exercise 4: Robot Control

Install the WEBOTS simulator for simulated e-Puck robots (Version 7). See the installation instructions on the Ilias at Exercise4 (Webots Installation und Activation.pdf). The password is NOT the one indicated in this document!! It will be revealed to you at your lab session. Download from Exercise4 the example project folder with example worlds and a controller example (bangBangBasicReactive.java). In a first step, work your way through the tutorials 1 - 4 of the Webots User Guide (select "help"). Whenever you will find example code in C language, use Java instead. For this purpose, look at the example world "World_with_wallobstacles_and_epuck1.wbt" and the controller BangBangBasicReactive.java

1) (4 Pts)
Build proportional controllers in Java for a robot in a control loop architecture with the simulator Webots. Use modularization in your code: a class hierarchy where subclasses for behaviors a) to d) are very simple to implement. Be aware that you will test your code in the lab later on a real e-Puck, so you must easily scale your controllers to eventual discrepancies between real and simulated sensor behavior and environmental values (reflection, light).
The following behaviors have to be implemented:

a) when you place a robot looking in the direction of a light bulb, it should run towards the light bulb, running directly into it. Use all light sensors (an example: "ls1") for controlling. Predefined world: world_with_wall_light_and_epuck

b) same as a), but now the robot should stop in proximity of the light bulb without running into it

c) the robot should push a ball balancing it. It should work also when, at the beginning, the ball is not perfectly positioned in front of the robot.
Predefined world: World_with_wall_ball_and_epuck

d) when positioned with the left side towards a wall, the robot starts running along all 4 walls without ever running into a wall: a wall-following behavior.
Predefined world: start first with World_with_wall_and_epuck, then use a more difficult one: World_with_wallobstacles_and_epuck

2) (4 Pts)
build bang-bang controllers for behavior a) – d). You may use three different controls: run straight ahead, turn a bit (always the same "bit") to the right, turn a bit to the left.
due date for the solution to this exercise is 21/12/2015
deliverables: UML diagrams for 2); source code, class files for
the controllers to be placed in the simulator's directory "controllers" for 3)+4)