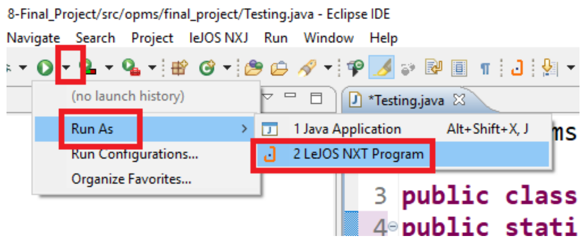
Summer School 2018 – Final Project

Task 1 (Setup)

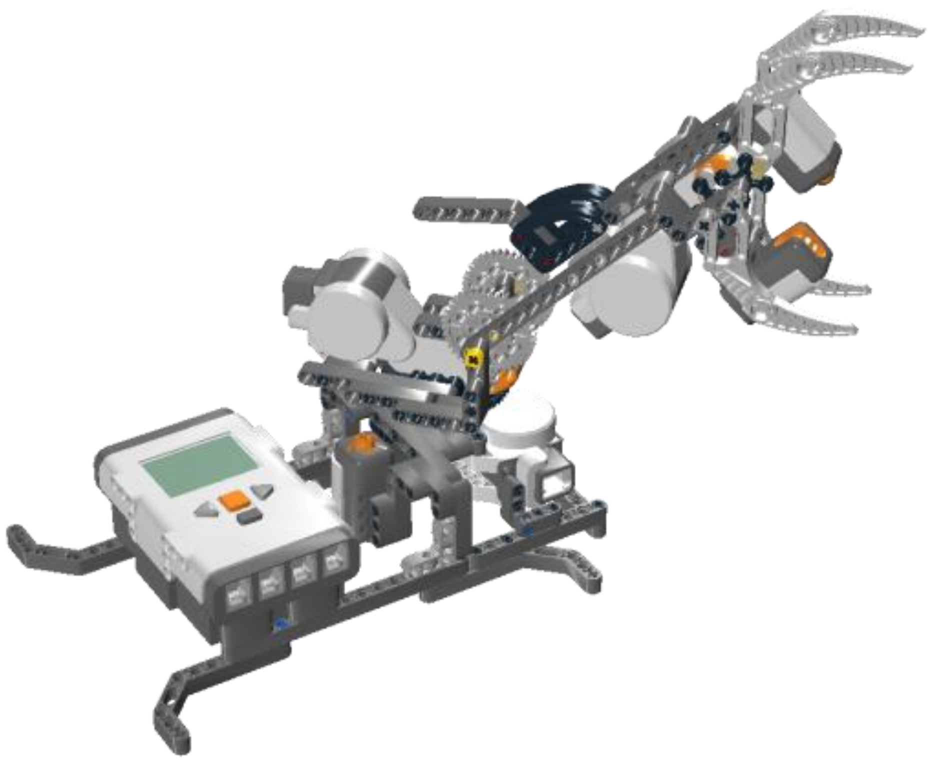
1. Go to the project template for the final project.
2. Have a look at the different classes and get familiar with the code in them.
3. Implement a new class `Testing` which has a main()-method and which will be used throughout this project to test your implementations.
4. In this method, add `System.out.println("I am a robot");`
   Make sure that the robot is on!
5. Send the code to the robot and it will execute the commands, see the screenshot below.



8-Final_Project/src/opms/final_project/Testing.java - Eclipse IDE
Navigate Search Project leJOS NXJ Run Window Help
(no launch history)
Run As
Run Configurations...
Organize Favorites...
*Testing.java
1 Java Application Alt+Shift+X, J
2 LeJOS NXT Program

```
public class
public stati
```

6. The print message is printed on the robot display instead of the eclipse console. To keep the messages on the display add `Button.waitForAnyPress();` at the end of your program. Execute your code again. Make sure that you see the output printed to the display. Press any button to exit the program.

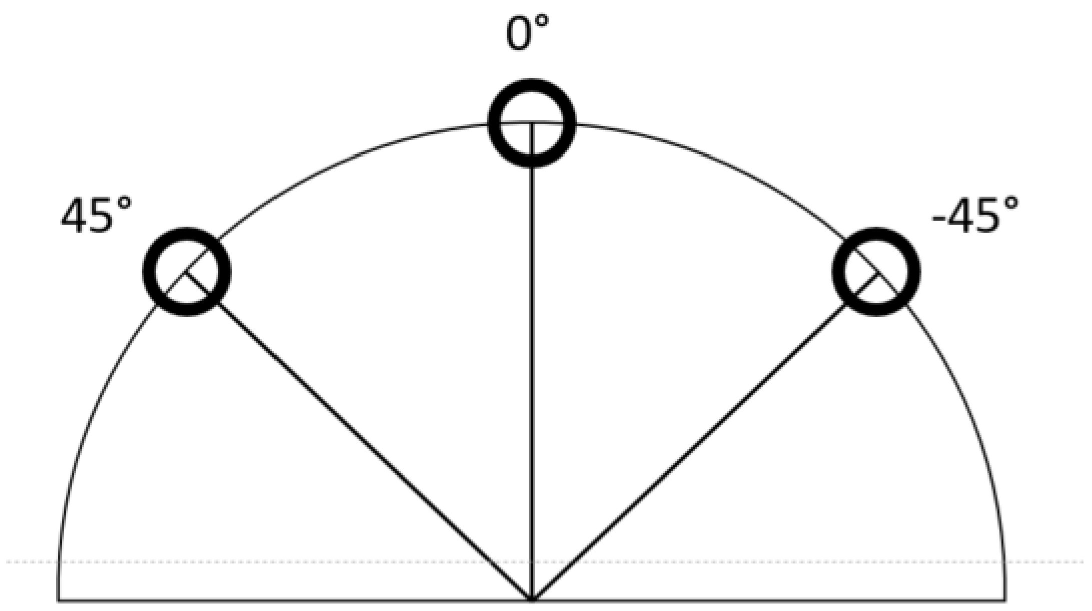


IMA - RWTH Aachen University
Institute of Information Management in Mechanical Engineering IMA

Task 2 (Actuators)

In this task, you will enable the robot to perform the movements necessary for solving the project task.

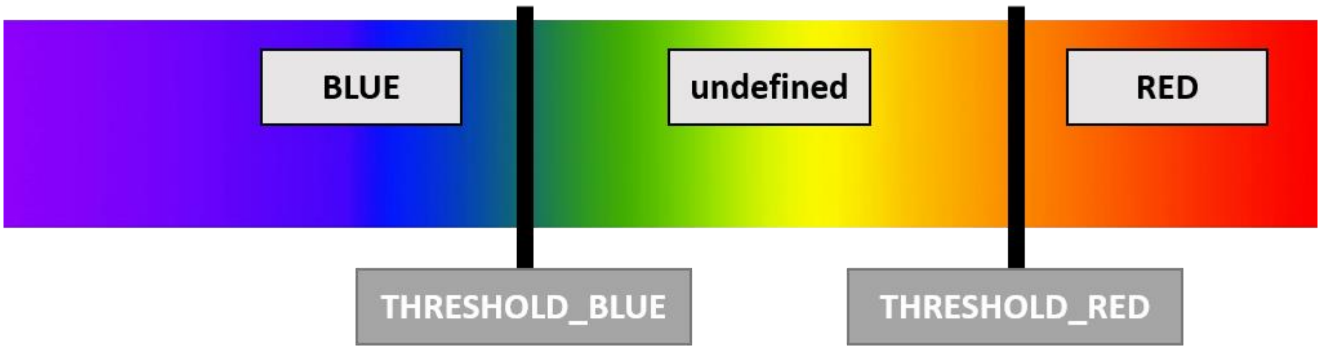
1. Open the class `Actuators`. Look at the different methods understand their purpose.
2. Write a new method *moveAboveRing* which takes the index of an ring as an argument. The function does not return anything. First make sure that the ring index provided is either 0, 1 or 2. Throw an IndexOutOfBoundsException if it is not.
3. Calculate the angle to which the robot should rotate to. Assume that the rings are at -45°, 0° and 45°.
4. Use the Motor C to rotate the robot according to the angle you determined. Test your code by calling the *moveAboveRing* function from the `Testing` class.
5. As you might have noticed, the robot does not turn as much as expected. This is because the gear translation has to be taken into account. The gears used have 55 and 8 teeth. Calculate the transmission and store this value in the constant TRANSMISSION_ROTATION. Then use TRANSMISSION_ROTATION to correct your calculation of the angle.



Task 3 (Sensors)

In this task, you will enable the robot to use its sensors and perceive its surroundings.

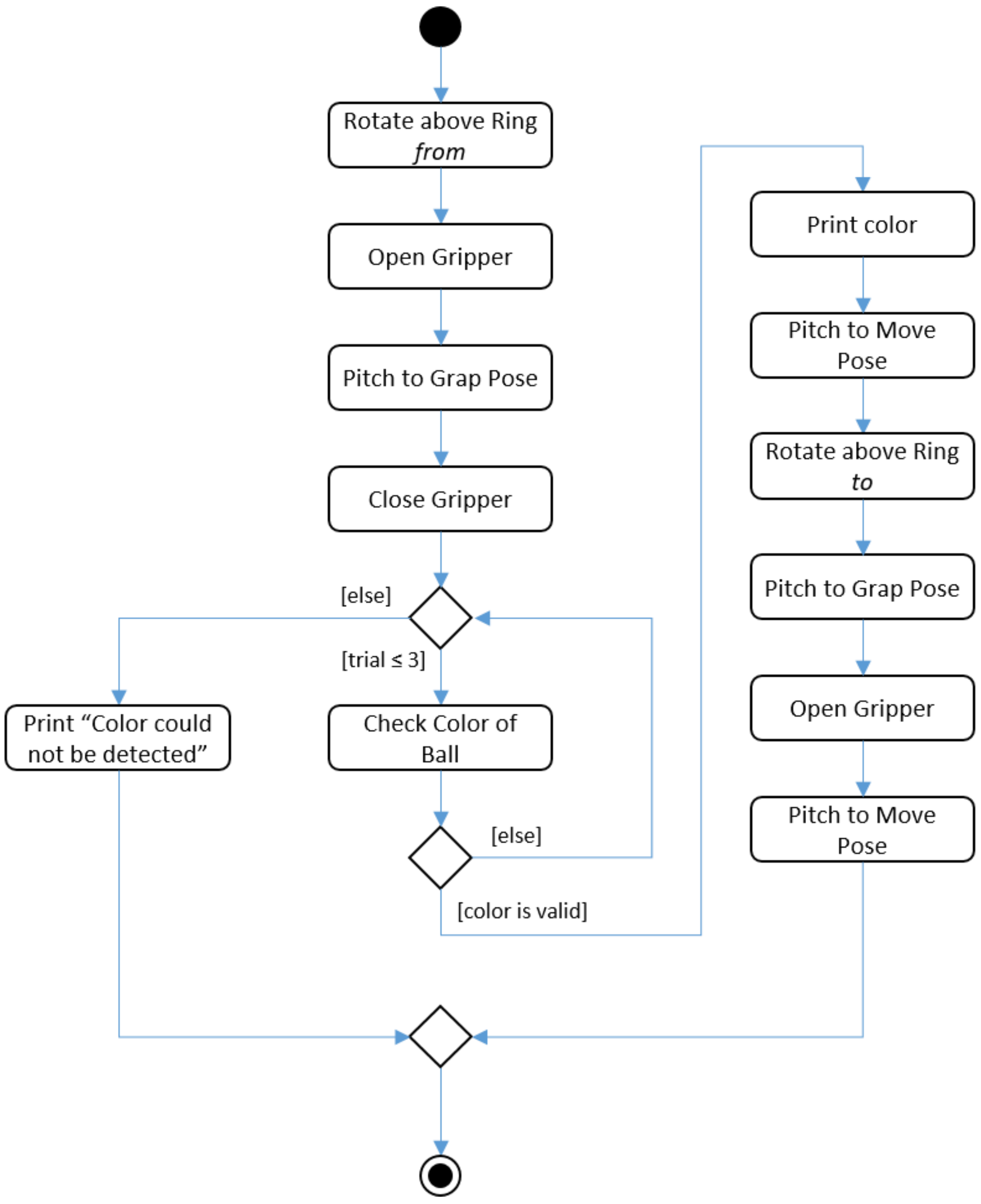
1. Open the class Sensor.
2. Instantiate a LightSensor *light*. Like the UltrasonicSensor, the constructor requires a port number. Use port S4.
3. Write a function `getColor` which reads the normalized value of the light sensor. Use the THRESHOLD_RED and THRESHOLD_BLUE to determine if the object in front of the sensor is red or blue. The function should return a String which is either "RED" or "BLUE".
4. Create a new exception `InvalidColorDetectedException`. Throw this exception in your function `getColor` if the color detected is neither red nor blue.
5. Test your function from your Testing class. Place the balls in front of the sensor and check if the right color is detected.



Task 4 (Basic Movements)

Now you will enable the robot to move a ball from one ring to another. The robot will use this skill to solve the final challenge of this project.

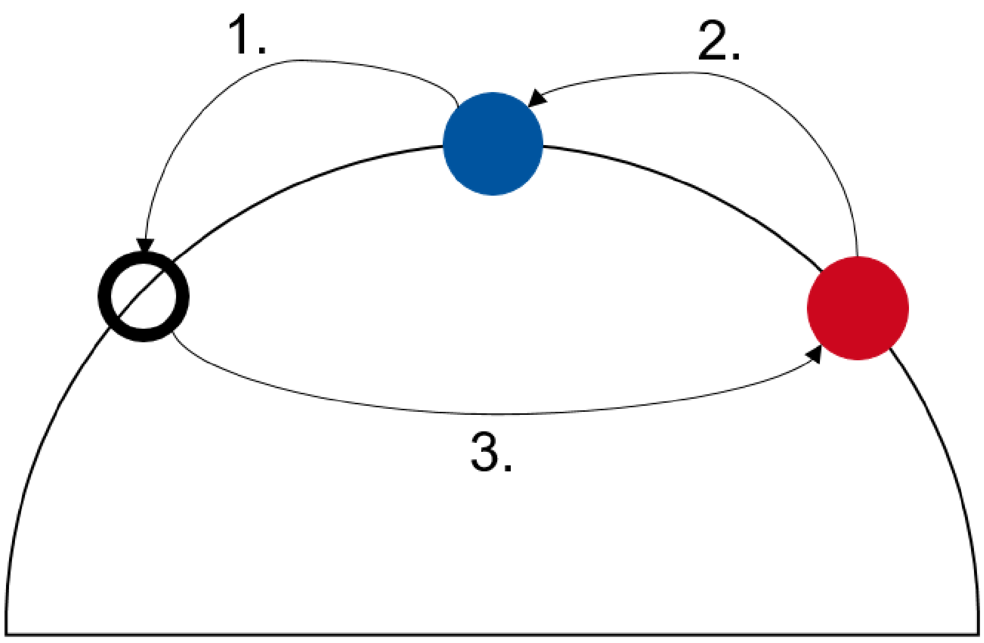
6. Open the class Robot and create a new method called `moveBall`. The method should take two Integer arguments called *from* and *to*. The input arguments denote **from** which ring and **to** which ring the robot should move a ball.
7. Implement the behavior described by the following activity diagram



Task 5 (Full Program)

Finally, you will implement the switching of two balls by the robot.

1. In the class `Robot` create a new class *run*.
2. Assume that the balls are on ring 0 and 1. Implement the switching of the balls. Use the method *moveBall* from Task 4.
3. Test your implementation using the `Testing` class.



Task 6 (Additional Functionality)

Now that the robot is able to switch the balls between ring 0 and 1, the next step is switching balls between arbitrary rings.

1. Go back to the class `Actuator`. Implement a new function called *moveToCheckRing*. Similar to the function *moveAboveRing* from Task 2, the new function takes a ring id as argument. Instead of moving directly over the ball, move the robot 20° right of the ring to place the ultrasonic sensor over the ring.
2. Extend your method *run* in the class `Robot`. Use *moveToCheckRing* method to move over all rings subsequently and determine, which rings have balls on them and which one does not.
3. Adjust the code moving the balls between ring 0 and 1 to move the balls between any rings, depending on which ring was found to be empty.