

Lego Mindstorms RCX / NXT mit Java
- Installation und erste Schritte -

Semesterarbeit im Rahmen
der Weiterbildungsmaßnahme VLIN

vorgelegt von

Carsten Rohe

Georg-Büchner-Gymnasium

Hirtenweg 20

30926 Seelze

<http://www.rohe-web.de>

[mail\(at\)rohe-web.de](mailto:mail(at)rohe-web.de)

August 2008

Inhaltsverzeichnis

1	Einleitung	1
2	Treiberinstallation für den RCX / NXT	2
2.1	Benötigte Software	2
2.2	Installation	3
3	Installation von Java für den RCX	4
3.1	Benötigte Software	4
3.2	Installation	5
4	Installation von Java für den NXT	6
4.1	Benötigte Software	6
4.2	Installation	7
5	Konfiguration des Java-Editors	8
6	Erste Schritte mit Java und dem RCX	11
6.1	Ein erstes Programm	11
6.2	Kleine Befehlsübersicht	12
6.3	Programmierung eines Timers	13
7	Erste Schritte mit Java und dem NXT	15
7.1	Ein erstes Programm	15
7.2	Kleine Befehlsübersicht	16

1 Einleitung

Leg Mindstorms Roboterbaukästen bieten vielfältige Möglichkeiten für einen sinnvollen Einsatz im Informatikunterricht der Sekundarstufen I und II. Während in der Sekundarstufe I häufig „einfach“ strukturierte Programmierumgebungen wie das Lego Robotics Inventions System (RIS) oder die Programmiersprache Not Quite C (NQC) benutzt werden, bietet es sich für ältere Schülerinnen und Schüler an, die Programmiersprache Java zu benutzen. Hier steht mit den lejos-Paketen (LegoJavaOperationSystem) ein kostenloses Java-System für Lego-Roboter zur Verfügung.

Leider ist die Kombination von Java mit den Lego Robotersystemen RCX bzw. NXT nicht gerade Standard. Die Installation ist recht kompliziert. Gute *schülergerechte* Dokumentationen zur Programmierung sind rar.

In dieser Arbeit wird zunächst die Installation für die beiden Robotervarianten RCX und NXT erläutert. Beschrieben wird die Treiberinstallation, die Installation der Java-Pakete und die Konfiguration eines geeigneten Editors, dem speziell für Schulen entwickelten Editor „Java-Editor“ von Gerhard Röhner. Grundlage der Beschreibungen sind die bei Erstellung dieser Arbeit im August 2008 aktuellen Versionen der Treiber und Programme.

Anschließend folgt ein Abschnitt zu ersten Schritten der Programmierung in Java. Der grundsätzliche Aufbau eines Programmes und die wesentlichen roboterspezifischen Befehle werden beschrieben und durch kurze Beispiele näher erläutert.

2 Treiberinstallation für den RCX / NXT

2.1 Benötigte Software

Für die Kommunikation mit dem Roboterbaustein (Brick) benötigt man verschiedene Treiber. Diese sollte man zunächst vollständig über das Internet herunterladen. Da für die Programmierung in Java der Roboter über eine spezielle Firmware verfügen muss, die im Java-System enthalten ist, sollte man auch die „normale“ Lego-Firmware herunterladen, um diese gegebenenfalls wieder zurückspielen zu können.

NXT-Treiber für Windows XP (MINDSTORMS NXT Driver v1.02):
USB-Treiber für die Kommunikation mit dem NXT-Brick unter Windows XP.

Link: <http://mindstorms.lego.com/support/updates/>

NXT-USB-Treiber für die Programmierung des NXT mit Java:
Den Download nach dem richtigen Dateinamen `libusb-win32-filter-bin-0.1.12.1.exe` auswählen.

Link: <http://libusb-win32.sourceforge.net/>

aktuelle Lego-Firmware für den NXT (LEGO MINDSTORMS NXT Firmware v1.05): Standardfirmware von Lego für den NXT. Für die Programmierung in Java nicht notwendig.

Link: <http://mindstorms.lego.com/support/updates/>

RCX-Tower Treiber für Windows XP (Version: 1.64): USB-Treiber für den Tower des RCX-Bricks unter Windows XP.

Link: <http://www.lego.com/eng/service/downloads/patches/Tower164.zip>

aktuelle Lego-Firmware für den RCX: Standardfirmware von Lego für den RCX. Für die Programmierung in Java nicht notwendig.

Link: unter google nach dem Dateinamen `firm0328.lgo` suchen.

2.2 Installation

Der NXT-Treiber lässt sich ohne Probleme nach dem Entpacken der Installationsdatei durch `setup.exe` installieren.

Möchte man den NXT mit Java programmieren, benötigt man einen Treiber (`libusb`), der die Kommunikation über die USB-Schnittstelle ermöglicht. Dieser lässt sich ohne weitere Konfiguration einfach mit der Setup-Datei installieren. Wichtig: Bevor der `libusb`-Treiber installiert wird, muss der Mindstorms-Treiber installiert werden und der NXT per USB angeschlossen sein. Am besten im Geräte-Manager überprüfen, ob der NXT unter „Lego Devices“ richtig aufgeführt wird. Zusätzlich lässt sich nach der `libusb` Installation mit dem `libusb`-Tool „Test-Program“ eine Liste aller USB-Geräte anzeigen. Hier muss dann auch der Mindstorms-NXT auftauchen.

Die aktuelle Firmware kann nur übertragen werden, in dem der NXT-Brick im „Firmware-Empfangsmodus“ ist. Dazu wird das USB-Kabel abgezogen und der Reset-Knopf mindestens 4 Sekunden lang gedrückt. Der Knopf befindet sich an der Rückseite oben-links versteckt in einem „Loch“. Im richtigen Modus tickt der Brick leise. Dann das USB-Kabel wieder anschließen und die Firmware mit der entsprechenden Software übertragen.

Da für den PC der NXT im Firmware-Empfangsmodus ein anderes USB-Gerät darstellt, muss man bei eventuell auftretenden Kommunikationsfehlern den `libusb`-Treiber einfach noch einmal installieren.

Für den USB-Tower des RCX installiert man den Treiber Version 1.64. Bei der Installation werden eventuell Dateien nicht gefunden. Diese liegen dann in den Windows-Systemverzeichnissen `/system32` bzw. `/system32/drivers` oder im Installationsordner des Treibers.

3 Installation von Java für den RCX

3.1 Benötigte Software

Java SE Development Kit (JDK) (Version: 1.4.2): Für das Erzeugen von Java-Klassen (also Java-Programmen) benötigt man ein Programm zum Übersetzen des Quelltextes in eine echte Klasse. In Kombination mit dem notwendigen Lejos kann man so Lego-Roboter-Programme erzeugen. Für die Programmierung des RCX-Bricks hat sich die Version 1.4.2 als geeignet erwiesen. Aktuellere Versionen sind teilweise inkompatibel mit dem Lejos-Java-System.

Link: <http://java.sun.com/javase/downloads/index.jsp>

Für die ältere Version 1.4.2 am besten direkt nach dem Dateinamen `j2sdk-1_4_2_12-windows-i586-p.exe` googlen.

Lejos für den RCX (Version: 2.1.0): Der RCX-Baustein benötigt eine spezielle Firmware, um mit Java erstellte Programme zu empfangen und diese auszuführen. Das `LejoJavaOperationSystem` (kurz: Lejos) bringt diese Eigenschaften mit. Außerdem liefert es die notwendigen Klassenbibliotheken für das Programmieren des Roboters in Java, also die speziellen Roboter-Befehle. Das Paket ist in der Datei `lejos_win32_2_1_0.zip` komplett enthalten.

Link: <http://lejos.sourceforge.net/index.php>

Hilfdateien (APIs): Mit Hilfe der APIs (Applications Programming Interface) lassen sich für alle Befehle Informationen abrufen, in denen die genaue Syntax und Verwendung des Befehls erläutert wird. Dies gibt es zum einen für das Standard-Java (`j2sdk-1_4_2-doc.zip`), zum anderen für die speziellen Lego-RCX-Klassen über das Lejos-Paket (`lejos_win32_2_1_0.doc.zip`).

Java-Editor (aktuelle Version: 8.02b): Dieser Editor ist sehr gut für das Java-Programmieren von Lego-Robotern geeignet, da er einen speziellen Mindstorms-Modus besitzt. Die Konfiguration ist sehr einfach. Die APIs werden im Editor unterstützt. Die aktuelle Version unterstützt beide Lego-Roboter-Versionen und liegt in der Datei `javaeditor.zip` vor.

Link: <http://lernen.bildung.hessen.de/informatik/javaeditor/>

3.2 Installation

Java JDK /SDK: Zuerst wird das Java JDK installiert. Mit den Standardeinstellungen die Installation durchführen und den Pfad merken (sinnvoller Pfad: C:\Programme\Java\...).

Lejos: Die Datei `lejos_win32_2_1_0.zip` nach C:\lejos entpacken. Weitere Einstellungen werden erst im Editor vorgenommen.

API fürs JDK: Die Java-APIs `j2sdk-1_4_2-doc.zip` werden in ein Unterverzeichnis `\docs` der Java-Installation entpackt.

API für RCX-Lejos: Die RCX-lejos-APIs werden in den Ordner C:\lejos\docs entpackt. (Eventuell den eigenen Installationspfad anpassen.)

Java-Editor: Die gezippte Datei `javaeditor.zip` in irgendein Verzeichnis entpacken und dann mit `setup.exe` zum Beispiel nach C:\Programme\Javaeditor installieren. Danach können die Installationsdateien wieder gelöscht werden. Wichtig: Bei der Installation angeben, dass die Programmeinstellungen nicht in der Registry sondern in einer extra INI-Datei abgespeichert werden sollen. Dies ist notwendig, wenn man RCX- und NXT-Roboter mit dem JavaEditor programmieren möchte. Am besten installiert man den Editor dann in unterschiedlichen Ordnern doppelt und passt jeweils die entsprechende Konfiguration (siehe Seite 8) an.

4 Installation von Java für den NXT

Die Installation für den NXT ist in vielen Teilen identisch zum RCX. Da aber größtenteils andere Programmversionen benutzt werden, wird die Vorgehensweise auch für den NXT ausführlich erläutert.

4.1 Benötigte Software

Java SE Development Kit (JDK) (aktuelle Version: Java SE6 Update7): Für das Erzeugen von Java-Klassen (also Java-Programmen) benötigt man ein Programm zum Übersetzen des Quelltextes in eine echte Klasse. In Kombination mit dem notwendigen Lejos kann man so Lego-Roboter-Programme erzeugen.

Link: <http://java.sun.com/javase/downloads/index.jsp>

Lejos für den NXT (aktuelle Version: NXJ 0.6beta): Der NXT-Baustein benötigt eine spezielle Firmware, um mit Java erstellte Programme zu empfangen und diese auszuführen. Das LegoJavaOperationSystem (kurz: Lejos) bringt diese Eigenschaften mit. Außerdem liefert es die notwendigen Klassenbibliotheken für das Programmieren des Roboters in Java, also die speziellen Roboter-Befehle. Aktuell sind für den „neuen“ Legobrick NXT erst beta-Versionen erhältlich.

Link: <http://lejos.sourceforge.net/index.php>

Hilfdateien (APIs): Mit Hilfe der APIs (Applications Programming Interface) lassen sich für alle Befehle Informationen abrufen, in denen die genaue Syntax und Verwendung des Befehls erläutert wird. Dies gibt es zum einen für das Standard-Java (`jdk-6-doc.zip`), zum anderen für die speziellen Lego-RCX-Klassen über das Lejos-Paket (bereits im NXT-Lejos enthalten).

Java-Editor (aktuelle Version: 8.02b): Dieser Editor ist sehr gut für das Java-Programmieren von Lego-Robotern geeignet, da er einen speziellen Mindstorms-Modus besitzt. Die Konfiguration ist sehr einfach. Die APIs werden im Editor unterstützt. Die aktuelle Version unterstützt beide Lego-Roboter-Versionen und liegt in der Datei `javaeditor.zip` vor.

Link: <http://lernen.bildung.hessen.de/informatik/javaeditor/>

4.2 Installation

Java JDK /SDK: Zuerst wird das Java JDK (`jdk-6u7-windows-i586-p.exe`) installiert. Mit den Standardeinstellungen die Installation durchführen und den Pfad merken (sinnvoller Pfad: `C:\Programme\Java\...`).

Lejos: Die Datei `lejos_NXJ_win32_0_6_0beta.zip` nach `C:\lejos` entpacken. Weitere Einstellungen werden erst im Editor vorgenommen.

API fürs JDK: Die Java-APIs `j2sdk-1_4_2-doc.zip` werden in ein Unterverzeichnis `\docs` der Java-Installation entpackt.

API für RCX-Lejos: Die lejos-APIs sind bereits im lejos-Paket enthalten und sollten im Unterordner `\docs\apidocs` liegen.

Java-Editor: Die gezippte Datei `javaeditor.zip` in irgendein Verzeichnis entpacken und dann mit `setup.exe` zum Beispiel nach `C:\Programme\Javaeditor` installieren. Danach können die Installationsdateien wieder gelöscht werden. Wichtig: Bei der Installation angeben, dass die Programmeinstellungen nicht in der Registry sondern in einer extra INI-Datei abgespeichert werden sollen. Dies ist notwendig, wenn man RCX- und NXT-Roboter mit dem JavaEditor programmieren möchte. Am besten installiert man den Editor dann in unterschiedlichen Ordnern doppelt und passt jeweils die entsprechende Konfiguration (siehe Seite 8) an.

5 Konfiguration des Java-Editors

Nach erfolgreicher Installation der benötigten Treiber und Java-Pakete muss noch die Programmierumgebung konfiguriert werden. Der speziell für Schulen entwickelten Editor „Java-Editor“ von Gerhard Röhner bietet genau die gewünschten Eigenschaften. Da man den Editor direkt auf einen „Mindstorms-Modus“ einstellen kann, ist der Konfigurationsaufwand sehr gering. Zudem lassen sich über die Symbolleiste direkt Programme kompilieren und auf den Brick übertragen. Die Benutzeroberfläche ist sehr übersichtlich und gut strukturiert.

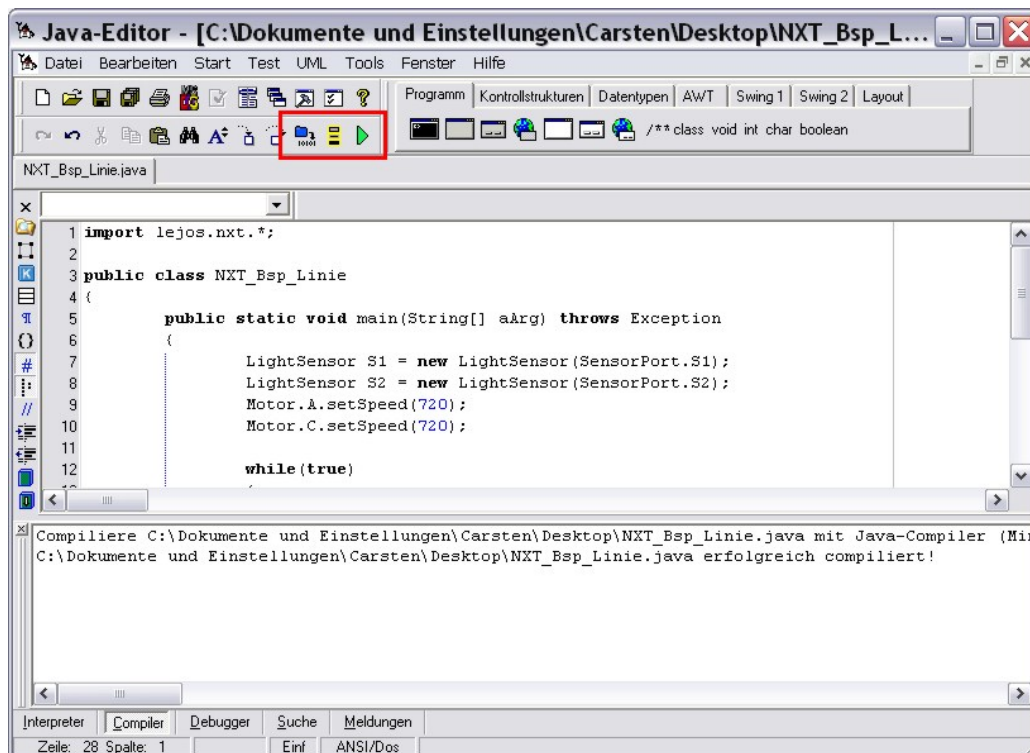


Abbildung 1: Die Benutzeroberfläche des Java-Editors ist gut strukturiert. Durch die Einbindung in die Symbolleiste lassen sich Programme direkt an den Roboter übertragen.

Beim RCX und NXT sind die Einstellungen durch die verschiedenen Java-Pakete und lejos-Versionen etwas unterschiedlich. Daher sind die Angaben entsprechend anzupassen. Im Folgenden wird die Konfiguration für den NXT erläutert.

Nach den Installationen sollte der Rechner einmal neu gestartet werden.

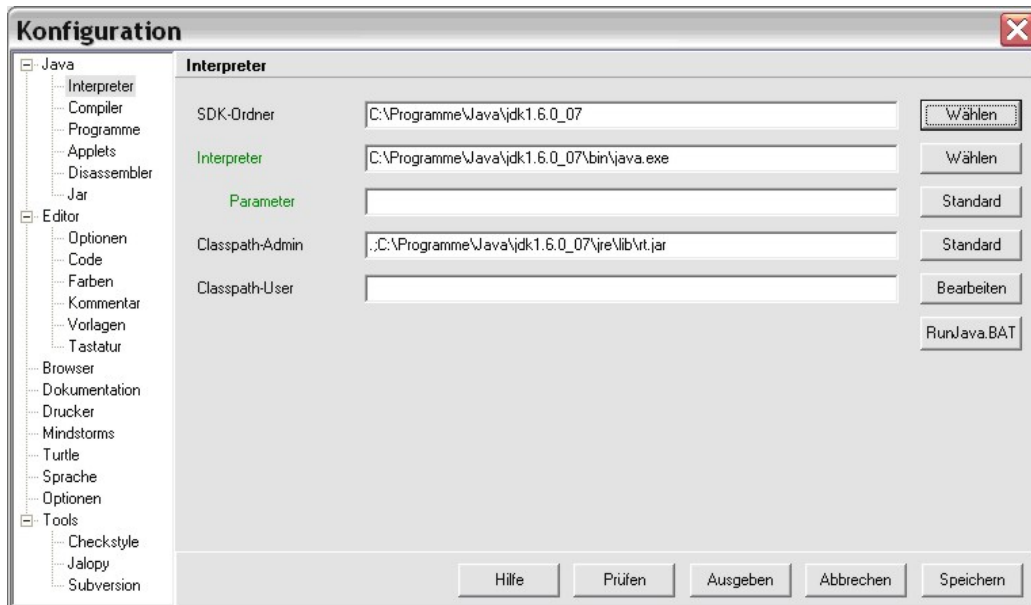


Abbildung 2: Einstellungen für die benutzte Java-Version: Die Pfadangaben sind entsprechend anzupassen. Durch die Eingabe des SDK-Ordners werden die anderen Pfade automatisch richtig hinzugefügt.

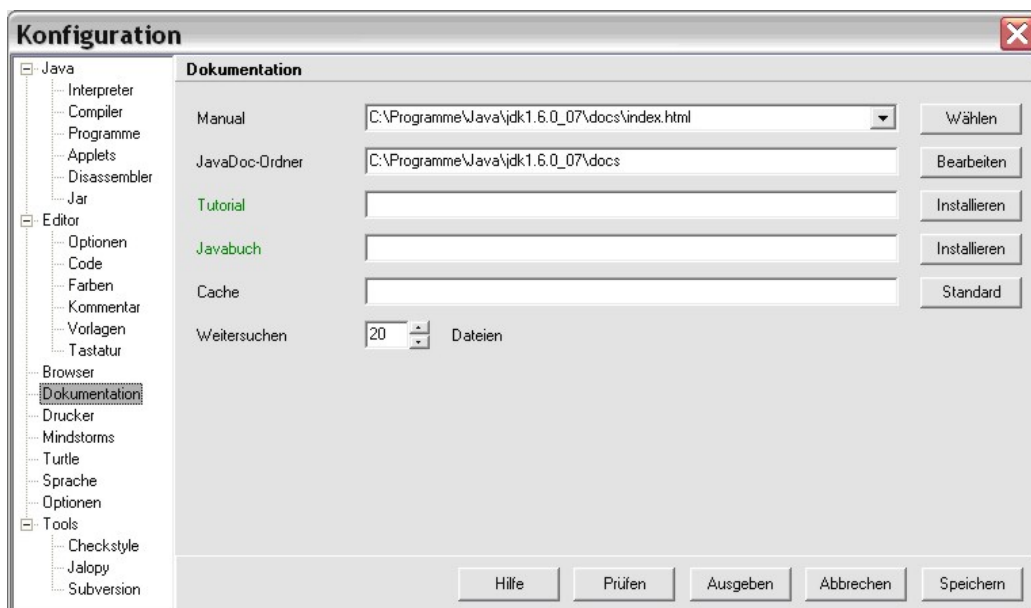


Abbildung 3: Einstellungen für die Dokumentationen / APIs: Die Pfadangaben sind hier ebenfalls entsprechend anzupassen.

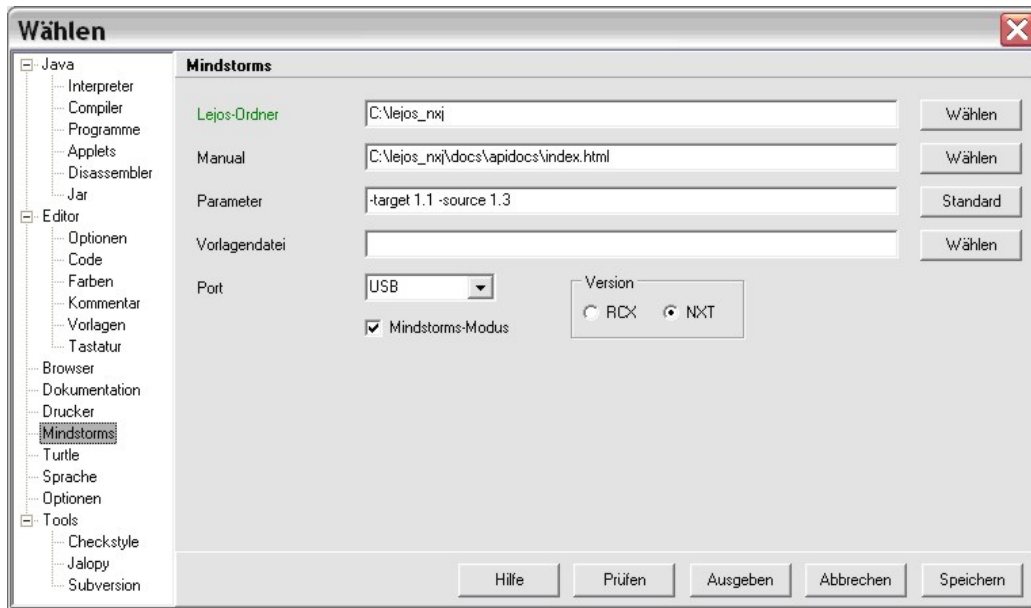


Abbildung 4: Einstellungen für das „Lego-Java“ lejos: Die Pfadangaben sind entsprechend anzupassen. Die Parameter werden automatisch angepasst. Wichtig ist die Auswahl des richtigen Mindstorms-Modells und des Mindstorms-Modus.

Nun das Programm „JavaEditor“ aufrufen und über das Menü **Fenster** den Punkt **Konfiguration** öffnen. Hier werden alle Einstellungen getätigt.

Jetzt sollte das System fertig konfiguriert sein. Zur Sicherheit kann man Rechner den noch einmal neu starten. Als ersten Test sollte man die Lejos-Firmware auf den RCX / NXT übertragen. Dies geht über das gelbe Symbol in der Leiste (Download Lejos Firmware). Die Datei für das Update wird vom Programm ohne Pfadangabe direkt gefunden. Beim NXT muss vor dem Übertragen zunächst der Resetknopf auf der Rückseite gedrückt werden (siehe Seite 3).

6 Erste Schritte mit Java und dem RCX

6.1 Ein erstes Programm

Durch das folgende Programm wird an Anschluss S1 ein Lichtsensor angemeldet. Der Motor an Ausgang A wird auf „vorwärts“ eingestellt. In einer ständigen Wiederholung wird in der Anzeige der Wert des Lichtsensors angezeigt. Der Motor wird gestoppt, sobald der Lichtsensor einen bestimmten Wert (hier 60) überschreitet. (Erläuterungen zu den roboterspezifischen Befehlen auf Seite 12.)

```
import josx.platform.rcx.*;
public class StopLinie
{
    public static void main(String[] aArg) throws Exception
    {
        Sensor.S1.setTypeAndMode(3,0x80);
        Sensor.S1.activate();
        Motor.A.setPower(3);
        Motor.A.forward();
        while(true)
        {
            LCD.showNumber(Sensor.S1.readValue());
            if(Sensor.S1.readValue()>60)
            {
                Motor.A.stop;
            }
        } // Ende von while(true)
    } //Ende von main
}
```

Durch `import josx.platform.rcx.*` werden die Klassen geladen, in denen die roboterspezifischen Befehle (Klassen) abgelegt sind. Das eigentliche Programm ist durch eine Java-Klasse mit dem Namen `StopLinie` definiert, daher ist auch der Dateiname als `StopLinie.java` zu wählen! Der Aufruf der Hauptmethode lautet `public static void main(String[] aArg) throws Exception`. Diese Methode wird beim Start des Programmes ausgeführt und ist vergleichbar mit dem `task main()` in Not Quite C (NQC).

Ein minimaler Programmrumpf ist folgender:

```
import josx.platform.rcx.*;
public class NixZuTun
{
    public static void main(String [] aArg) throws Exception
    {
        // Hier kämen die Anweisungen hin.
    }
}
```

6.2 Kleine Befehlsübersicht

6.2.1 Motoren

Allgemeine Syntax:

Motor.A.XXXX (steuert den Motor an Ausgang A an)

Wichtige Befehle:

forward()	vorwärts
backward()	rückwärts
setPower(int aPower)	Kraft einstellen (Werte von 0 bis 7)
stop()	anhalten
isForward()	liefert „true“ bei vorwärts
flt()	Leerlauf

6.2.2 Sensoren

Allgemeine Konfiguration:

Sensor.S1.XXXX (steuert den Sensor an Eingang 1 an)

Wichtige Befehle:

activate()	aktivieren
readValue()	liefert den aktuellen Wert
setTypeAndMode(int aType, int aMode)	Typ und Modus einstellen *)
readRawValue()	liefert den aktuellen raw-Wert

*) aType: 0=RAW, 1=TOUCH, 2=TEMP, 3=LIGHT, 4=ROTATION

aMode: 0x00=RAW, 0x20=BOOL, 0x40=EDGE, 0x60=PULSE, 0x80=Percent, 0xA0=DEGC, 0xC0=DEGF, 0xE0=ANGLE

Details zu allen Befehlen kann man im Lejos-API nachlesen. Dieses kann man im Java-Editor im Menü durch Hilfe/Mindstorms aufrufen. Besonders hilfreich ist das Studium von Beispielprogrammen, an denen man sehr gut die Verwendung der einzelnen Befehle und deren Verknüpfung ablesen kann.

6.3 Programmierung eines Timers

Am folgenden Beispiel soll noch einmal die prinzipielle Programmierung deutlich werden. Die Programmierung eines Timers verwendet die Prinzipien von Java und ist für ungeübte Programmierer nicht ganz einfach zu verstehen. Im Gegensatz zu einer einfachen Sprache wie Not Quite C steht in Java nicht direkt ein Timer zur Verfügung, der einfach mit `Timer(a)`; aufgerufen werden kann.

```
import josx.platform.rcx.*;
import josx.util.*;
public class TimerTest implements TimerListener
{
    public static void main(String [] aArg) throws Exception
    {
        LCD.clear();
        TextLCD.print("Timer-Test");
        Timer meinTimer = new Timer(1000, new TimerTest());
        meinTimer.start();
        // Hier noch zusätzliche Befehle...
        meinTimer.stop();
    }
    public void timedOut()
    {
        Sound.beep();
    }
}
```

Der Timer wird als Objekt der Klasse `Timer` erzeugt. Im angegebenen Programm ist dies der Timer `meinTimer`, der durch den Befehl `Timer meinTimer = new Timer(1000, new TimerTest());` entsteht. Dabei werden als Übergabewerte an die Konstruktormethode ein Zeitintervall in Millisekunden und das auf den Timer zu reagierende Objekt benannt. Die entsprechenden Klassen für den Timer werden über `import josx.util.*;` eingebunden. Im vorliegenden Beispiel „tickt“ der Timer im 1000ms Takt, also jede Sekunde.

`TimerTest` soll auf den Timer reagieren, das ist in diesem Fall das Programm selbst. Damit ein Objekt auch auf den Timer reagiert, muss `implements TimerListener` angegeben werden. In der Methode `timedOut()` wird festgelegt, was getan werden soll, wenn der Timer sich meldet. Hier würde jedes Mal ein kurzer Ton abgespielt werden, insgesamt also im Sekundentakt. Der Timer wird über die Methoden `start()` bzw. `stop()` gestartet bzw. angehalten.

7 Erste Schritte mit Java und dem NXT

7.1 Ein erstes Programm

Durch das folgende Programm wird an den Anschlüssen S1 und S2 jeweils ein Lichtsensor angemeldet. Die Motoren an den Ausgängen A und C werden auf eine Geschwindigkeit von 720°/s eingestellt, also 2 Umdrehungen pro Sekunde. In einer ständigen Wiederholung werden die Lichtsensoren überprüft und die Drehrichtung der Motoren entsprechend angepasst. Prinzipiell ließe sich so ein Roboter auf einer vorgegebenen Linie sensorgesteuert fahren. (Erläuterungen zu den roboterspezifischen Befehlen auf Seite 16).

```
import lejos.nxt.*;
public class NXTBspLinie
{
    public static void main(String[] aArg) throws Exception
    {
        LightSensor S1 = new LightSensor(SensorPort.S1);
        LightSensor S2 = new LightSensor(SensorPort.S2);
        Motor.A.setSpeed(720);
        Motor.C.setSpeed(720);
        while(true)
        {
            if(S1.readNormalizedValue() < 450)
                Motor.A.backward();
            else
                Motor.A.forward();

            if(S2.readNormalizedValue() < 450)
                Motor.C.backward();
            else
                Motor.C.forward();
        } // Ende von while(true)
    } //Ende von main
}
```

Durch `import lejos.nxt.*;` werden die Klassen geladen, in denen die roboterspezifischen Befehle abgelegt sind. Das eigentliche Programm ist durch eine Java-Klasse mit dem Namen „NXTBspLinie“ definiert, daher ist auch der Dateiname als `NXTBspLinie.java` zu wählen! Die Hauptmethode `main` wird über `public static void main(String[] aArg) throws Exception`

aufgerufen. Diese wird beim Start des Programms ausgeführt und ist vergleichbar mit dem `task main()` in Not Quite C (NQC).

Ein minimaler Programmrumpf ist folgender:

```
import lejos.nxt.*;
public class NixZuTun
{
    public static void main(String[] aArg) throws Exception
    {
        // Hier kämen die Anweisungen hin.
    }
}
```

7.2 Kleine Befehlsübersicht

7.2.1 Motoren

Allgemeine Syntax:

`Motor.A.XXXX` (steuert den Motor an Ausgang A an)

Wichtige Befehle:

<code>forward()</code>	vorwärts
<code>backward()</code>	rückwärts
<code>setSpeed(int speed)</code>	Geschwindigkeit einstellen (Grad pro Sekunde)
<code>stop()</code>	anhalten
<code>isForward()</code>	liefert „true“ bei vorwärts
<code>flt()</code>	Leerlauf

7.2.2 Sensoren

Allgemeine Konfiguration:

```
LightSensor Sens1 = new LightSensor(SensorPort.S1);
```

Am Port1 ist ein Lichtsensor angeschlossen, der unter der Bezeichnung Sens1 ansprechbar ist.

```
TouchSensor Bumms = new TouchSensor(SensorPort.S2);
```

Am Port2 ist ein Berührungssensor angeschlossen, der unter der Bezeichnung Bumms ansprechbar ist.

```
ColorSensor FarbSens = new ColorSensor(SensorPort.S3);
```

Am Port3 ist ein Farbsensor mit der Bezeichnung FarbSens angeschlossen.

Wichtige Befehle:

<code>readNormalizedValue()</code>	liefert den aktuellen Wert eines Lichtsensors
<code>readValue()</code>	liefert einen normierten Wert, der prozentual zwischen einem min-Wert <code>calibrateLow()</code> und einem max-Wert <code>calibrateHigh()</code> liegt. Diese Fixpunkte kann man selbst festlegen.
<code>isPressed()</code>	liefert zurück, ob der Tastsensor gedrückt ist
<code>getGreen()</code>	liefert den Grünwert(0..255) bei einem Farbsensor

Details zu allen Befehlen kann man im Lejos-API nachlesen. Dieses kann man im Java-Editor im Menü durch Hilfe/Mindstorms aufrufen. Besonders hilfreich ist das Studium von Beispielpogrammen, an denen man sehr gut die Verwendung der einzelnen Befehle und deren Verknüpfung ablesen kann.