontrolleinheit aussieht, ist in der Abbildung 9 links und seine Implementierung in dem Quellcode 2 dargestellt.

Abbildung 9: Classroom-Quiz-MIDlet.

Quellcode 2: Classroom-Quiz-MIDlet.

```
public class CQ extends MIDlet {

    private Form cqForm;
    private List answers;
    private Display display;
```

```
public CQ()
{
    cqForm = new Form("Classroom-Quiz");

    String[] strings = new String[] { "Bitte wählen Sie:",
                                      "      A", "      B",
                                      "      C", "      D" };
    answers = new List("Classroom-Quiz", Choice.IMPLICIT,
                       strings, null);
}


protected void startApp()
{
    if(display != null) return;

    display = Display.getDisplay(this);

    answers.setCommandListener(new CommandListener() {
        public void commandAction(Command c, Displayable d) {
            if(answers.getSelectedIndex() == 0) return;

            String chosen = (answers.getString(answers.
                getSelectedIndex())).trim();

            cqForm.deleteAll();
            cqForm.append("Sie haben \"" chosen
                          + "\" gewählt.\nBitte erlauben Sie ggw. die"
                          + "Verbindung und warten Sie bis Ihre"
                          + "Antwort versandt ist.\n\n");
            display.setCurrent(cqForm);

            Sender sender = new Sender(chosen);
            Thread thread = new Thread(sender);
            thread.start();
        }
    });

    display.setCurrent(answers);
}

...
```

```
    private class Sender implements Runnable
    {
        private String data;

        public Sender(String data) { this.data = data; }

        public void run()
        {
            ...
        }
    }


    protected void pauseApp() { }


    protected void destroyApp(boolean b) { }
}
```

Der Konstruktor der Klasse List hat vier Parameter: die Überschrift, die Art der Liste, ein Array von Strings, die zu Einträgen der Liste werden, und ein Array von Images. Hier hat die Liste fünf Einträge, wobei der erste („Bitte wählen Sie:“) einerseits als eine Aufforderung zum Wählen dient, und andererseits als erster Listeneintrag im Voraus gewählt ist, wodurch keine Antwort-Möglichkeit bevorzugt wird. Man kann nämlich bei einer Liste nicht erreichen, dass alle Eintrage nicht selektiert sind. Die Lücken am Anfang der letzten vier Einträge dienen lediglich zum Einrücken. Die Art der Liste ist Choice.IMPLICIT, was bedeutet, dass ein Eintrag dann selektiert ist, wenn man auf ihm mit der Kursor steht.

Die beiden ersten und die letzte Zeilen in der Methode startApp() sind bereits bekannt. Mit der Methode setCommandListener setzt man analog zu AWT oder Swing einen Listener, der die Methode commandAction implementiert. Im Fall einer Liste, wird diese Methode beim Drücken der Taste „SELECT“ ausgeführt. Als Erstes in dieser Methode muss man natürlich sicher gehen, dass nicht der erste (besser gesagt der nullte) Eintrag (also „Bitte wählen Sie:“) ausgewählt wird, und die Methode verlassen, falls dies der Fall ist, sonst wird zuerst der Inhalt des ausgewählten Eintrags, und zwar ohne die vorstehenden Lücken in dem String chosen gespeichert und danach der Inhalt des Bildschirms geändert, indem die Liste durch die Form cqForm ersetzt wird: Zuerst wird der Inhalt dieser Form sicherheitshalber gelöscht und danach wird ein Mitteilungs-String hinzugefügt und mit der bereits bekannten Methode des Bildschirms setCurrent

die Liste durch die Form ersetzt (siehe Abbildung 9 rechts). Weiter muss die eigentliche Versendung des ausgewählten Buchstabens stattfinden. Der Grund, warum dies in einem Thread passieren muss, ist, dass Java nicht garantiert (und im Prinzip das auch nie tut), dass die setCurrent-Methode, sofort den gewünschten Effekt bringt: Dieser Effekt tritt erst nach einer (sehr) kurzen Pause auf, und zwar nur dann, wenn das Programm oder zumindest das aktuelle Thread nicht beschäftigt ist. D.h., wenn man die setCurrent-Methode aufruft und unmittelbar danach etwas anderes macht, dann wird diese Methode darauf warten, bis diese andere Beschäftigung zu Ende ist, bevor sie den Inhalt des Bildschirms tatsächlich wechselt. Das würde aber bedeuten, dass man, nachdem man den Buchstaben gewählt hat, nicht sofort die Mitteilung sieht, sondern erst, wenn die Versendung abgeschlossen ist und es sowieso zu spät für eine solche Mitteilung ist. Es gibt unterschiedliche Möglichkeiten, dieses sehr bekannte Problem bei MIDlets zu lösen: Hier wird das Programm einfach in einem anderen Thread fortgesetzt, damit die setCurrent-Methode durch nichts gestört wird.
Die eigentliche Implementierung der Kommunikation mit dem Rechner wird in dem nächsten Abschnitt behandelt.

4 Kommunikation zwischen Powerpoint und MIDlets

Damit die Mobiltelefonen eine Nachricht an den Rechner schicken können, müssen sie eine drahtlose Kommunikationstechnologie unterstützen. Bei den meisten Mobiltelefonen heißt diese Technologie *Bluetooth*, die auch in MIDlets benutzt werden kann (siehe Abschnitt 4.1.3).
Wenn der Rechner diese Nachrichten annehmen soll, muss auf ihm ein bestimmter Server, der Bluetooth-Server laufen. Dieser Server ist in dieser Arbeit in Java programmiert.
Damit diese Nachrichten bei Powerpoint ankommen, muss es darüber hinaus eine Schnittstelle zwischen dem Bluetooth-Server und VBA geben.

4.1 Bluetooth

Die Bluetooth-Technologie wird von seinen Entwicklern [16] so definiert:
„Kabellose Bluetooth-Technologie ist ein Kommunikationssystem für kurze Strecken, das für das Ersetzen von Kabeln zwischen mobilen und/oder stationären elektronischen Geräten gedacht ist. Die Hauptmerkmale von der kabellosen Bluetooth-Technologie sind Robustheit, geringer Stromverbrauch und geringer Preis. ... Das Bluetooth-Kernsystem besteht aus einem Radio-Transceiver, einem Basisband und einem Protokollstapel. Dieses System bietet Dienste, die die Verbindung von Geräten und Austausch von einer Auswahl an Data-Klassen zwischen ihnen erlaubt. ... “
Hier sind einige Erklärungen zu den genannten Begriffen: Transceiver ist ein Gerät, das sowohl als Sender als auch als Empfänger dient (Transmitter + Receiver = Transceiver); Basisband ist ein Frequenzbereich, in dem sich das zu übertragene Signal befindet; Protokollstapel ist eine konzeptuelle Architektur von Netzwerk-Protokollen, die als Schichten in einem Stapel zusammengefasst werden.

4.1.1 Technische Details

Bluetooth-Geräte sind im Wesentlichen Funkgeräte, die in dem Frequenzbereich zwischen 2,402 GHZ und 2,480 GHZ arbeiten. Je nach Akku-Leistung und entsprechender Reichweite werden sie bezüglich der Leistung in drei Klassen unterteilt:

Klasse	**Max. Leistung**	**Reichweite**
Klasse 1	100 mW	100 m
Klasse 2	2,5 mW	10 m
Klasse 3	1 mW	1 m

Dabei gehören die meisten Mobiltelefone zu der zweiten Klasse, was bedeutet, dass sie die Reichweite von etwa 10 m haben. Für die Adressierung wird in allen Bluetooth-Geräten eine eindeutige 48-bit Adresse, die sogenannte Bluetooth-Adresse benutzt. Die Bluetooth-Technologie erlaubt für ein Gerät das Aufbauen von einem so genannten Piconetz (s.u.), in welchen 8 aktive und bis zu 200 passive Geräte (s.u.) verbunden sein können.

Den gleichen Frequenzbereich wie Bluetooth benutzen auch WLAN, schnurlose Telefone und Mikrowellen. Es ist also höchstwahrscheinlich, dass die Kommunikation mittels Bluetooth durch die Interferenzen mit den Wellen dieser Geräte stark gestört wird. Um dies zu vermeiden, benutzt die Bluetooth-Technologie die sogenannten Frequenz- oder Kanal-Sprünge: Das Basisband wird in 79 Kanäle, jeder 1MHz breit, unterteilt und die überzusendende Nachrichten werden in kleine Pakete zerlegt und Bluetooth-Geräte senden jedes Paket durch einen anderen Kanal, wobei dieser Kanal-Wechsel zufällig und 1600 mal pro Sekunde passiert. Dadurch wird natürlich das Risiko der Interferenz stark reduziert, denn unter diesen 79 Kanälen sind die meisten nicht gestört und falls ein gestörter „erwischt“ wird, muss nur das entsprechende Paket nachgesendet werden. Die moderne Geräte können sogar die „gestörten“ Kanäle im Vorfeld aufspüren und ausschließen, so dass keine Nachsendung von Paketen nötig ist.

Wenn zwei Bluetooth-Geräte kommunizieren und dabei die oben erklärte zufällige Frequenz-Sprünge machen, muss sichergestellt werden, dass die beiden auch „gleich springen“. Das ist der Zeitpunkt, wo die Master-Slave-Beziehung ins Spiel kommt, wobei das Master-Gerät dieses Sprung-Muster angibt und das Slave-Gerät dem gehorcht.
Anderes als etwa bei Festplatten verleiht die Master-Rolle Bluetooth-Geräten keine besondere Autorität, sondern dient nur zu dieser Synchronisation von Frequenz-Sprüngen. Ein Master kann in einem Netz 7 *aktive* Slaves und 200 *passive* Slaves an sich binden, wobei ein solches Netz als **Piconetz** bezeichnet wird. In der Abbildung 10 ist ein Piconetz schematisch dargestellt: Die grauen Slaves sind passiv, die schwarzen aktiv. Es kann auch sein, dass einige Geräte zwar nah genug zu dem Master sind und potenziell seine Slaves werden können, aber trotzdem nicht zu dem Netz gehören: Man sagt dann, dass sie sich in dem Nachbarschaft-Bereich befinden.
Außer Piconetz gibt es in der Bluetooth-Technologie auch das Konzept des sogenannten Scatternetzes, eines Netzes von Piconetzen. In einem Scatternetz kann ein Slave zu zwei verschiedenen Piconetzen gehören, wobei ein Gerät, das als Salve zu einem Piconetz

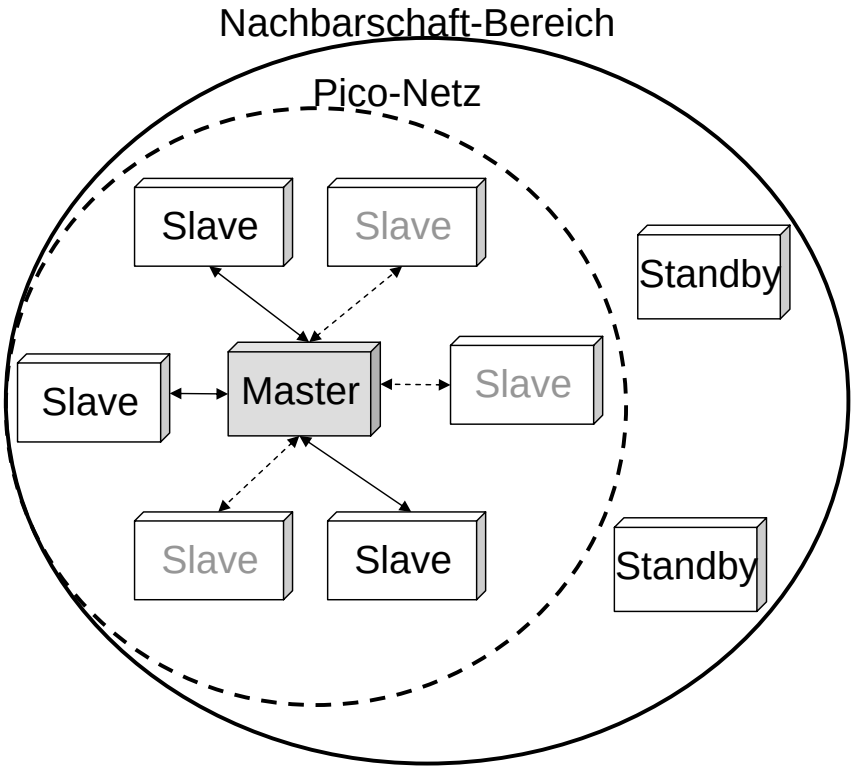


Abbildung 10: Piconetz.

zugeordnet ist, in einem anderen als Master fungieren kann. Trotz des beeindruckenden Namens bringen Scatternetze praktisch keine Vorteile und werden von den Meisten Geräten nicht unterstützt, so dass man sie ruhig außer Betracht lassen kann.
Über die Bedeutung von Piconetzen kann man sehr lange diskutieren, allerdings hat man als Programmierer (zumindest wenn man Java benutzt) keinen Einfluss auf deren Verwaltung, denn die Regelung geschieht auf der Hardware-Ebene und als Programmierer kann man lediglich versuchen, nachzuvollziehen, was in den Geräten geschieht.
In der Bluetooth-Welt wird ein Gerät, das die Verbindung initiiert, automatisch zu einem Master. Wenn man an die Frequenz-Sprünge denkt, wird das auch verständlich, denn wer als erster spricht, soll auch das Sprung-Muster angeben und deshalb zum Master werden und nur sehr wenige Geräte können (aber erst nach der Verbindung) die Rollen tauschen (so dass Master zu einem Slave wird und umgekehrt). In der Netzwerk-Sprache sind dann die Bluetooth-Clients die Master und die Bluetooth-Server Slaves.
[17][18][19]

4.1.2 Bluetooth-Verbindungen

Bei den Bluetooth-Verbindungen, so wie bei allen Netzwerk-Verbindungen müssen sowohl der Client (Initiator der Verbindung) als auch der Server (Akzeptor der Verbindung) bestimmte Schritte durchgehen, damit eine Nachricht übersendet wird. Da diese Schritte bei Clients und Servers sich unterscheiden, spricht man nicht von einer Verbindung allgemein, sonder von einer Ausgangsverbindung (die von dem Client zu Server gerichtet ist) und einer Eingangsverbindung (die sich vom Server zu Client richtet). In der Abbildung 11 sind diese Schritte schematisch dargestellt. Aus der Abbildung ist ersichtlich, dass sowohl Client als auch Server sich auf ein Transportprotokoll und eine Port-Nummer einigen müssen, während der Client noch zusätzlich wissen muss, zu welchem Gerät er die Nachricht senden muss. Diese drei Schritte (Wahl eines Zielgeräts, eines Transportprotokolls und eines Ports) werden im Folgenden beschrieben.

Wahl eines Zielgeräts

Jedes Bluetooth-Gerät hat eine eindeutige Bluetooth-Adresse und, wenn man diese kennt, kann man das Gerät direkt ansprechen. Dieser Vorgang dauert weniger als eine Sekunde. Diese Adresse wird in allen Schichten des Verbindungprozesses benutzt, von unteren, Hardware-nahen Protokoll-Ebenen bis zu den oberen, also anderes als etwa im Ethernet, wo die MAC-Adresse durch IP-Adresse ersetzt wird. Deshalb kann man

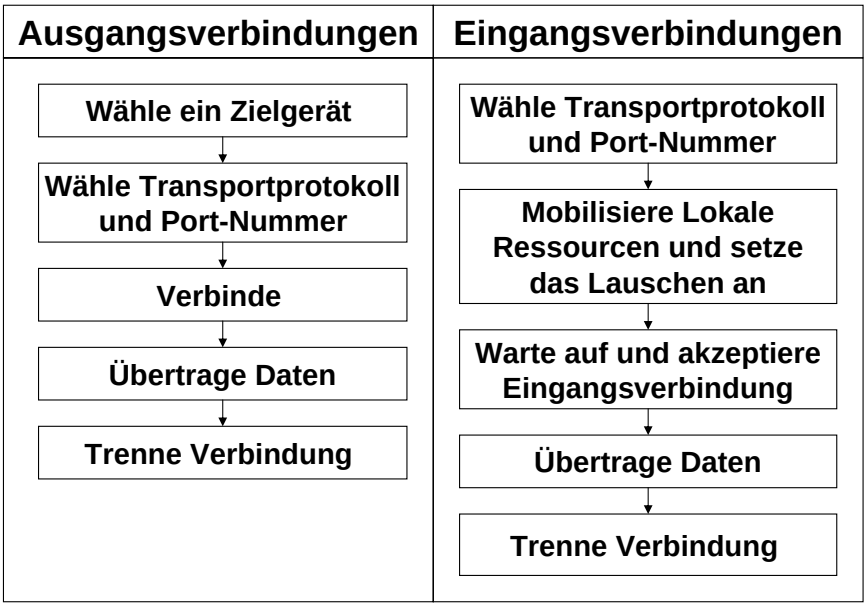


Abbildung 11: Netzwerk-Verbindungen.

diese Adresse auch in den Bluetooth-Programmen (auch in Java-Programmen) benutzen. Bluetooth erlaubt auch einen menschenfreundlichen Namen (*Friendly Name*) für jedes Bluetooth-Gerät, der von dem Benutzer dieses Geräts geändert werden kann. Allerdings kann man diesen Namen zu der Identifizierung nicht benutzen, weil er die Eindeutigkeit nicht garantiert.
Wenn man die Adresse des Geräts nicht kennt, kann man in Bluetooth eine Geräte-Anfrage (*Device Inquiry*) starten und sich zu allen Geräten in der Umgebung zu verbinden zu versuchen und erst dann überprüfen, ob das Gerät den nötigen Dienst (s.u.) anbietet. Dieser Vorgang kann allerdings bis zu 15 Sekunden dauern.

An dieser Stelle sei auf den weit verbreiteten Irrtum hingewiesen, dass ein Bluetooth-Gerät, dass den Nachbarschaft-Bereich von einem anderen betritt irgendwie „mitteilt“, dass es da ist. Das passiert nie und die einzige Möglichkeit für das zweite Gerät zu erfahren, ob ein Gerät dazugekommen ist, ist, die Geräte-Anfrage durchzuführen.

Wahl eines Transportprotokolls

Ähnlich wie beim Internet und anderen Netzwerk-Konzepten definiert die Bluetooth-Spezifikation einen Protokollstapel, in welchem nur vier Protokolle in Java ansprechbar

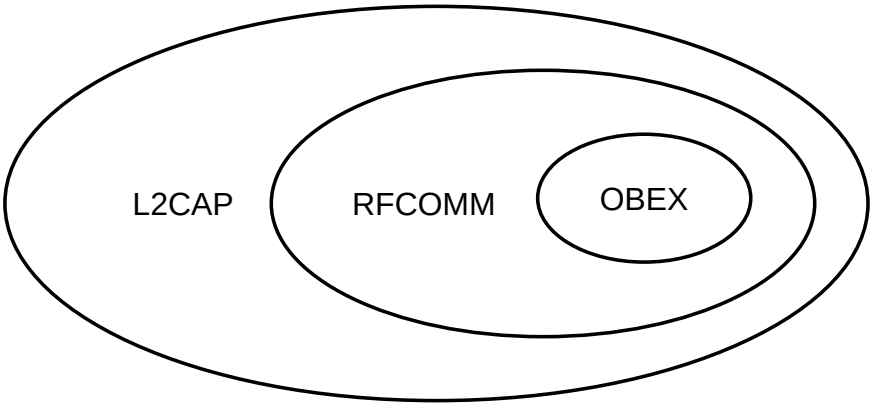


Abbildung 12: Transportprotkolle.

sind, so dass hier die Inspektion des Protokollstapels auf sie beschränkt wird. Drei dieser Protokolle, und zwar *L2CAP*, *RFCOMM* und *OBEX* sind Transportprotokolle und sind ineinander geschachtelt und zwar so, dass RFCOMM auf L2CAP basiert und OBEX auf RFCOMM. Diese Schachtelung ist in der Abbildung 12 dargestellt. Das vierte Protokoll *SDP* dient zu der Identifizierung eines Dienstes. Die Bedeutung dieser Protokolle wird im Folgenden erklärt.

L2CAP (**Logical Link Control and Adaption Protocol**) ist ein paketbasiertes Protokoll und kann daher mit dem bekannten UDP verglichen werden. Allerdings hat L2CAP nur 16'384 Ports anstatt von 65'535 bei UDP zur Verfügung. Ein Vorteil von L2CAP ist aber, dass die versandten Pakete in der ursprünglichen Reihenfolge ankommen und nicht wie bei UDP völlig vermischt sein können. Im Unterschied zu UDP, das als sehr unzuverlässig (wegen seinem „Best-Effort"-Prinzip) gilt, besitzt L2CAP verschiedene Zuverlässigkeitsmodi: Die Pakete werden entweder niemals nachgeliefert (Best-Effort), oder werden so lange nachgeliefert bis Erfolg oder Fehler eintritt (Standardeinstellung), oder werden beim Versuchen nach einer bestimmten Zeit verworfen. Standardmäßige Einstellung ist also die sicherste. Die Änderung dieser Zuverlässigkeitsstufe ist im Wesentlichen der einzige Zweck von L2CAP. Da aber L2CAP die Basis von den beiden anderen Transportprotokollen ist, bewirkt jede Änderung an diesem Protokoll auch die entsprechenden Änderungen an ihnen.
In meisten Fällen braucht man L2CAP nicht zu benutzen, weil RFCOMM und OBEX für den meisten Anforderungen genügen.

RFCOMM (**Radio Frequency Communication**) ist als ein Emulator für die serielle Schnittstelle RS-232 (COMM2 Port) gedacht. RFCOMM ist ein datenstrombasiertes Protokoll und ähnelt sich daher TCP, benutzt aber nur 30 Ports, verglichen mit 65'535 Ports bei TCP, weshalb auch keine Ports für Standardanwendungen reserviert sind. Solange man an L2CAP die als Standard eingestellte Zuverlässigkeitsstufe nicht ändert, gilt RFCOMM genauso zuverlässig wie TCP. Somit ist RFCOMM eine gute Wahl, wenn man die datenstrombasierte Datenübertragung machen will.

OBEX (**Object Exchange**) basiert auf RFCOMM und dient dazu, den Datenstrom in Objekte zu verpacken, was besonderes beim Übersenden von Dateien von Vorteil ist. Bei meisten Mobiltelefonen ist das OBEX Profile implementiert, was bedeutet, dass die Geräte grundsätzlich Datenpakete per OBEX versenden und empfangen können, allerdings gibt es darunter viele Telefone, die OBEX in Java nicht unterstützen.

Wahl eines Ports

In der Bluetooth-Welt sind die Ports anderes benannt: Bei L2CAP heißen sie *Protocol Service Multiplexer* und bei RFCOMM werden sie als *Kanäle* bezeichnet. Im Folgenden werden sie allerdings für bessere Klarheit weiterhin als Ports bezeichnet.

Wenn ein Client eine Ausgangsverbindung initiiert, muss er wissen auf welchen Port der Server lauscht. Wenn der Server eine Standardanwendung ist und einen wohldefinierten und wohlbekannten Port benutzt, dann braucht der Client nur diese Port-Nummer zu kennen. Beim Internetprotokoll TCP geht man so vor, dass man auf der Seite des Clients und der des Servers einfach eine freie Port-Nummer hartcodiert. Bei Bluetooth-Kommunikationen, insbesondere im Hinblick auf RFCOMM mit nur 30 Ports eignet sich dieses Vorgehen nicht, denn man weiß im Vorfeld nicht, welche Ports frei sind. Bluetooth löst dieses Problem mit Hilfe des SDP (*Service Discovery Protocol*). Jedes Bluetooth-Gerät besitzt einen SDP Server, der auf einen Port mit der wohlbekannten Nummer lauscht. Wenn ein Bluetooth-Server gestartet wird, registriert er seine Beschreibung und seine Port-Nummer mit dem SDP Server auf dem lokalen Gerät. (Da jeder Server einen *Dienst* (Service) anbietet, wird der Server auch als Dienst bezeichnet, wenn man von der Seite des Clients darauf schaut.) Wenn eine Bluetooth-Client-Anwendung sich mit diesem Gerät verbindet, übergibt sie die Beschreibung des Dienstes, den sie sucht, dem SDP Server und dieser liefert die Liste aller Dienste, die zu dieser Beschreibung passen,

inklusive der Port-Nummer. Damit erlaubt Bluetooth eine dynamische Zuordnung von Ports: Die Server-Anwendung kann bei seiner Registrierung einfach ersten freien Port nehmen, um die Konflikte zu vermeiden. Dieser Vorgang ist schematisch in der Abbildung 13 dargestellt: Links ist die Standardvorgehensweise, bei welcher der Client im Vorfeld weiß auf welchen Port der Server lauscht, während rechts die Funktionsweise von SDP Server geschildert ist, bei welcher der Client zuerst diese Port-Nummer bei dem SDP Server mittels eines wohlbekannten Ports nachfragen muss.

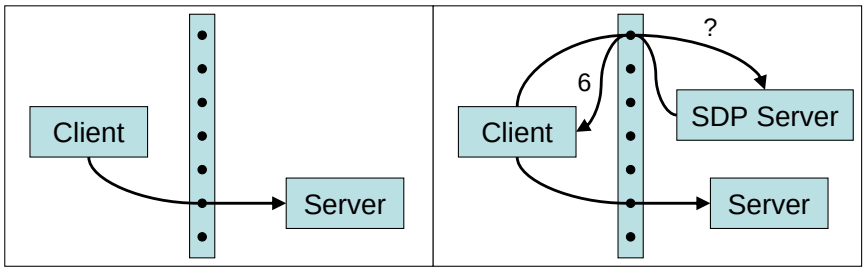


Abbildung 13: Funktionsweise von SDP Server.

Es ist klar, dass die Beschreibung des Dienstes eindeutig sein muss, damit die Client-Anwendung diesen Dienst eindeutig identifiziert. Wenn ein Bluetooth-Gerät zu einem Zeitpunkt nur einen Dienst anbietet, gibt es natürlich kein Problem, aber damit das Programm robust ist, muss es auch die Fälle berücksichtigen, bei welchen mehrere Dienste angeboten werden. Die Beschreibung des Dienstes, der sogenannte *Service Record* kann viele Einträge enthalten, inklusive des Namens des Dienstes, der bei der Server-Anwendung vergeben und zu der Identifizierung benutzt werden kann. Allerdings benutzt man meistens für die Identifizierung einen anderen Eintrag des Service Records: die *Service ID*. Als diese Service IDs werden UUIDs (Universally Unique Identifiers) benutzt, die zwar Eindeutigkeit nicht garantieren, allerdings wegen ihrer Größe (128-bit) die Wahrscheinlichkeit eines Konflikts sehr stark reduzieren. Die Idee von den Service IDs ist also die folgende: Anstatt eine Port-Nummer hartzucodieren, wird eine UUID sowohl bei dem Server als auch bei dem Client hartcodiert.
[19]

4.1.3 Bluetooth in Java

Die Java 2 Micro Edition bietet nötige Klassen und zwar im Rahmen des API *JSR-082* (Java Specification Request 82) an, damit man Bluetooth auf den Mobilgeräten, die Bluetooth eben können, in MIDlets benutzen kann: Die entsprechenden Pakete sind

javax.bluetooth und javax.obex (nicht auf allen Mobilgeräten vorhanden). (javax.obex ist kein Teil von javax.bluetooth, weil die Idee von OBEX ursprünglich von irDA importiert wurde und dadurch von Bluetooth unabhängig war und auch so bleiben soll.) Darüber hinaus benötigt man die speziellen IO-Klassen und Interfaces, die in dem Paket javax.microedition.io enthalten sind.

Um Bluetooth auf einem PC mit J2SE benutzen zu können, benötigt man ebenfalls eine Implementation der Bluetooth API. Dafür stehen einige kostenpflichtige, wie kostenlose Bibliotheken zur Verfügung, wobei in dieser Arbeit die Bibliothek *BlueCove* [20] verwendet wurde, deren Homepage auch sehr nützliche Beispiele kostenlos zum Herunterladen anbietet.

4.2 Bluetooth im Classroom-Quiz

Bei der Implementierung des Classroom-Quizes kann man im Wesentlichen zwei Vorgehensweisen wählen: Der Rechner dient als Client und die Mobiltelefone als Server, oder umgekehrt, der Rechner wird Server und die Mobiletelefone die Clients.
Im ersten Fall ist der Rechner der Master und kann bis zu 200 Mobiltelefone in einem Piconetz an sich binden, wobei nur 7 davon aktiv sein können. Das Problem dabei ist, dass der Rechner die Adressen von allen Mobiltelefonen nicht kennt, so dass er zuerst die Geräte-Anfrage starten muss, die bis 15 Sekunden dauert, dann die Liste von Geräten erstellen und später immer mit diesen Geräten arbeiten, so dass wenn ein Gerät im Laufe dazu kommt, kann es nicht teilnehmen, es sei denn man startet die Geräte-Anfrage nochmal.
In dieser Arbeit wurde für die zweite Variante entschieden: Die Mobiltelefone sind Clients und damit Master und der Rechner dient als Server. Damit kann man alle Geräte nicht in einem Piconetz zusammenbinden, sondern es können Piconetze aus höchstens zwei Geräten (1 Rechner + 1 Mobiltelefon) existieren. In dem MIDlet wird die Bluetooth-Adresse des Rechners hartcodiert, damit die Verbindung besonderes schnell ist. Die Vorgehensweise ist die folgende: Die Mobiltelefone verbindet sich der Reihe nach mit dem Rechner, versenden ihre Nachrichten und bauen die Verbindungen danach sofort ab. Wenn ein Mobiltelefon eine Verbindung zu initiieren versucht, während der Rechner beschäftigt ist, wird in dem MIDlet ein Fehler verursacht, der aber ignoriert werden kann und das Telefon versucht es solange erneut, bis es Erfolg hat. Da diese Verbindungen sehr schnell ablaufen, werden die Mobilgeräte ziemlich schnell abgearbeitet.
Da es sich bei diesen Nachrichten um einfache Strings handelt, ist hier RFCOMM von

allen Transportprotokollen die beste Wahl.
In den kommenden Abschnitten wird die konkrete Implementierung in Java sowohl des Servers als auch des Clients vorgestellt.

4.2.1 Classroom-Quiz-Bluetooth-Server

Quellcode 3: Classroom-Quiz-Bluetooth-Server

```
StreamConnectionNotifier notifier;
try {
    notifier = (StreamConnectionNotifier)
               Connector.open( "btspp://localhost:"
                             + "11111111111111111111111111111114");
} catch (IOException e) { return; }

while(true) {
    try {
        StreamConnection conn = notifier.acceptAndOpen();

        InputStream in = conn.openInputStream();

        String answer = null;
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        answer = br.readLine();

        ...

        conn.close();
    } catch (IOException e) { }
}
```

Bei den Bluetooth-Verbindungen benutzt Java die Idee von Sockets, nennt sie allerdings anderes: Das Interface Connection steht allgemein für die Sockets und von ihm werden unter anderen das Interface StreamConnectionNotifier für die lauschenden RFCOMM Sockets und das Interface StreamConnection für die verbundenen RFCOMM Sockets abgeleitet. Die Methode Connector.open liefert eine Connection, die noch auf StreamConnectionNotifier gecastet werden soll. Als das Argument für diese Methode wird ein String erwartet, der unter anderem die Information über Transportprotokolle, Bluetooth-Adresse des lokalen Geräts und die Beschreibung des Dienstes beinhaltet. Der String „btspp://localhost:11111111111111111111111111111114“ hier bedeutet, dass RFCOMM (btspp = Bluetooth Serial Port Protocol) benutzt wird, die Bluetooth-Adresse

des lokalen Geräts nicht explizit angegeben wird (localhost) und dass dieser Dienst mit der willkürlich gewählten UUID 11111111111111111111111111111114 markiert ist.
Nachdem der lauschende Socket (hier notifier) erzeugt ist, kann man mit seiner Hilfe einen verbunden Socket (hier conn) mit der Methode notifier.acceptAndOpen() erzeugen. Diese Methode blockiert das Programm solange, bis ein Client eine Verbindung zu diesem Server initiiert. Danach kann man versuchen mit der Methode conn.openInputStream() einen InputStream dem Client entgegen zu nehmen, was hier auch geschieht, wobei der Inhalt des Streams in den String answer gespeichert wird. Wenn man das Classroom-Quiz mit Hilfe von diesem Programm betreibt, dann wird der Inhalt dieser Variable „A“, „B“, „C“ oder „D“ sein.
Das ganze ist in eine unendliche Schleife verpackt, so dass der Server ununterbrochen von den Mobiltelefonen die Nachrichten empfangen kann.

4.2.2 Classroom-Quiz-Bluetooth-Client

Damit das oben in dem Quellcode 2 beschriebene Classroom-Quiz-MIDlet als Bluetooth-Client funktioniert, wird die Methode run() um die folgenden Zeilen ergänzt:

Quellcode 4: Classroom-Quiz-Bluetooth-Client.

```
RD rd = new RD(MAC);

UUID[] uuidSet = new UUID[]{new UUID("11111111111111111111111111111114",
                                     false)};

boolean done = false;

while(!done){
    try {
        SERVICE = "";
        synchronized(serviceSearchMonitor) {
            LocalDevice.getLocalDevice().getDiscoveryAgent().
                searchServices(null, uuidSet, rd, listener);
            serviceSearchMonitor.wait();
        }

        if(SERVICE.length() == 0) {
            cqForm.deleteAll();
            cqForm.append("Es besteht keine Bluetooth-Verbindung!");
            display.setCurrent(cqForm);
            Thread.sleep(5000);
            display.setCurrent(answers);
            break;
        }

        StreamConnection conn = (StreamConnection)Connector.open(SERVICE);
        OutputStream out = conn.openOutputStream();
        out.write(data.getBytes()+"\r\n");
        out.flush();
        out.close();
        done = true;
        conn.close();
    }

    catch (IOException e)          { }
    catch (InterruptedException e) { }
}

display.setCurrent(answers);
```

Da die Geräte-Anfrage zu lange dauert (siehe Abschnitt 4.1.2) wird hier der Rechner direkt bei seiner Bluetooth-Adresse angesprochen. Dafür wird diese Adresse zuerst als eine globale String-Variable MAC hartcodiert, denn in Java werden solche Adressen als einfacher String aus 12 Zeichen dargestellt, wie etwa „0123456789A“. Dann benötigt man eine Instanz der Klasse RemoteDevice, die das Zielgerät repräsentiert. Diese hat aber einen Konstruktor mit dem Modifier protected, so dass man die Instanzen dieser Klasse nicht direkt erzeugen kann, sondern eine eigene Klasse von dieser Klasse ableiten muss:

```
private class RD extends RemoteDevice {protected RD(String s){super(s);}}
```

Im Prinzip könnte man nicht nur die Bluetooth-Adresse, sondern auch die Port-Nummer hartcodieren (in meisten Fällen ist sie einfach „1“), aber damit das Programm robust ist, wird hier nach dem gewünschten Dienst auf dem Gerät gesucht. Um diesen Dienst zu finden, muss man natürlich dieselbe UUID wie beim Server benutzen. In Java gibt es die Klasse „UUID“, so dass eine UUID einfach durch den Operator new erzeugt werden kann, wobei der zweite Parameter in dem Konstruktor besagt, ob diese UUID kurz sein soll. Der Grund, warum hier ein Array von solchen UUIDs und nicht einfach eine einzelne UUID benutzt wird, ist der, dass es möglich sein soll, einen Dienst für noch mehr Eindeutigkeit nicht nur mit einer, sondern mit mehreren UUIDs zu beschreiben. Im Prinzip passiert das automatisch, dass in dem Service Record (siehe Abschnitt 4.1.2) mehrere UUIDs eingetragen werden: Wenn zum Beispiel in dem Server aus dem vorherigen Unterunerabschnitt die Zeile Connector.open("btspp://localhost:11111111111111111111111111111114"); ausgeführt wird, landet nicht nur diese UUID sondern auch eine andere, wohldefinierte UUID 0x1101 in dem Service Record, um zu symbolisieren, das hier der Dienst einen seriellen Port benutzt. Damit kann man dieses Array noch um diese UUID erweitern, was aber an dieser Stelle überflüssig wäre, weil die hier willkürlich gewählte UUID allein für die Eindeutigkeit ausreicht.
Sowohl das Zielgerät als auch die UUID werden in der Methode searchServices benutzt, die einen DiscaveryListerner (hier listener) anstößt, damit dieser den gewünschten Dienst findet. Das Konzept des DiscoveryListerner wird später noch erläutert. Die Methode searchServices kann nur von dem Objekt DiscoveryAgent ausgeführt werden, welches dem lokalen Bluetooth-Gerät zugeordnet ist. Diese Methode erwartet als Parameter ein Array von Eigenschaften des Dienstes, die von dem Service Record gelesen werden können, ein Array von UUIDs, die den gewünschten Dienst beschreiben, das Zielgerät und einen DiscoveryListerner. Die Methode searchServices blockiert das Programm nicht, deshalb ist es nötig, auf das Event, das durch den DiscoveryListerner ausgelöst wird (s.u.) zu warten,

bevor das Programm weiterläuft. Deshalb wird hier der Monitor serviceSearchMonitor, der einfach global als Objekt von Typ Object erzeugt wird, benutzt.
Der DiscaveryListerner überschreibt die global definierte String-Variable SERVICE mit der Adresse des gesuchten Dienstes (s.u.): Ein Dienst in Java ist einfach ein String, der die Bluetooth-Adresse des Zielgeräts, sowie auch die entsprechende Port-Nummer enthält (zum Beispiel „btspp://0123456789AB:1“ - somit ist der Zweck der Methode searchServices, zu überprüfen, ob die Port-Nummer wirklich „1“ ist). Wenn das Mobiltelefon nicht imstande ist, eine Bluetooth-Verbindung mit dem Server herzustellen, bleibt der String nach der Methode searchServices genauso wie er vorher war, deshalb wird im Code sein Inhalt jedes Mal vor dieser Methode „gelöscht“ und nach der Methode geprüft, ob er immer noch leer ist. Wenn dies der Fahl ist, wird die entsprechende Fehlermeldung ausgegeben und die while-Schleife abgebrochen.
Mit der bereits bekannten Methode Connector.open und diesem Dienst als Argument wird auch hier ein Socket geöffnet, wobei er hier bereits verbunden ist. Von diesem Socket wird dann ein OutputStream genommen, damit der Client auf den entsprechenden Port den String data (data ist die Antwort auf die Quiz-Frage in dem Classroom-Quiz-MIDlet) schreibt. Mit der letzten Zeile wechselt der Bildschirm wieder zu der Auswahlliste.
In dem folgenden Code ist die Implementation des DiscaveryListerner dargestellt:

Quellcode 5: DiscoverListener für die Dienst-Suche.

```
private DiscoveryListener listener = new DiscoveryListener()
{
    public void deviceDiscovered(RemoteDevice rd, DeviceClass dc) { }
    public void inquiryCompleted(int discType) { }

    public void servicesDiscovered(int transID, ServiceRecord[] sr) {
        SERVICE = null;
        for (int i = 0; i < servRecord.length && SERRVICE == null; i++) {
            SERVICE = sr[i].getConnectionURL(ServiceRecord.
                NOAUTHENTICATE_NOENCRYPT, false);
        }
    }

    public void serviceSearchCompleted(int transID, int respCode) {
        synchronized(serviceSearchMonitor){
            serviceSearchCompletedEvent.notifyAll();
        }
    }
};
```

DiscaveryListerner ist ein Interface, dessen Implementierung für zwei Sachen benutzt werden kann: für die Geräte-Anfrage (Device Discovery) oder für die Dienst-Suche (Service Discovery). Für das erste dienen die Methoden deviceDiscovered (wird aufgerufen, wenn ein Bluewtooth-Gerät in der Umgebung gefunden wird) und inquiryCompleted (wird aufgerufen, wenn die Suche nach den Geräten zu Ende ist); für das zweite dienen die Methoden servicesDiscovered (wird aufgerufen, wenn ein (oder mehrere) Dienst gefunden wird) und serviceSearchCompleted (wird aufgerufen, wenn die Suche nach den Diensten abgeschlossen ist). Da in diesem Fall keine Geräte-Anfrage stattfindet (weil die Bluetooth-Adresse hartcodiert ist), haben die ersten beiden Methoden leere Rümpfe. Der erste Parameter in den beiden Methoden servicesDiscovered und serviceSearchCompleted ist die sogenannte Transaction ID, die die Dienst-Suche identifizieren soll, die zu diesem Fund geführt hat. Da in diesem Programm höchstens eine Dienst-Suche zu jedem Zeitpunkt geschieht, wird dieser Parameter in diesem Programm auch nicht berücksichtigt. Der zweite Parameter der Methode servicesDiscovered liefert ein Array von den Service Records (also Dienst-Beschreibungen), weil es mehrere Dienste geben kann, die zu dem Array von UUIDs als dem zweiten Parameter der Methode searchServices passen. Diesem Array von Service Records wird der erste existierende Dienst entnommen, und zwar geschieht dies durch die Methode getConnectionURL des Service Records, die als Parameter die Sicherheitsstufe (hier weder Authentifizierung noch Verschlüsselung verlangt) und Aufforderung das Gerät auf jeden Fall zum Master in dem Piconetz zu machen (hier wird das nicht verlangt) und als Ergebnis den String der Art „btspp://0123456789AB:1“ zurückliefert, der in der Methode Connector.open benutzt werden kann. In der Methode serviceSearchCompleted wird lediglich die Beendung der Dienst-Suche mitgeteilt, indem der Monitor serviceSearchMonitor die Blockierung des Programms in dem Quellcode 4 aufhebt. Von der Benutzung des zweiten Parameters respCode, der mitteilt, ob die Dienst-Suche normal oder fehlerhaft abgeschlossen wurde, wurde hier abgesehen.

4.3 Schnittstelle zwischen Powerpoint und Mobiltelefonen

Bis jetzt wurde beschrieben, wie Powerpoint mit dem Java-Classroom-Quiz-Server (Quellcode 1) und ein MIDlet mit dem Java-Bluetooth-Server (Quellcode 3) kommunizieren. Was noch bleibt, ist die Erläuterung der Verknüpfung zwischen dem Java-Classroom-Quiz-Server und dem Java-Bluetooth-Server. Diese werden in einem gemeinsamen Java-Projekt vereint, wobei es offensichtlich ist, dass beide parallel laufen müssen, weil sonst der Java-Bluetooth-Server mit der Methode notifier.acceptAndOpen() das Programm ständig blockieren und die Kommunikation zwischen Powerpoint und dem Java-Classroom-

Quiz-Server unterbrechen würde. Aus diesem Grunde wird der Java-Classroom-Quiz-Server in dem Main-Thread und Java-Bluetooth-Server in einem separaten Thread gestartet.
Um die Nachrichten von Mobiltelefonen in den beiden Java-Servern zu verwalten und so zu formatieren, dass Powerpoint sie annehmen kann, wurde eine Map<String, Integer> angelegt und für beide Threads zugänglich gemacht. Diese ordnet den Antworten „A“-“D“ ihre Häufigkeit zu. Wenn ein Quiz gestartet wird, muss der Inhalt dieser Map gelöscht werden. Wenn eine Nachricht von einem Mobiltelefon ankommt, muss der entsprechende String zu dieser Map hinzugefügt werden, wobei die entsprechende Häufigkeit sich um eins erhöhen muss. Damit während eines Quizes niemand zweimal abstimmen kann, wird das MIDlet so erweitert, dass nicht nur die ausgewählte Antwort, sondern auch die eindeutige Kennung des entsprechenden Mobiltelefons mitgeliefert werden: Die beiden werden zu einem String konkateniert. Als solche Kennung eignet sich besonderes gut die Bluetooth-Adresse, die garantiert eindeutig ist und die mit der Methode LocalDevice.getLocalDevice().getBluetoothAddress() als ein String innerhalb von einem MIDlet beschafft werden kann. Auf der Server-Seite wird dann zusätzlich zu der oben genannten Map analog eine Liste List<String> angelegt, die diese Bluetooth-Adressen verwaltet. Wenn eine Nachricht bei dem Bluetooth-Server ankommt, wird zuerst überprüft, ob die Bluetooth-Adresse des sendenden Mobiltelefones nicht bereits in der Liste enthalten ist: Wenn dies der Fall ist, wird die Nachricht ignoriert, sonst wird der Buchstabe zu der Map und die Adresse zu der Liste hinzugefügt. Diese Liste muss ebenfalls bei jedem Start des Quizes gelöscht werden.
Die Abbildung 14 schildert den Aufbau des Projekts schematisch.

4.4 Erhöhung der Zuverlässigkeit

Um die Haltbarkeit des Projekts zu überprüfen, wurden zahlreiche Teste durchgeführt. Mit dem bis jetzt beschriebenen Verfahren waren die Teste mit bis zu 7 Mobiltelefonen, die gleichzeitig eine Nachricht versandten, erfolgreich. Allerdings kamm es bei der gleichzeitigen Versendung von mehr als 10 Mobiltelefonen manchmal dazu, dass das Mobilgerät zwar eine erfolgreiche Versendung mitteilte, aber bei dem Server keine Nachricht ankam. Damit die Kommunikation zuverlässiger wird, wurde das Echo-Prinzip in das MIDlet und in den Bluetooth-Server eingebaut: Wenn ein MIDlet eine Nachricht sendet, erwartet es eine Antwort von dem Server zurück und erst wenn diese Antwort kommt gibt es sich zufrieden, ansonsten sendet es die Nachricht nochmal. Das MIDlet kann sogar unterscheiden, ob der Benutzer nicht zum ersten Mal während desselben Quizes

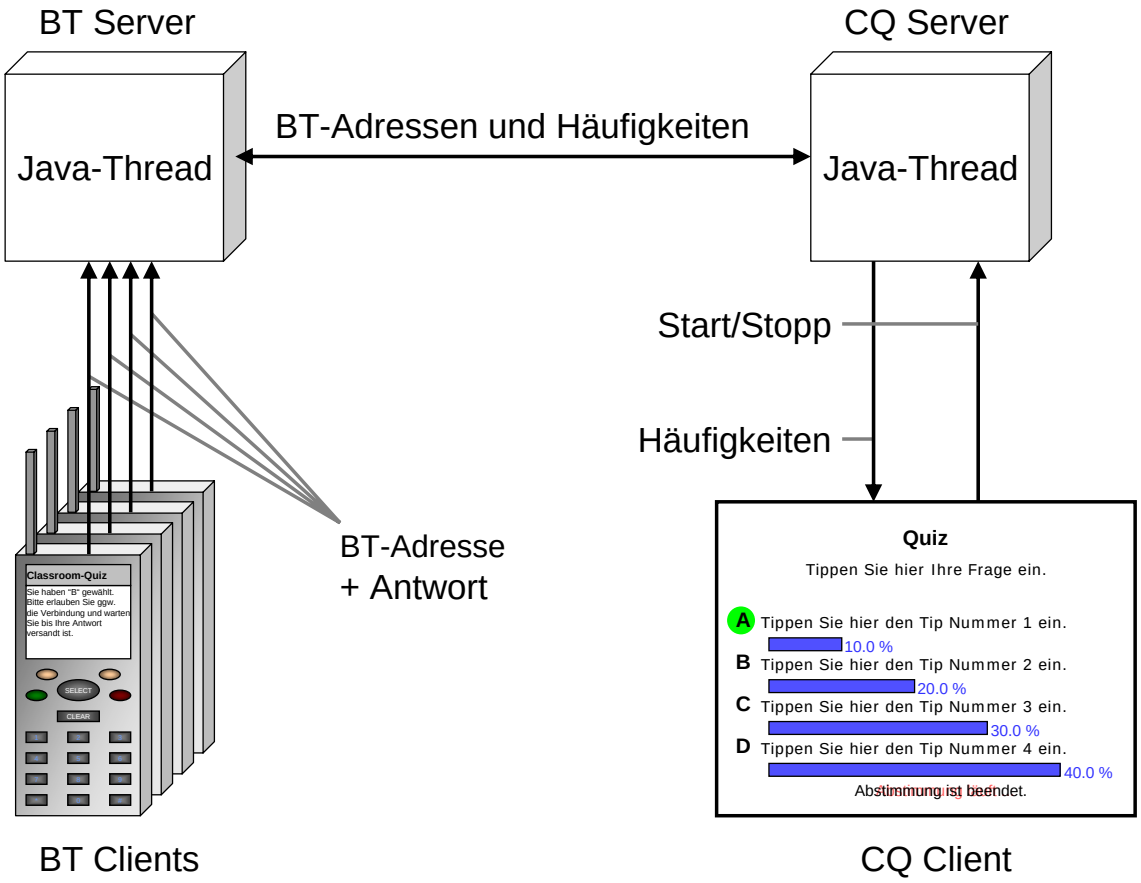


Abbildung 14: Zusammenstellung der einzelnen Komponenten.

antwortet, und entsprechende Mitteilung (dass seine Antwort nicht angenommen wird) auf den Telefon-Bildschirm ausgeben. Bei der Auswerung des Echos in MIDlet bedeutet hier 0, dass das MIDlet nochmal senden muss, 1, dass es die Mitteilung über das erfolgreiche Versenden ausgeben muss, 2, dass es warnen muss, das wiederholte Abstimmung nicht zulässig ist.
Im Folgenden werden sowohl Bluetooth-Server (Quellcode 3 als auch Bluetooth-Client (Quellcode 2) um diese Fähigkeit ergänzt, wobei im Hinblick auf Classroom-Quiz-Server auf den Fall, dass eine Nachricht von einem und demselben Mobiltelefon wiederholt ankommt, nicht eingegangen wird. Die entsprechenden Ergänzungen sind grau markiert.

Quellcode 6: Ergänzung des Classroom-Quiz-Bluetooth-Servers

```
while(true) {
    try {
        StreamConnection conn = notifier.acceptAndOpen();

        InputStream in = conn.openInputStream();
        OutputStream out = conn.openOutputStream();

        String answer = null;
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        answer = br.readLine();
        ...
        OutputStream out = conn.openOutputStream();
        if(answer == null){
            out.write(0);
        }
        else out.write(1);
        out.flush();

        conn.close();
    } catch (IOException e) { }
}
```

Zusätzlich zu dem InputStream wird auch ein OutputStream out des Sockets sonn angelegt. Mit seiner Hilfe wird mittels out.write(0) dem zurzeit mit dem Server verbundenen Gerät mitgeteilt, dass die Antowrt nicht angekommen ist, falls der String answer gleich null ist. Mit out.write(1) bekommt das Mobiltelefon Bestätigung, dass die Abstimmung erfolgreich war. Es ist wichtig, dass weder in.close(); noch out.close(); aufgerufen wird, weil sonst die Verbindung zu früh abbricht.

Quellcode 7: Ergänzung des Classroom-Quiz-MIDlets.

```
while(!done){
    try {
        SERVICE = "";
        synchronized(serviceSearchCompletedEvent) {
            LocalDevice.getLocalDevice().getDiscoveryAgent().
                searchServices(null, searchUuidSet, rd, listener);
            serviceSearchCompletedEvent.wait();
        }

        if(SERVICE.length() == 0) {
            cqForm.deleteAll();
            cqForm.append("Es besteht keine Bluetooth-Verbindung!");
            display.setCurrent(cqForm);
            Thread.sleep(5000);
            display.setCurrent(answers);
            break;
        }

        byte echo = 0;

        StreamConnection conn = (StreamConnection)Connector.open(SERVICE);
        OutputStream out = conn.openOutputStream();
        InputStream in = conn.openInputStream();

        out.write((data +"\r\n").getBytes());
        out.flush();

        for (int i = 0; i < 10 && in.available()==0; i++){
            Thread.sleep(500);
            }

        if(in.available() != 0){
            done = true;
            echo = (byte)in.read();
            if(echo == 1) {
                cqForm.append("Ihre Antwort (\""
                              + data.substring(13)
                              + "\") wurde verschickt.\n"
                              + "Bitte warten Sie bis die Wähl-Funktion"
                              + "wieder freigeschaltet wird.");
            }
```

```
            if(echo == 2) {
                cqForm.deleteAll();
                cqForm.append("SIE HABEN BEREITS ABGESTIMMT!");
            }
            display.setCurrent(cqForm);
            Thread.sleep(5000);
            display.setCurrent(answers);
        }
        else Thread.sleep(500);

        out.close();
        in.close();
        conn.close();
    }
    catch (IOException e) {
        try {
            Thread.sleep(500);
        } catch (InterruptedException e1) { }
    }
    catch (InterruptedException e) { }
}
```

Auf der Seite von MIDlet wird das von dem Server zurückkommende Echo gelesen. Die übliche Methode in.read(); lässt sich hier ohne Weiteres nicht anwenden, denn, wenn eine Nachricht von dem Mobiltelefone nicht ankommt, dann sendet der Server nichts zurück und das MIDlet hängt dann einfach an dieser Zeile unendlich, indem es auf das Echo vom Server wartet. Deshalb wird eine andere Methode zusätzlich zu read() benutzt, um sicherzugehen, dass der InputStream nicht „leer“ ist. Diese Methode heißt available() und soll die Anzahl der in dem InputStream zum Lesen bereit stehenden Bytes zurückgeben, wobei diese Methode das Programm nicht blockiert.
Somit kann man direkt nach der Zeile, in welcher das MIDlet die Nachricht in den OutputStream schreibt und danach out.flush(); aufruft, in einer Schleife, die eine kleine Pause zwischen den Schritten macht, eine bestimmte (relativ kurze) Zeit lang abfragen, ob in.available() == 0; gilt. Wenn auch nach dem Beenden der Schleife immer noch dies gilt, dann muss das MIDlet die Versendung (hier nach einer Pause von 500 ms) wiederholen, ansonsten kann es mit in.read(); das Echo in eine Variable (hier echo) lesen und entsprechend handeln: Wenn diese Variable gleich 1 ist, wird die erfolgreiche Versendung dem Mobiltelefon-Benutzer mitgeteilt, wenn sie gleich 2 ist, wird mitgeteilt, dass in dem laufenden Quiz von diesem Gerät bereits abgestimmt wurde.

Fazit und Ausblick

Das Arbeiten an jedem der drei Bestandteilen dieses Projekts (Makros für Powerpoint, MIDlets-Programmierung und Bluetooth-Kommunikation) war an einigen Stellen problematisch.
Die Erstellung von Makros für Powerpoint hat sich als sehr unangenehm erwiesen, weil viele VBA-Features, die von Enteicklern eigentlich vorgesehen sind, nicht funktionieren: Wie bereits erwähnt, kann man von Powerpoint auf innere Prozeduren eines installierten Add-Ins nicht zugreifen; einige Windows Active-X-Komponenten, wie „Winsock“ können ohne Weiteres überhaupt nicht verwendet werden; einige Powerpoint-spezifische Prozeduren und Funktionen haben keinen Effekt bei ihrer Ausführung, so kann man zum Beispiel mit Hilfe von VBA nicht erreichen, dass zwei oder mehr Formen in einem Animationsschritt gleichzeitig auf der Folie erscheinen.
Ein besonderes Problem bei VBA-Programmierung für PowerPoint ist die Tatsache, dass Powerpoint alle Fehlermeldungen unterdrückt, die im Vollbildmodus entstehen, so dass man beim Testen im Vollbildmodus gar nicht merkt, dass ein Fehler aufgetreten ist.
Im Großen und Ganzen ergibt sich der Eindruck, dass Powerpoint zwar für lokale Lösungen, wie etwa das „Aufmotzen“ von einer gegebenen Präsentation gut eignet, aber für solche Aufgaben wie Classroom-Quiz, wo die Portabilität entscheidend ist, schneidet Powerpoint nicht besonderes gut ab.

Bei Mobiltelefonen ist es aufgrund der unterschiedlichen Firmware schwierig ein MIDlet zu entwickeln, das bei allen Mobiltelefonen gleich funktioniert. Denn erstens sieht ein MIDlet bei jedem Mobiltelefon wegen unterschiedlicher Farben, Schriftarten, Rahmen usw. unterschiedlich aus und zweitens laufen nicht unbedingt die in MIDlet benutzten Features auf allen Geräten gleich. Wenn man also ein MIDlet entwickeln will, das auf möglichst vielen Mobilgeräten laufen soll, muss man sich zuerst auf das Low-Level API verzichten, aber auch bei dem High-Level API muss man so wenig Eigenschaften der GUI-Komponenten wie möglich benutzen, denn bereits die Verwendung der anderen Schriftgröße für irgendeine Schaltfläche bringt einige Mobiltelefone zum Programmfehler, während bei anderen Geräten dieses MIDlet problemlos funktioniert.
Besonderes mühsam ist das Debugging von MIDlets auf den Mobiltelefonen, denn nach dem Kompilieren von einem MIDlet muss es auf das Mobilgerät übertragen werden, was einige Bestätigungen sowohl auf dem Mobiltelefon als auch auf dem PC benötigt und deshalb ziemlich lange dauert.

Bei Bluetooth ergibt sich das oben beschriebene Problem, dass wenn viele Clients nacheinander zu einem Server eine Nachricht schicken, geht diese verloren, wodurch sich die Bluetooth-Technologie in diesem Szenario als nicht zuverlässig zeigt.

Da die oben beschriebenen Probleme zumindest für die (wenigen) zur Verfügung stehende Mobiltelefone gelöst werden konnten und ein im Großen und Ganzen funktionierendes System in dieser Arbeit entstanden ist, dessen Einsatzfähigkeit mit mit bis zu 12 Mobiltelefonen erfolgreich getestet wurde, darf man zuversichtlich sein, dass die Aufgabe dieser Arbeit gelöst worden ist.

Im Rahmen der Weiterentwicklung könnte man noch versuchen die Bluetooth-fähige Mobiltelefone mit den „WLAN-fähigen“ Desktops zu ergänzen indem der Java-Server parallel zur Verwaltung der Bluetooth-Kommunikation noch die Antworten über WLAN empfängt.

Insbesondere, wenn man an die Vorlesungen denkt, die auch „life“ on-line von den Studenten über das Internet angeschaut werden, könnte man noch einen Webserver erstellen, damit diese Studenten bei einem Quiz über das Internet auch abstimmen könnten.

In Bezug auf Bluetooth könnte man versuchen, die Master- uns Slave-Rolle bei Rechnern und Mobiltelefonen zu vertauschen, so dass in einem Piconetz der Rechner ein Client und die Telefone zu Servern werden. Man könnte dann das MIDlet um einige weitere Quizes erweitern, also gäbe es nicht nur das Quiz mit Antwortmöglichkeiten 'A'-'D', sondern auch ein „Ya/Nein-Quiz“ oder ein Quiz, in welchem man ein Wort als eine Antwort auf dem Mobiltelefon eintippen muss. Somit könnte man in einer Vorlesung diese Quizes variieren: Der Rechner schickt eine Nummer als Nachricht an die Mobiltelefone, wobei diese Nummer für eine bestimmte Quiz-Art steht, worauf das MIDlet das nötige Quiz startet und, wenn eine Antwort gewählt oder eingegeben ist, diese Antwort als Echo an den Rechner zurückschickt. Da dieses Szenario nicht getestet wurde, kann an dieser Stelle auch nicht garantiert werden, dass es funktioniert: Zuerst weiß man nicht, ob der Client so lange warten würde, bis der Server antwortet, oder selbst die Verbindung vorzeitig abbrechen würde und zweitens ist es nicht klar, wie diese Server parallel abgearbeitet werden können, denn sonst müssten die Studenten der Reihe nach Antworten, was einfach zu lange dauern würde.

Verweise

Visual Basic for Applications

[1] David Boctor
Microsoft Office2000 Visual Basic for Applications Fundamentals, 1999

[2] Reed Jacobson
Microsoft Excel 2007 Visual Basic for Applications Step by Step
Microsoft Press, 2007

[3] Paul McFedries
Visual Basic for Applications Unleashed
Sams Puslishing, 1997

Java 2 Micro Edition

[8] C. Enrique Ortiz, Eric Giguère
The Mobile Information Device Profile for Java 2 Micro Edition: Professional Developer's Guide
John Wiley & Sons, Inc., 2001
`http://www.j2medeveloper.com/midpbook`

[9] James White, David Hemphill
Java 2 Micro Edition: Java in Small Things
Manning Publications Co., 2002

[10] Michael Kroll, Stefan Haustein
Java 2 Micro Edition Application Development
Sams Publishing, 2002

[11] Martin de Jode, Jonathan Allin, Darren Holland, Alan Newman, Colin Turfus
Programming Java 2 Micro Edition on Symbian OS
John Wiley & Sons Ltd, 2004

[12] Sun Microsystems
Sun Java Wireless Toolkit for CLDC
`http://java.sun.com/products/sjwtoolkit`

Bluetooth

[16] Bluetooth SIG
Specification of the Bluetooth System, Core v2.1 , 2007
www.bluetooth.com

[17] Timothy J. Thompson, Paul J. Kline, C Bala Kumar
Bluetooth Application Programming with the Java APIs, Essentials Edition
Morgan Kaufmann Publishers, 2008

[18] Brent A. Miller, Chatschik Bisdikian
Bluetooth Revealed: The Insiders Guide to an Open Specification for Global Wireless Communications
Prentice Hall Professional Technical Reference, 2002

[19] Albert S. Huang, Larry Rudolf
Bluetooth Essentials for Programmers
Cambridge University Press, 2007

[20] Bluecove
BlueCove
http://www.bluecove.org

Webservices

[6] Microsoft
SOAP Toolkit 3.0
http://www.microsoft.com/downloads/details.aspx?familyid=C943C0DD-CEEC-4088-9753-86F052EC8450\&displaylang=en

[7] theserverside.de
WebService in Java
http://www.theserverside.de/webservice-in-java

Erklärung

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig angefertigt und keine Hilfsmittel außer denen in der Arbeit angegebenen benutzt habe.

Osnabrück, den 29.09.2008 .....

Sergiy Krutykov