hochschule mannheim

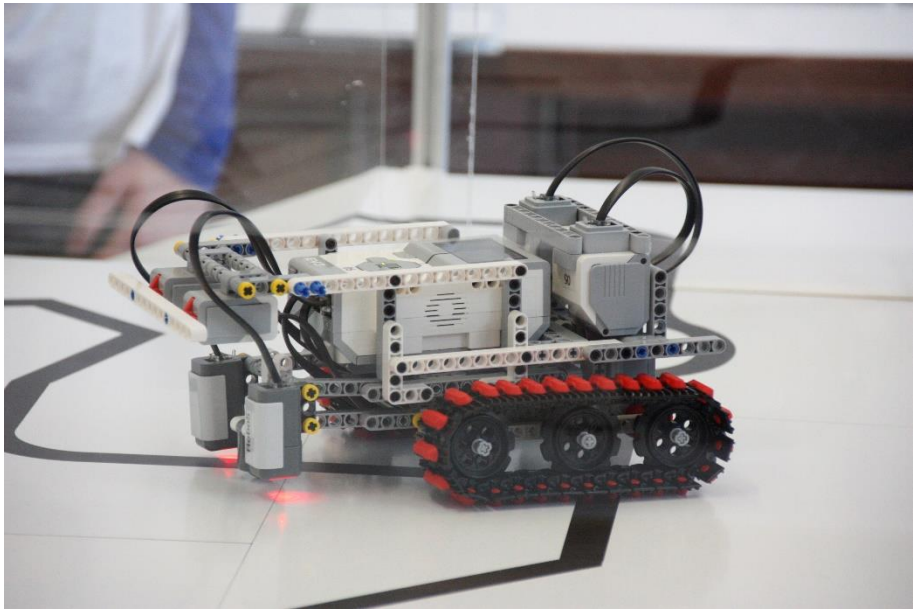
Fakultät für Informatik, Institut für Robotik

# Laborpraktikum I
# Legorobotik in JAVA – EV3

Ute Ihme

# DAS LEGO® MINDSTORMS® System

## Das EV3 System



**Prinzip von LEGO® MINDSTORMS®**

- Roboter wird gebaut mit
  - programmierbarem LEGO® Stein
  - bis zu 4 Motoren oder Lampen
  - bis zu 4 Sensoren
  - LEGO® TECHNIC Teile
- Erstellung eines Steuerprogramms am Computer
- Übertragen des Programms auf den Roboter
- Testen des Programms

# DAS LEGO® MINDSTORMS® System

## Motoren

Quelle: Lego

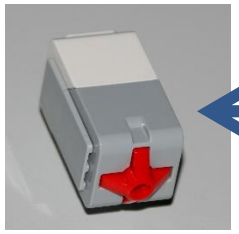
Motoren werden an die **Anschlüsse A, B, C und D** angeschlossen.

### Servomotor

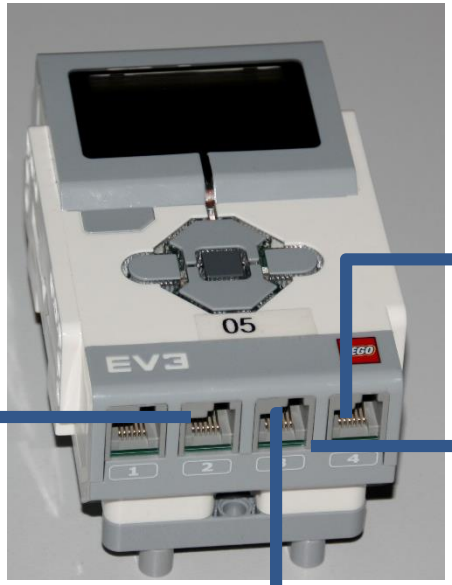
- Verfügt über integrierten **Rotationssensor**
  - misst Geschwindigkeit und Abstand
  - Leitet Ergebnisse an NXT Stein weiter
- Motor kann auf einen Grad genau gesteuert werden
- Kombinationen mehrerer Motoren möglich
  - arbeiten ggf. mit gleicher Geschwindigkeit

# DAS LEGO® MINDSTORMS® System

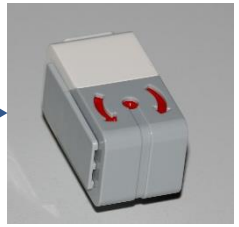
## Standardsensoren



Berührungssensor
Tastsenor




Ultraschallsensor



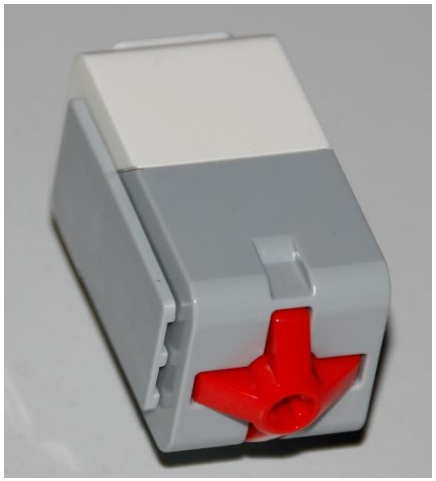
Gyrosensor

Farbsensor

Sensoren werden an die **Anschlüsse 1, 2, 3 und 4** angeschlossen.

DAS LEGO® MINDSTORMS® System

# Berührungssensor / Tastsensor

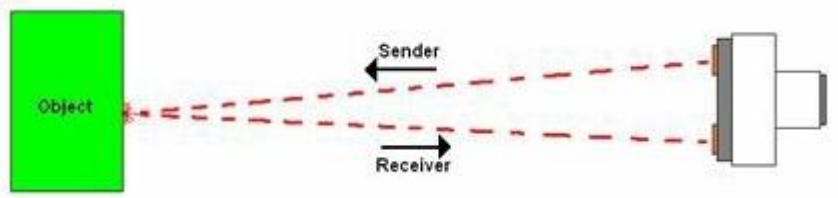


- Abfrage, ob Sensor gedrückt
- Werte des Sensors
  - 0: Sensor nicht gedrückt
  - 1: Sensor gedrückt

# DAS LEGO® MINDSTORMS® System

## Ultraschallsensor

- Sensor sendet Ultraschall aus
- Schall wird von Hindernis reflektiert
- Reflektierter Schall wird vom Empfänger registriert
- Aus Laufzeit des Schalls kann auf die Entfernung geschlussfolgert werden
- Messbereich: 3 bis 250 cm
- Messgenauigkeit: +/- 1 cm
- Messwerte werden in Meter ausgegeben



DAS LEGO® MINDSTORMS® System

## Colorsensor

- Verfügt über mehrere Moden, z. B.
  - Bestimmung des Farbwertes (ColorID)
  - Bestimmung der reflektierten Helligkeit
- Zur Ausleuchtung kann eine LED eingeschaltet werden

DAS LEGO® MINDSTORMS® System

## Colorsensor – ColorID Mode

- Bestimmung der Farbe
- Jede Farbe hat einen Wert
- Werte für EV3 Colorsensor

| Wert | Farbe |
|---|---|
| -1 | keine |
| 0 | Rot |
| 1 | Grün |
| 2 | Blau |
| 3 | Gelb |
| 4 | Magenta |
| 5 | Orange |
| 6 | Weiß |
| 7 | Schwarz |
| 8 | Pink |
| 9 | Grau |
| 10 | Hellgrau |
| 11 | Dunkelgrau |
| 12 | Zyan |
| 13 | Braun |

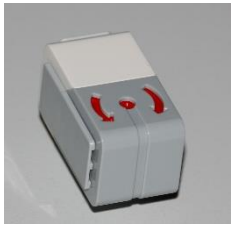
DAS LEGO® MINDSTORMS® System

# Colorsensor – ambient Light Mode

- Messung der Helligkeit mittels Fotodiode
- Helle Fläche reflektiert mehr Licht als dunkle
- Messbereich:
  - 0: dunkel
  - 100: hell
- Zur Ausleuchtung kann eine LED eingeschaltet werden

# DAS LEGO® MINDSTORMS® System

## Gyrosensor



- Messung der Drehbewegung und der Richtungsänderung
- Messbereich bis 440 °/s
- Messgenauigkeit; 1kHz
- Erfassungsrate: 1kHz

DAS SPIELFELD: Legostadt

# Allgemeiner Aufbau

# DAS SPIELFELD: Legostadt

## Aufgaben

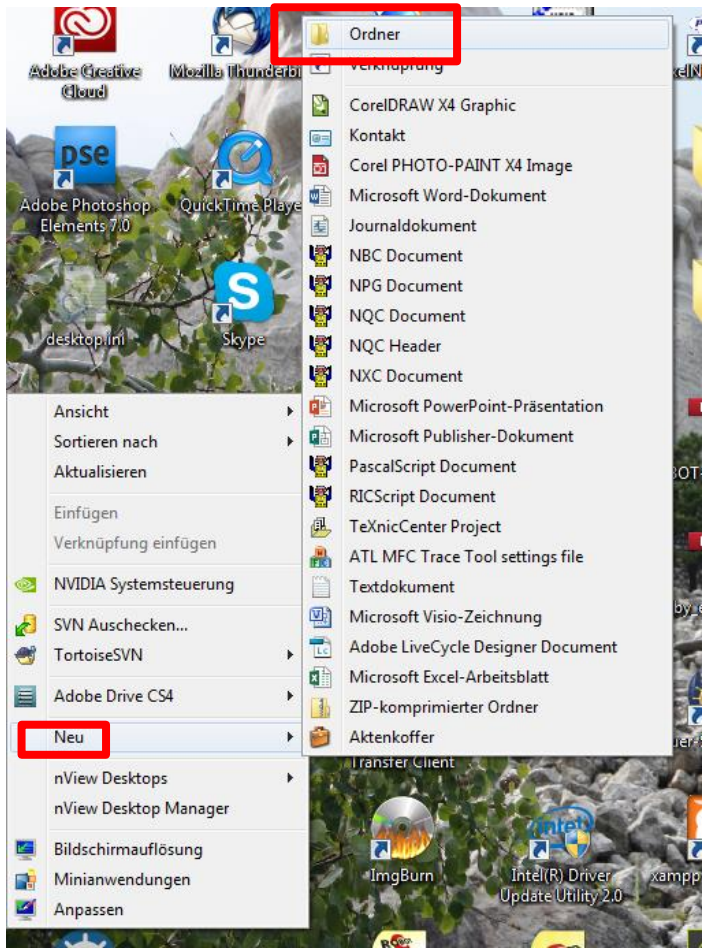
# Start der Entwicklungsumgebung

## Starten von Eclipse

Soll nicht im Standardverzeichnis gearbeitet werden:
Erstellen eines neuen Directories für Lego-Java-Programme

Im Ordner:
Lokaler Datenträger/Benutzer/IhrKonto

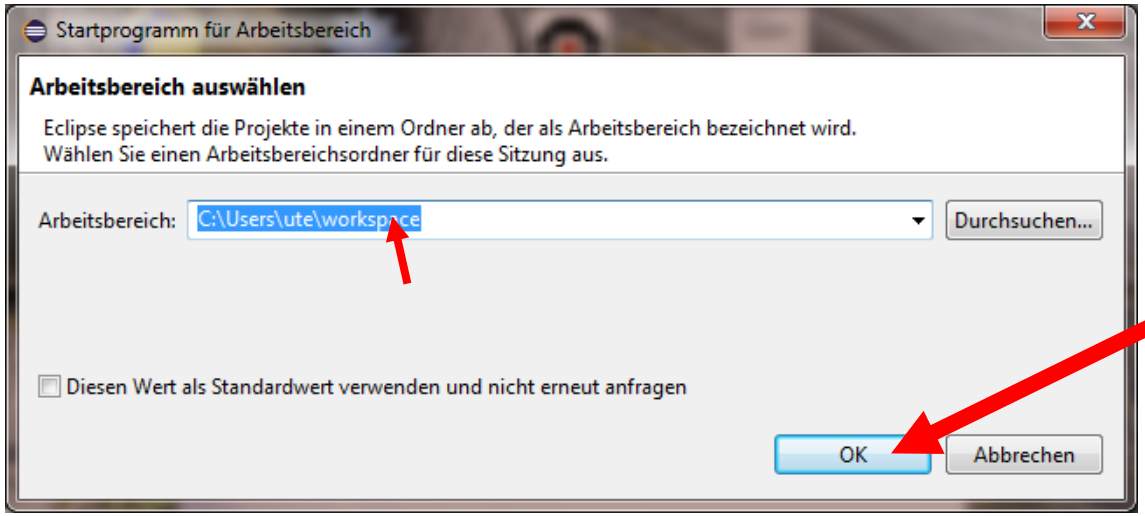
Ordnername: workspace_Lego



# Start der Entwicklungsumgebung

## Starten von Eclipse

1. Starten der VM aus bwLehrpool
2. Starten von Eclipse
3. Auswahl des Arbeitsbereiches
   Standardeinstellungen übernehmen oder
   ggf. Directory wechseln

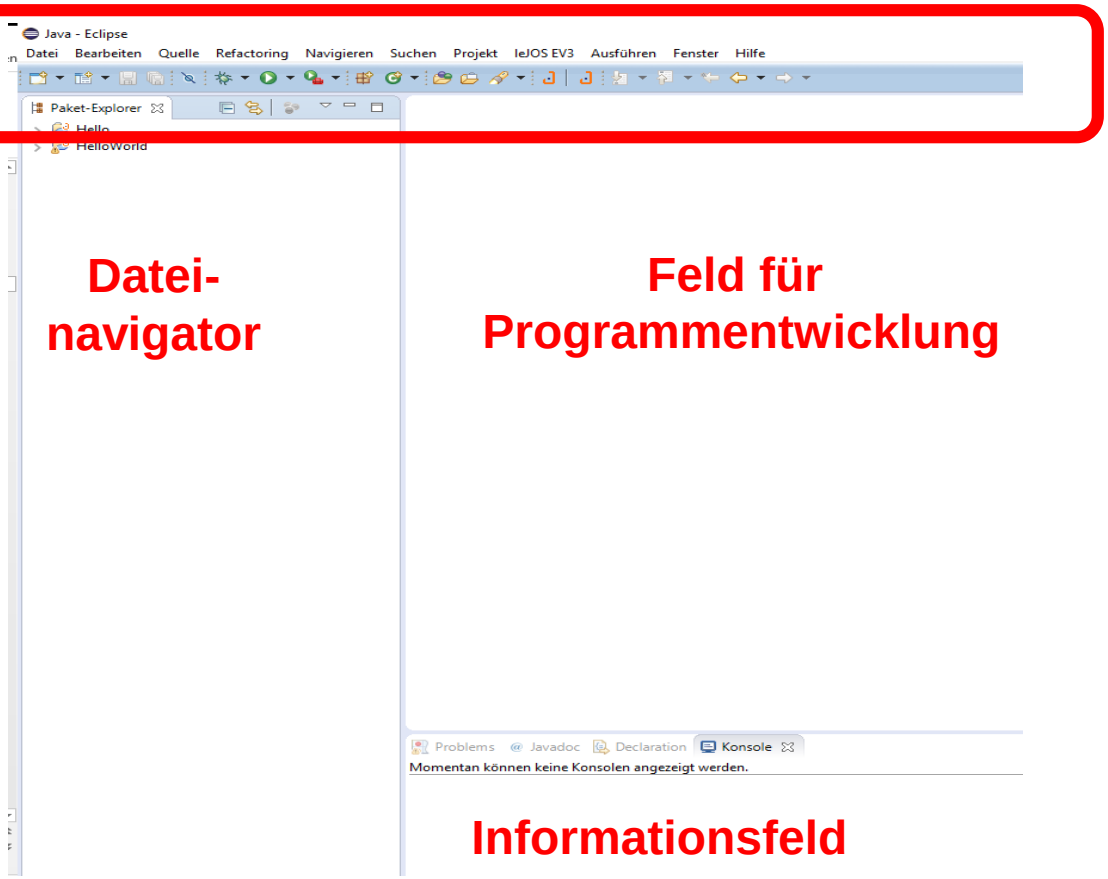



auf OK-Button drücken

# Start der Entwicklungsumgebung

## Starten von Eclipse

Arbeitsfläche von Eclipse




**Navigationsleiste**

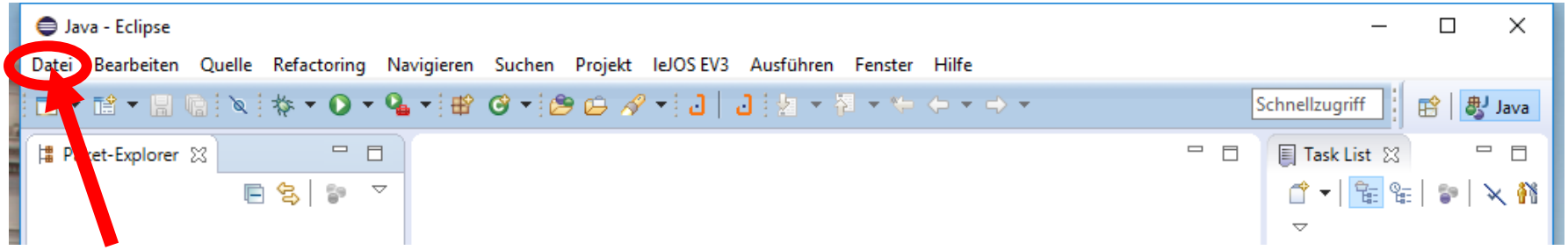
**Datei-navigator**

**Feld für Programmentwicklung**

**Informationsfeld**

Start der Entwicklungsumgebung

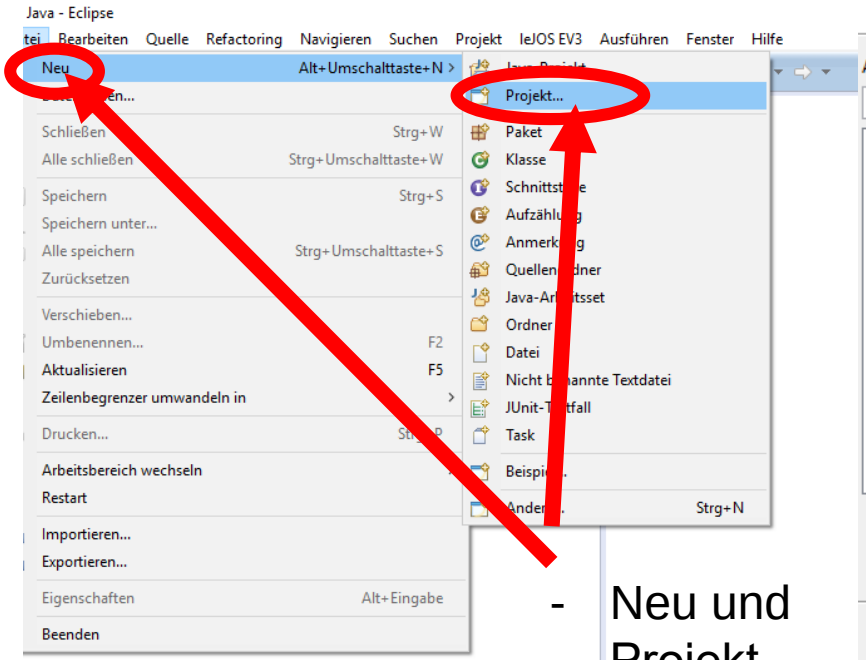
# Erstellen von Projekten und Klassen



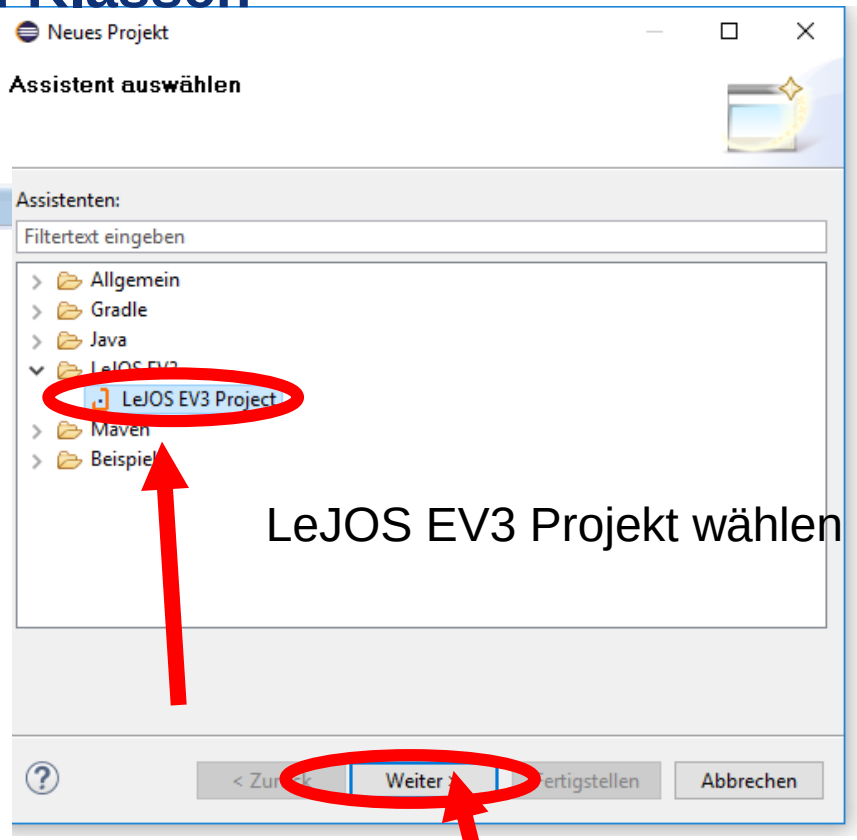

Datei wählen

# Start der Entwicklungsumgebung

## Erstellen von Projekten und Klassen




- Neu und
- Projekt… auswählen

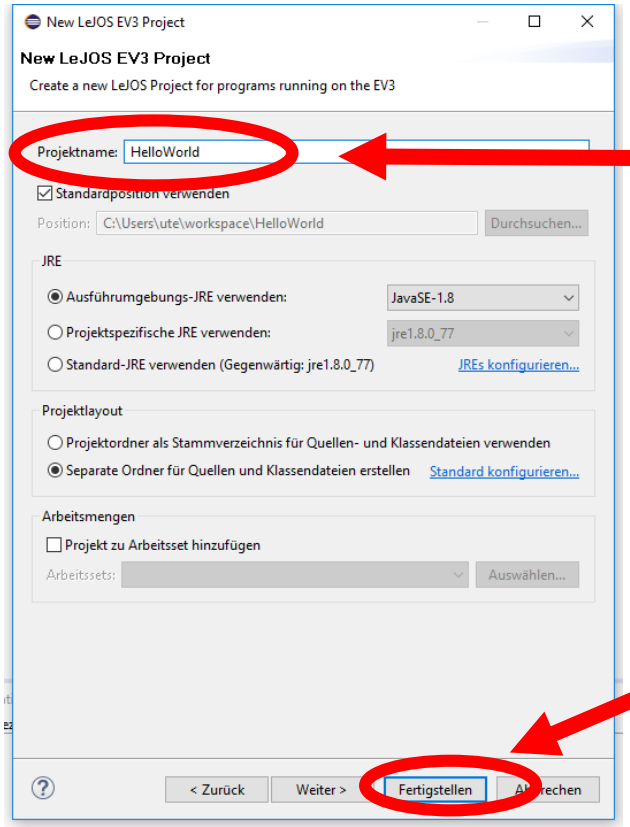



LeJOS EV3 Projekt wählen

auf Weiter> drücken

Start der Entwicklungsumgebung

# Erstellen von Projekten und Klassen

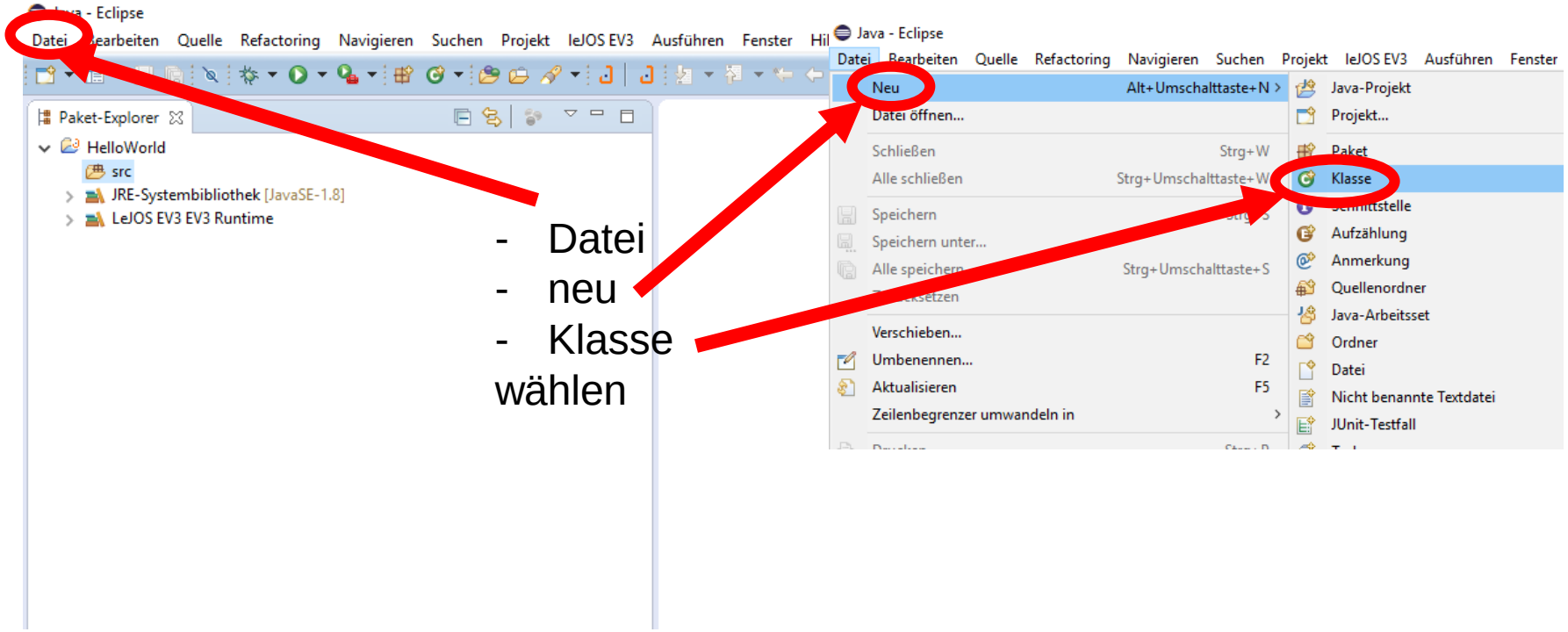



Projektname eingeben

Fertigstellen drücken

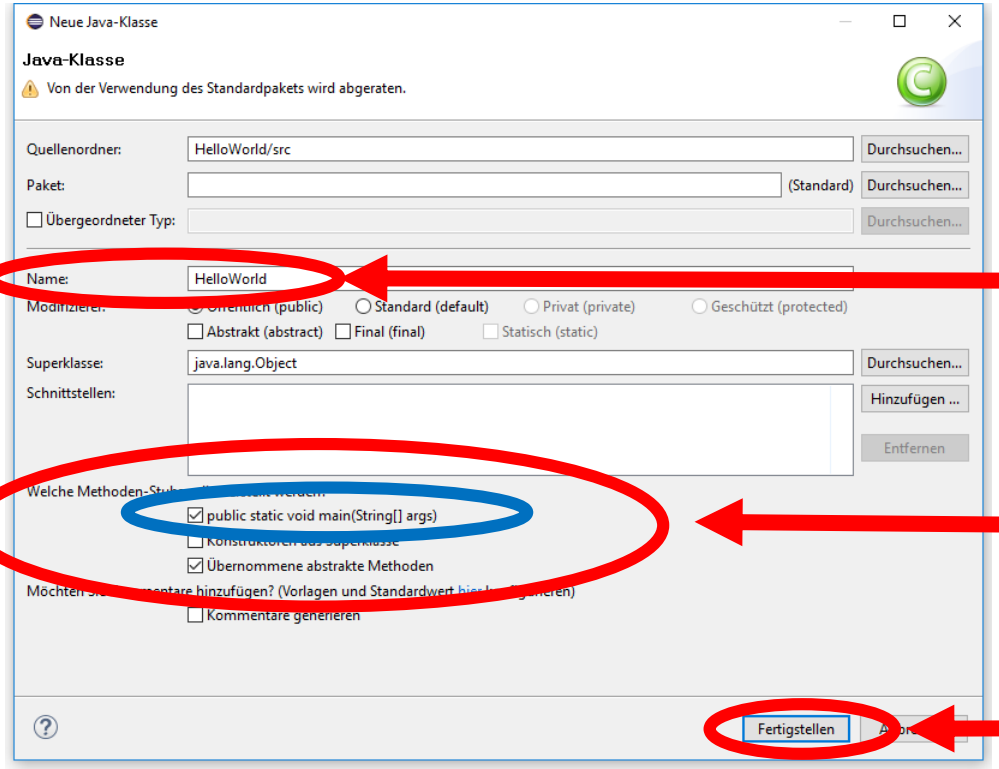
## Start der Entwicklungsumgebung

# Erstellen von Projekten und Klassen



## Start der Entwicklungsumgebung

# Erstellen von Projekten und Klassen




1. Klassennamen eingeben

2. Hier zusätzlich **public static void main** ankreuzen, wenn Klasse main Methode enthalten soll.

3. Fertigstellen drücken

JAVA Code

## Projekten und Klassen

- Jedes JAVA Programm besteht aus Klassen.

```
public class Berechnung {
    // hier wird der Programmcode eingefügt
}
```

- Eine der Klassen muss eine **main Methode** besitzen.
  Nur eine Klasse im Projekt darf eine **main Methode** besitzen.

```
public class Berechnung {
    public static void main(String[] args) {
        // hier wird der Programmcode eingefügt
    }
}
```

JAVA Code

# Methoden

- Klassen können Methoden enthalten.
- Beispielcode:

```
public class Beispiel_Methode {
    public static void main(String[] args) {
        // Aufruf der Methode hello
        hello();
    }

    // Methode hello
    public static void hello(){
        System.out.println("hello");
    }
}
```

# Hinweise zur Bearbeitung der Praktikumsaufgaben

- Jede Aufgabe des Spielfeldes ist eine eigenständige Aufgabe.
  D. h. jede Aufgabe kann einzeln gelöst werden und muss nicht mit anderen Aufgaben kombiniert werden.
- Lösungsvorschlag:
  a) Erstellen Sie ein Projekt mit einer Klasse, die eine Main Methode enthält und löschen den nicht mehr benötigten Inhalt.
  oder
  b) Erstellen Sie ein Projekt mit einer Main-Klasse und arbeiten mit Methoden (siehe Folie 22).
  Die nicht mehr verwendeten Methodenaufrufe werden in der Main-Methode als Kommentar gesetzt; ihr Quelltext kann im Programmcode erhalten bleiben.



# Erste Schritte: Bildschirmanzeigen

1. Nutzung des Standard JAVA Befehls

```
System.out.println("Hello World");
```

2. Nutzung des lejos Befehls
   a) für Strings `LCD.drawString(String, Spalte, Zeile);`

```
LCD.drawString("Hello Friend", 0, 1);
```

   b) Für Zahlen `LCD.drawInt(zahl, Spalte, Zeile);`

```
LCD.drawInt(7, 0, 2);
```

Für die Nutzung der lejos LCD Befehl ist folgende import-Funktion notwendig:

```
import lejos.hardware.lcd.LCD;
```



# Erste Schritte: Bildschirmanzeigen

3. Löschen des Displays

```
LCD.clearDisplay();
```



## Erste Schritte: Pausenbefehle

1. Warten darauf, dass ein Knopf des EV3 Steins gedrückt wird

```
Button.waitForAnyPress();
```

Für die Nutzung dieses lejos Befehls wird die import-Funktion benötigt:

```
import lejos.hardware.Button;
```

2. Nutzung eines leJos Pausen – Befehls: msDelay

```
Delay.msDelay(1000);
```

Für die Nutzung dieses lejos Befehls wird die import-Funktion benötigt:

```
import lejos.utility.Delay;
```



## Erste Schritte: Beispielprogramm

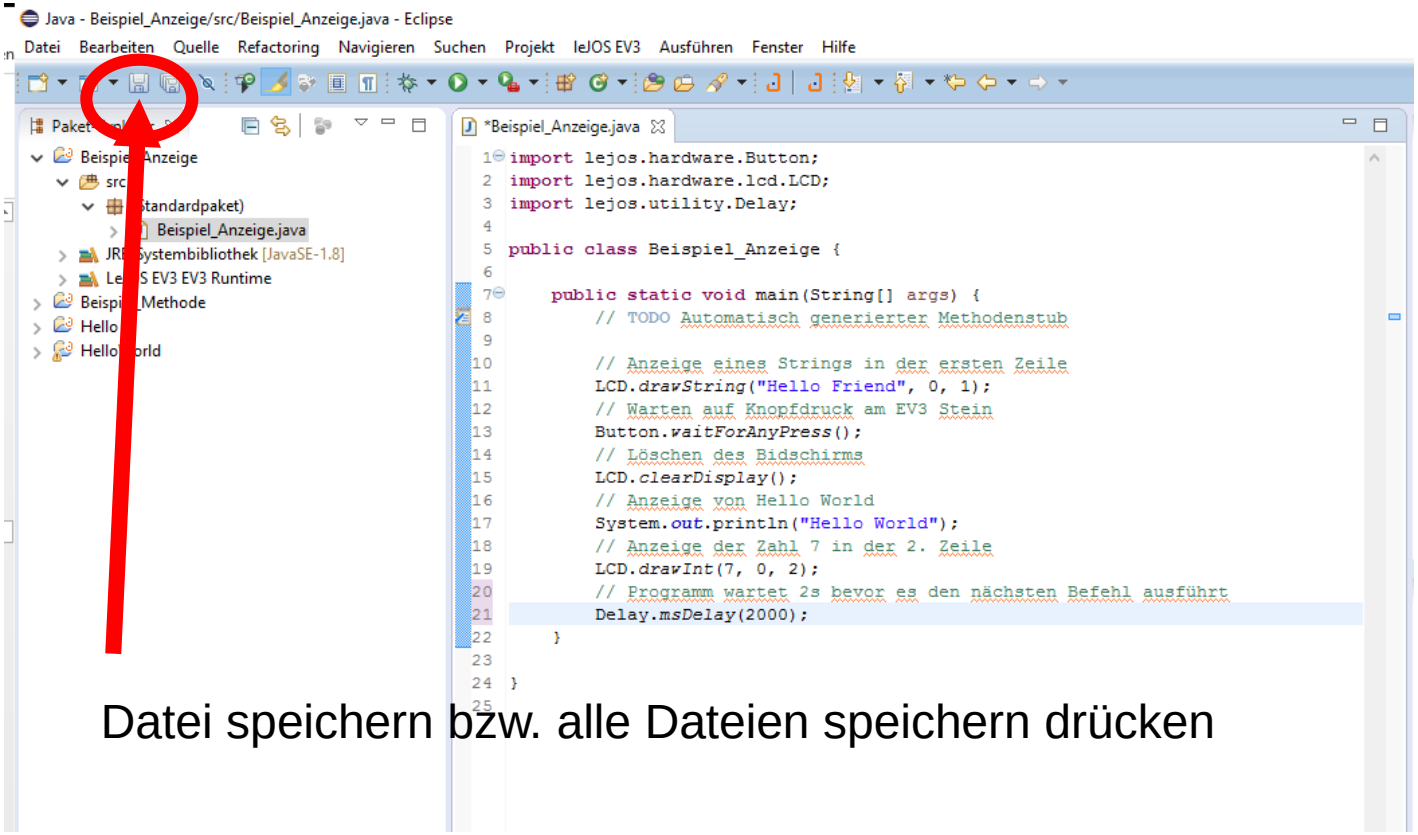
```
import lejos.hardware.Button;
import lejos.hardware.lcd.LCD;
import lejos.utility.Delay;

public class Beispiel_Anzeige {

   public static void main(String[] args) {

      // Inhalt nächste Folie
   }

}
```



## Erste Schritte: Beispielprogramm

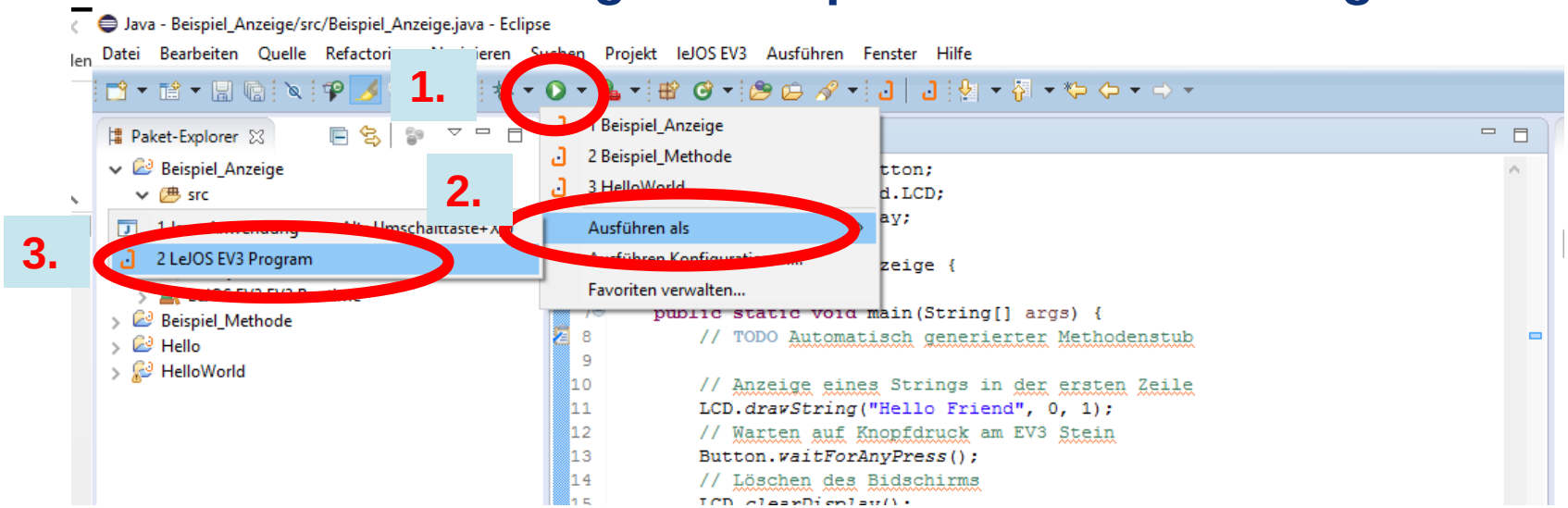
```
public static void main(String[] args) {
    // Anzeige eines Strings in der ersten Zeile
    LCD.drawString("Hello Friend", 0, 1);
    // Warten auf Knopfdruck am EV3 Stein
    Button.waitForAnyPress();
    // Löschen des Bidschirms
    LCD.clearDisplay();
    // Anzeige von Hello World
    System.out.println("Hello World");
    // Anzeige der Zahl 7 in der 2. Zeile
    LCD.drawInt(7, 0, 2);
    // Programm wartet 2s bevor es den nächsten Befehl ausführt
    Delay.msDelay(2000);
}
```

# Erste Schritte: Programm Speicher und Übertragen




Datei speichern bzw. alle Dateien speichern drücken

# Erste Schritte: Programm Speicher und Übertragen




- Ausführen als LeJOS EV3 Programm wählen, zuvor EV3 Stein einschalten

Achtung: Das Programm auf dem EV3 startet selbstständig!!!

# DAS SPIELFELD: Legostadt

## Aufgabe 1: Fahrt zum Flughafen

Start: P1

Ende: Flughafenhalle

Der Roboter soll aus P1 zum Parkfläche am Flughafen fahren.

Ziel:

Lernen der Steuerung des Roboters.

- Geradeausfahren
- Kurvenfahren

# DAS SPIELFELD: Legostadt

## Steuerung zweier Motoren mittels Zeitangaben

Vorwärtsfahren:
Motor.***A.forward();***
Motor.***B.forward();***

Rückwärtsfahren:
Motor.***A.backward();***
Motor.***B.backward();***

Kurve
Motor.***A.forward();***
Motor.***B.backward();***

oder
Motor.***A.backward();***
Motor.***B.forward();***

Anhalten mit Bremsen:
Motor.***A.stop();***
Motor.***B.stop();***

Für die Nutzung der Motor-Befehle wird die import-Funktion benötigt:

```
import lejos.hardware.motor.Motor;
```

Überprüfen Sie, dass die Motoren in den Ports A und B angeschlossen sind! Wenn nicht, dann entweder entsprechend Umstecken oder Portangabe im Programm ändern!

# DAS SPIELFELD: Legostadt

## Steuerung zweier Motoren mittels Zeitangaben

Setzen einer definierten Geschwindigkeit:

```
Motor.A.setSpeed(400);
Motor.B.setSpeed(400);
```

Hinweise für das Spielfeld:

- Der Roboter legt bei einer Geschwindigkeit von 400 in 1 s eine Strecke von 18,5 cm.
- Die Motoren nicht zwischen den einzelnen Teilbewegungen stoppen
- Anfangsgeschwindigkeit auf 400 festlegen

# DAS SPIELFELD: Legostadt

## Steuerung zweier Motoren mittels Zeitangaben

Beispielprogramm:

```
import lejos.hardware.motor.Motor;
import lejos.utility.Delay;

public class MotorBeispiel {

    public static void main(String[] args) {

        //Inhalt nächste Folie

    }

}
```

Der Roboter fährt
- Mit einer Geschwindigkeit von 400
- Geradeaus
- Macht eine Kurve
- Fährt rückwärts

# DAS SPIELFELD: Legostadt

## Steuerung zweier Motoren mittels Zeitangaben

Beispielprogramm:

```
public static void main(String[] args) {

    // Bei Bedarf: Setzen einer Motorgeschwindigkeit
    Motor.A.setSpeed(400);
    Motor.B.setSpeed(400);

    //Vorwärts für 1s
    Motor.A.forward();
    Motor.B.forward();
    Delay.msDelay(1000);

    // Fortsetzung nächste Folie
```

Der Roboter fährt
- Mit einer Geschwindigkeit von 400
- Geradeaus
- Macht eine Kurve
- Fährt rückwärts

# DAS SPIELFELD: Legostadt

## Steuerung zweier Motoren mittels Zeitangaben

Beispielprogramm:

```
        // Kurve nach Links bzw. Rechts
        Motor.A.forward();
        Motor.B.backward();
        Delay.msDelay(500);
        // Rückwärts für 1 s
        Motor.A.backward();
        Motor.B.backward();
        Delay.msDelay(1000);
        // Anhalten der Motoren
        Motor.A.stop();
        Motor.B.stop();
    }
```

Der Roboter fährt
- Mit einer Geschwindigkeit von 400
- Geradeaus
- Macht eine Kurve
- Fährt rückwärts

# DAS SPIELFELD: Legostadt

## Aufgabe 1: Fahrt zum Flughafen

Start: P1

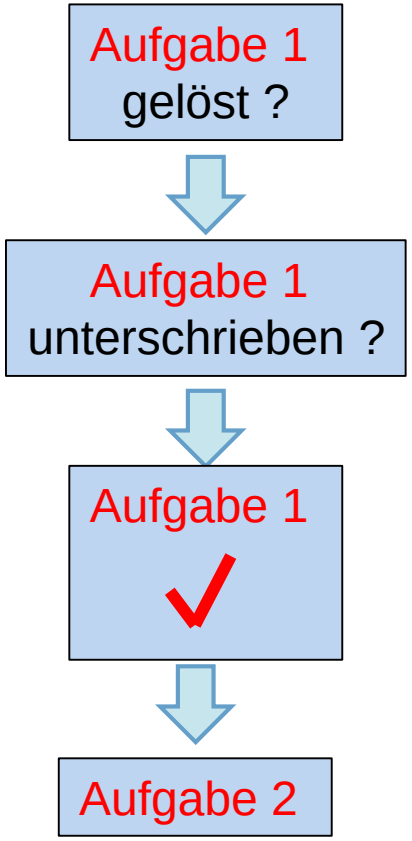
Ende: Flughafenhalle

Der Roboter soll aus P1 zum Parkfläche am Flughafen fahren.

Ziel:

Lernen der Steuerung des Roboters.

- Geradeausfahren
- Kurvenfahren



DAS SPIELFELD: Legostadt

# Aufgabe 2: Fahrt zum Krankenhaus auf verschiedenen Wegen

Start: P2

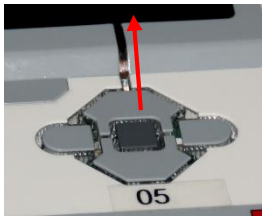
Ende: Parkfläche Krankenhaus

Der Roboter soll von P2 aus über 2 verschiedene Weg zum Krankenhaus fahren. Die Auswahl des Weges ist abhängig vom gedrückten Knopf des EV3 Steines.

Knopf Oben: über Cafe

alle anderen: über Hotel

Oben



# DAS SPIELFELD: Legostadt

## Die if-else Abfrage

```
if(<<Ausdruck>>){
  <<Anweisung>>
  …
  << Anweisung>>
}
else{
  << Anweisung>>
  …
  << Anweisung>>
}
```

Wenn der Ausdruck erfüllt ist, so werden die Anweisungen im if-Block erfüllt, ansonsten die Anweisung im else-Block.

Beispiel:

```
if(<<a==10>>){
  <<Anweisung>>
  …
  << Anweisung>>
}
else{
  << Anweisung>>
  …
  << Anweisung>>
}
```

DAS SPIELFELD: Legostadt

## Vergleichsoperatoren

| Operator | Beispiel | Wirkung |
|---|---|---|
| > | a > b | a größer als b |
| >= | a >= b | a größer oder gleich b |
| < | a < b | a kleiner als b |
| <= | a <= b | a kleiner oder gleich b |
| == | a == b | a ist gleich b |
| != | a != b | a ist ungleich b |

DAS SPIELFELD: Legostadt

## Abfrage von EV3 Buttons

Warten auf Knopfdruck:

```
Button.waitForAnyPress();
```

Abfrage, ob Knopf oben gedrückt ist:

```
Button.getButtons()==Button.ID_UP
```

# DAS SPIELFELD: Legostadt

## Beispielprogramm: if Abfrage

```
import lejos.hardware.Button;
import lejos.hardware.lcd.LCD;
import lejos.utility.Delay;

public class KnopfBeispiel {

   public static void main(String[] args) {

      // Inhalt nächste Folie

   }

}
```

Das Programm fragt ab, ob der linke oder ein anderer Knopf gedrückt wurde.

# DAS SPIELFELD: Legostadt

## Beispielprogramm:

```
public static void main(String[] args) {
    // Warten auf Knopfdruck
    LCD.drawString("Druecke Knopf", 0, 1);
    Button.waitForAnyPress();

    // Abfrage, ob Knopf oben gedrueckt ist
    if(Button.getButtons()==Button.ID_UP){
        LCD.drawString("Oben", 0, 2);
    }
    else{
        LCD.drawString("anderer", 0, 2);
    }
    Delay.msDelay(2000);
}
```

Das Programm fragt ab, ob der linke oder ein anderer Knopf gedrückt wurde.

DAS SPIELFELD: Legostadt

## Aufgabe 2: Fahrt zum Krankenhaus auf verschiedenen Wegen

Start: P2

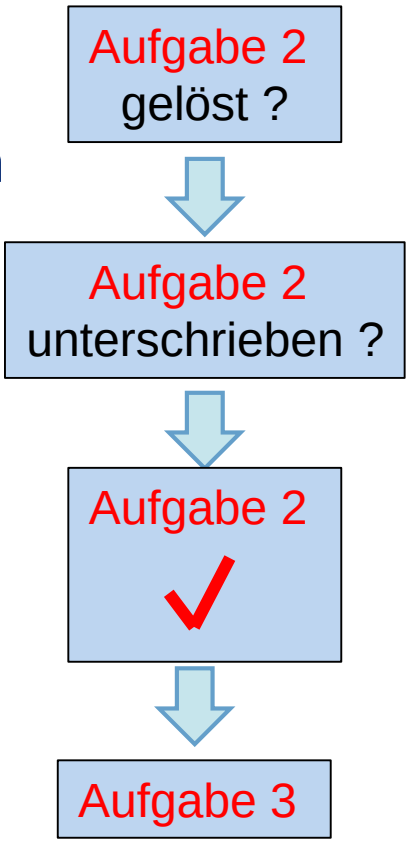
Ende: Parkfläche Krankenhaus

Der Roboter soll von P2 aus über 2 verschiedene Weg zum Krankenhaus fahren. Die Auswahl des Weges ist abhängig vom gedrückten Knopf des

EV3 Steines.

Knopf Oben: über Cafe

alle anderen: über Hotel



## DAS SPIELFELD: Legostadt

# Aufgabe 3: Beförderung von Fahrgästen zwischen Flughafen und Hotel

Start und Ende: Parkfläche Flughafen

Der Roboter soll als Shuttlebus Gäste zwischen Flughafen und Hotel hin und zurück befördern. An jedem Ort warten 3 Gäste. Es soll jeweils ein Gast transportiert werden.

Der Roboter startet per Knopfdruck, wenn der Gast eingestiegen ist. Der Roboter fährt die Strecke vom Flughafen zum Hotel vorwärts. Lässt den Gast ein- und aussteigen und fährt nach Knopfdruck die gleiche Strecke rückwärts zurück.

Auf den Parkflächen darf der Roboter neu ausgerichtet werden!

# DAS SPIELFELD: Legostadt

## Die for-Schleife

Eine Anweisung bzw. eine Folge von Anweisungen soll mehrfach wiederholt werden.

```
for(<<Startwert>>;<<Endwert>>;<<Erhöhung>>){
    <<Anweisung>>
    …
    <<Anweisung>>
}
```

Beispiel:

```
for(<i=1>;<i<=7>;<i++>){
    <<Anweisung>>
    …
    <<Anweisung>>
}
```

# DAS SPIELFELD: Legostadt

## Beispielprogramm: for Schleife

Das Programm gibt das Wort Test 4mal aus.

```
import lejos.hardware.lcd.LCD;
import lejos.utility.Delay;

public class BeispielFor {
   public static void main(String[] args) {

      LCD.clearDisplay();
      // Das Wort Test wird 4mal ausgegeben
      for(int i=1;i<=4;i++)
      {
         System.out.println("Test");
      }
      Delay.msDelay(4000);
   }
}
```

DAS SPIELFELD: Legostadt

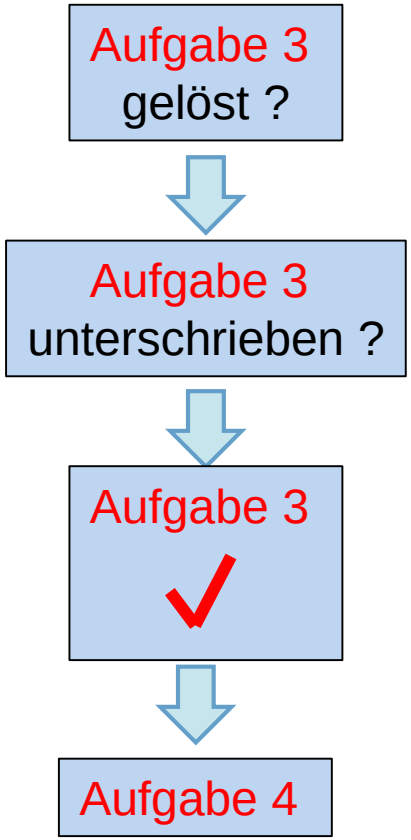
## Aufgabe 3: Beförderung von Fahrgästen zwischen Flughafen und Hotel

Start und Ende: Parkfläche Flughafen

Der Roboter soll als Shuttlebus Gäste zwischen Flughafen und Hotel hin und zurück befördern. An jedem Ort warten 3 Gäste. Es soll jeweils ein Gast transportiert werden.

Der Roboter startet per Knopfdruck, wenn der Gast eingestiegen ist. Der Roboter fährt die Strecke vom Flughafen zum Hotel vorwärts. Lässt den Gast ein- und aussteigen und fährt nach Knopfdruck die gleiche Strecke rückwärts zurück.

Auf den Parkflächen darf der Roboter neu ausgerichtet werden!



DAS SPIELFELD: Legostadt

## Aufgabe 4: Einparken mittels Tastsensor

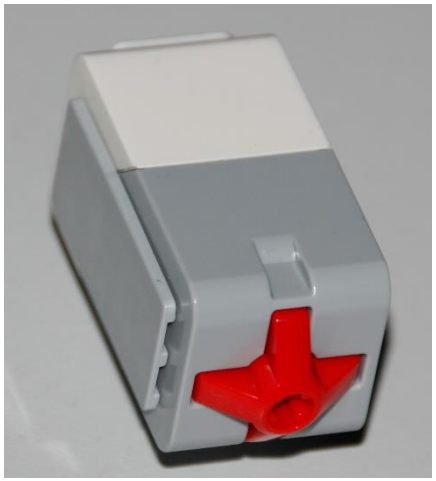
Start: Parkfläche vor Hotel

Ende: P3

Der Roboter soll rückwärts einparken. Er soll anhalten, wenn der Tastsensor die Bande berührt.

DAS SPIELFELD: Legostadt

## Berührungssensor / Tastsensor



- Abfrage, ob Sensor gedrückt
- Werte des Sensors
  - 0: Sensor nicht gedrückt
  - 1: Sensor gedrückt

DAS SPIELFELD: Legostadt

## Zur Arbeit mit Sensoren

**Folgende import – Funktionen werden benötigt**

```
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.*;
import lejos.robotics.*;
```

**Arbeitsanweisung:**
Fügen Sie **jetzt** diese drei Zeilen in ihr Programm an den Anfang, wo alle anderen import Funktionen stehen ein.

**Hinweis:**
Die Initialisierung der Sensoren und die Abfrage der Messwerte erfolgt bei allen Sensoren nach dem gleichen Prinzip.
Wichtig ist, dass stets der Port (S1, S2, S3 bzw. S4) in der Initialisierung verwendet wird, an dem der Sensor tatsächlich angeschlossen ist.

# DAS SPIELFELD: Legostadt

## Zur Arbeit mit dem Tastsensor

**Initialisierung:**

```
SensorModes sensor1 = new EV3TouchSensor(SensorPort.S1);
SampleProvider touch1 = sensor1.getMode("Touch");
```

Jeweiligen Anschlußport angeben (S1, S2, S3 oder S4)

**Abfrage der Messwerte:**

```
// Initialisierung der Messwerte
float pressed =0;
float sample[] = new float[touch1.sampleSize()];

// Abfrage der Sensorwerte
touch1.fetchSample(sample, 0);
pressed = sample[0];
```

Die Variable **pressed** enthält die Information, über den Zustand des Tastsensors. Diese gilt es im Programm abzufragen.

DAS SPIELFELD: Legostadt

## Die bedingte while-Schleife

Eine Anweisung bzw. eine Folge von Anweisungen soll bis eine bestimmten Bedingung nicht mehr erfüllt ist, wiederholt werden.

```
while(<<Bedingung>>){
<<Anweisung>>
    …
    <<Anweisung>>
}
```

Beispiel:

```
while(<<i<=5>>){
<<Anweisung>>
    …
    <<Anweisung>>
}
```

# DAS SPIELFELD: Legostadt

## Beispielprogramm: Tastsensor

Im Beispiel ist der Tastsensor am Port 1.

```
import lejos.hardware.lcd.LCD;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.*;
import lejos.robotics.*;
import lejos.utility.Delay;

public class TasterBeispiel {

   public static void main(String[] args) {
      // Inhalt nächste Folie

   }

}
```

Das Programm erhöht eine Variable um 1, bis der Tastsensor gedrückt wird und zeigt anschließend das Ergebnis an.

DAS SPIELFELD: Legostadt

## Beispielprogramm: Tastsensor

Im Beispiel ist der Tastsensor am Port 1.

```
public static void main(String[] args) {
    // Initialisierung Tastsensor
    SensorModes sensor1 = new  EV3TouchSensor(SensorPort.S1);
    SampleProvider touch1 = sensor1.getMode("Touch");

    // Initialisierung der Messwerte
    float pressed = 0;
    float sample[] = new float[touch1.sampleSize()];

    // Initialsierung einer Integervariablen
    int zahl=0;
```

Das Programm erhöht eine Variable um 1, bis der Tastsensor gedrückt wird und zeigt anschließend das Ergebnis an.

# DAS SPIELFELD: Legostadt

## Beispielprogramm: Tastsenor

Im Beispiel ist der Tastsensor am Port 1.

```
    LCD.drawString("Tastsensor druecken", 0, 1);
    while(pressed==0){
        // Abfrage Tastsensor
        touch1.fetchSample(sample, 0);
        pressed =sample[0];
        zahl=zahl+1;
    }
    LCD.drawString("zahl =", 0, 4);
    LCD.drawInt(zahl, 0, 3);
    Delay.msDelay(2000);

}
```

Das Programm erhöht eine Variable um 1, bis der Tastsensor gedrückt wird und zeigt anschließend das Ergebnis an.

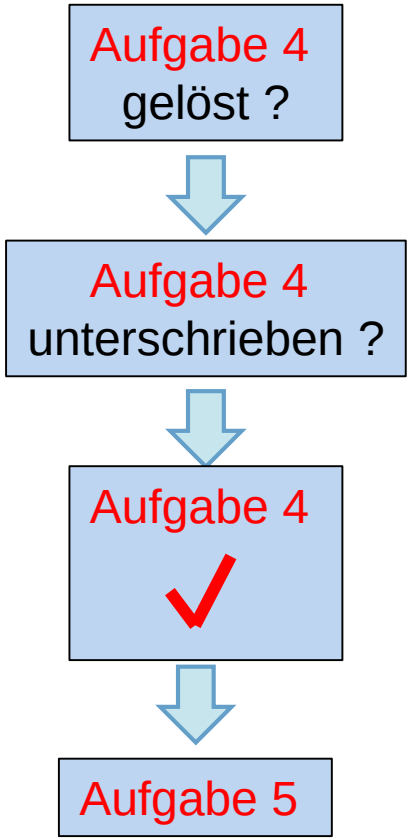
DAS SPIELFELD: Legostadt

## Aufgabe 4: Einparken mittels Tastsensor

Start: Parkfläche vor Hotel

Ende: P3

Der Roboter soll rückwärts einparken. Er soll anhalten, wenn der Tastsensor die Bande berührt.



DAS SPIELFELD: Legostadt

## Aufgabe 5: Einparken mittels Ultraschallsensor

Start: Parkfläche Schule

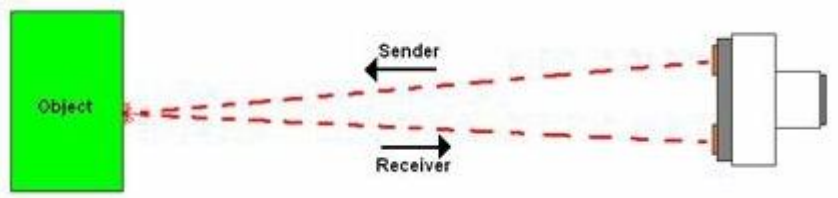
Ende: P1 – Garage

Der Roboter holt einen Schüler ab. Dabei parkt er selbstständig in die Garage ein. Er soll stehenbleiben, wenn der Abstand zur Wand kleiner als 5 cm ist.

DAS SPIELFELD: Legostadt

## Ultraschallsensor

- Sensor sendet Ultraschall aus
- Schall wird von Hindernis reflektiert
- Reflektierter Schall wird vom Empfänger registriert
- Aus Laufzeit des Schalls kann auf die Entfernung geschlussfolgert werden
- Messbereich: 3 bis 250 cm
- Messgenauigkeit: +/- 1 cm
- Messwerte werden in Meter ausgegeben



# DAS SPIELFELD: Legostadt

## Zur Verwendung des Ultraschallsensors

**Initialisierung:**

```
SensorModes sensor4 = new EV3UltrasonicSensor(SensorPort.S4);
SampleProvider us = sensor4.getMode("Distance");
```

Jeweiligen Anschlußport angeben (S1, S2, S3 oder S4)

**Abfrage der Messwerte:**

```
// Initialisierung der Messwerte
float distanz=10;
float sample[] = new float[us.sampleSize()];

// Abfrage der Messwerte
us.fetchSample(sample, 0);
distanz = sample[0];
```

# DAS SPIELFELD: Legostadt

## Beispielprogramm: Ultraschallsensor

```
import lejos.hardware.lcd.LCD;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.*;
import lejos.robotics.*;
import lejos.utility.Delay;

public class UltraschallBeispiel {
   public static void main(String[] args) {

      // Inhalt nächste Folie
   }
}
```

Das Programm zeigt die Entfernung in Metern an, solange der Abstand größer ist als 10 cm.

# DAS SPIELFELD: Legostadt

## Beispielprogramm: Ultraschallsensor

```
public static void main(String[] args) {

   // Initialisierung Ultraschallsensor
   SensorModes sensor4 = new EV3UltrasonicSensor(SensorPort.S4);
   SampleProvider schall1 = sensor4.getMode("Distance");

   // Initialisierung Messwerte
   float distanz=10;
   float sample[] = new float[schall1.sampleSize()];
```

Das Programm zeigt die Entfernung in Metern an, solange der Abstand größer ist als 10 cm.

# DAS SPIELFELD: Legostadt

## Beispielprogramm: Ultraschallsensor

```
    while(distanz>=0.1){
        // Abfrage der Messwerte
        schall1.fetchSample(sample, 0);
        distanz = sample[0];
        // Anzeige Messwerte
        LCD.drawString("Weg: "+distanz, 0, 1);
        Delay.msDelay(100);
    }
}
```

Das Programm zeigt die Entfernung in Metern an, solange der Abstand größer ist als 10 cm.

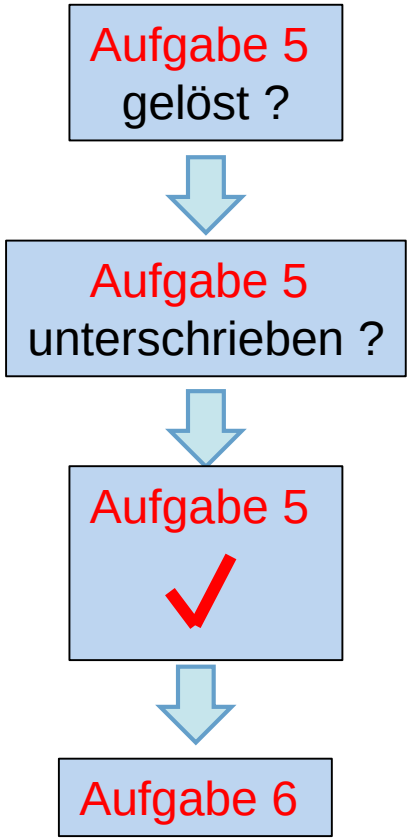
# DAS SPIELFELD: Legostadt

## Aufgabe 5: Einparken mittels Ultraschall-sensor

Start: Parkfläche Schule

Ende: P1 – Garage

Der Roboter holt einen Schüler ab. Dabei parkt er selbstständig in die Garage ein. Er soll stehenbleiben, wenn der Abstand zur Wand kleiner als 5 cm ist.



DAS SPIELFELD: Legostadt

## Aufgabe 6: Ausflugsziel

Start: P4

Ende: entsprechendes Farbfeld

Der Roboter soll in Abhängigkeit von ermittelten Farbe am entsprechenden Ausflugsziel anhalten. Das Farbfeld wird über eine Zufallszahl ermittelt (siehe Folie73). Die Zufallszahl soll angezeigt werden.

0 – Gelb (Farb-ID: 3)

1 – Blau (Farb-ID: 2)

2 – Schwarz (Farb-ID: 7)

3 – Rot (Farb-ID: 0)

DAS SPIELFELD: Legostadt

## Colorsensor – ColorID Mode

- Bestimmung der Farbe
- Jede Farbe hat einen Wert
- Werte für EV3 Colorsensor

| Wert | Farbe |
|---|---|
| -1 | keine |
| 0 | Rot |
| 1 | Grün |
| 2 | Blau |
| 3 | Gelb |
| 4 | Magenta |
| 5 | Orange |
| 6 | Weiß |
| 7 | Schwarz |
| 8 | Pink |
| 9 | Grau |
| 10 | Hellgrau |
| 11 | Dunkelgrau |
| 12 | Zyan |
| 13 | Braun |

# DAS SPIELFELD: Legostadt

## Zur Verwendung des Farbsensors (ColorID Mode)

**Initialisierung:**

```
SensorModes colorSensor = new EV3ColorSensor(SensorPort.S3);
SampleProvider col = colorSensor.getMode("ColorID");
```

Jeweiligen Anschlußport angeben (S1, S2, S3 oder S4)

**Abfrage der Messwerte:**

```
// Intialisierung der Messwerte
int sampleSize = colorSensor.sampleSize();
float[] sample = new float[sampleSize];
int farbe;

// Abfrage der Messwerte
col.fetchSample(sample, 0);
// Umrechung float in integer
farbe = (int)sample[0];
```

Die Variable **farbe** gibt den erkannten Farbwert aus. Diese ist abzufragen.

DAS SPIELFELD: Legostadt

## Beispielprogramm: Farbsensor

```
import lejos.hardware.Button;
import lejos.hardware.lcd.LCD;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.*;
import lejos.robotics.*;
import lejos.utility.Delay;

public class FarbsensorBeispiel {

   public static void main(String[] args) {

      // Inhalt nächste Folie
   }
}
```

Das Programm zeigt 4 Messwerte an.

DAS SPIELFELD: Legostadt

## Beispielprogramm: Farbsensor

```
public static void main(String[] args) {

   // Initialisierung Farbsensor
   SensorModes colorSensor1 = new EV3ColorSensor(SensorPort.S3);
   SampleProvider col1 = colorSensor1.getMode("ColorID");

   // Intialisierung der Messwerte
   int SampleSize = colorSensor1.sampleSize();
   float[] sample = new float[SampleSize];

   // Variable für den Farbwert
   int farbe;
   LCD.clearDisplay();
```

Das Programm zeigt 4 Messwerte an.

DAS SPIELFELD: Legostadt

## Beispielprogramm: Farbsensor

```
for(int i=1;i<=4;i++){
    LCD.drawString("Messung starten", 0, 1);
    LCD.drawString("Knopf druecken", 0, 2);
    Button.waitForAnyPress();
    // Messwert erfassen
    col1.fetchSample(sample, 0);
    // Umrechnung des Messwertes in eine Integervariable
    farbe = (int)sample[0];

    // Anzeige Messwert
    LCD.drawString("Farbwert:", 0, 3);
    LCD.drawInt(farbe, 0, 4);
    Delay.msDelay(2000);
    LCD.clearDisplay();
}
}
```

Das Programm zeigt 4 Messwerte an.

# DAS SPIELFELD: Legostadt

## Abfrage einer Zufallszahl

Benötigt wird die Import-Funktion:

```
import java.util.*;
```

Festlegung des Wertebereiches:

```
Random wuerfel = new Random();
```

Erzeugung einer Zufallszahl (integer) im Wertebereich 0…3:

```
int zahl zahl = wuerfel.nextInt(3);
```

DAS SPIELFELD: Legostadt

## Aufgabe 6: Ausflugsziel

Start: P4

Ende: entsprechendes Farbfeld

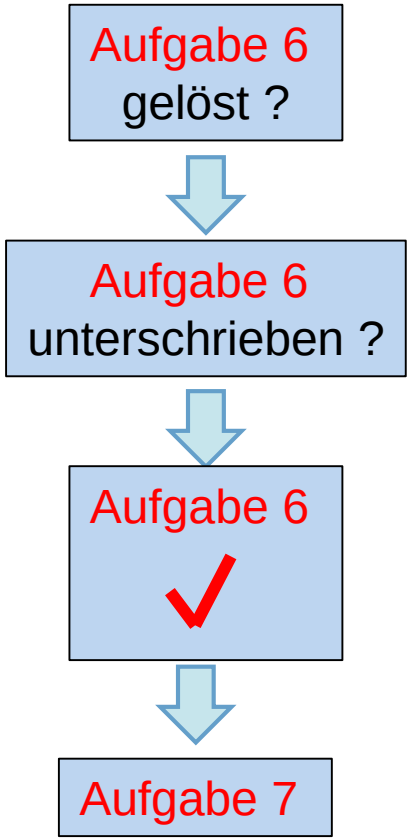
Der Roboter soll in Abhängigkeit von ermittelten Farbe am entsprechenden Ausflugsziel anhalten. Das Farbfeld wird über eine Zufallszahl ermittelt (siehe Folie 73). Die Zufallszahl soll angezeigt werden.

0 – Gelb (Farb-ID: 3)

1 – Blau (Farb-ID: 2)

2 – Schwarz (Farb-ID: 7)

3 – Rot (Farb-ID: 0)



## DAS SPIELFELD: Legostadt

# Aufgabe 7: Folge dem Weg zum Leuchtturm

Start: P3

Ende: Gelbes Feld beim Leuchtturm

Der Roboter soll der schwarzen Linie zum Leuchtturm folgen. Der Roboter soll anhalten, sobald das Endfeld (gelb) erreicht ist.