Software Engineering I
Proseminar

Winter 2007/2008

A general problem:

A company wants to automate some operations on a factory floor. The factory floor contains machines that process various parts. These machines have fixed positions on the floor. The company wants to introduce mobile robots to perform the following operations:

- Carrying parts between processing machines and between external deposits and machines
- Cleaning the floor
- Maintenance of machines
- Patrolling the floor to monitor machine operations

The company is willing to buy a fleet of specialized robots for these tasks. However, it needs first to simulate the system, in order to assess various performance issues.

You are asked to design and build a simulation tool for this system. Your software should be able to demonstrate robots at work on the factory floor.

Some of the assignments of this semester will deal with particular aspects of this complex problem.

Assignment 1

Due date: 18.10.2007, 10:00

Exercise 1

The company provides a layout of the factory floor (e.g., a blueprint), containing machine positions, access gates to the floor, and pathways between machines. Robots enter/exit the operational space through the gates and travel along the given pathways.

1. Think about the general design elements of your simulator and the relation between them. Describe a top-level model of the system.
2. Try to make your model general, by abstracting away particularities. To this end, consider other systems that are similar to the given one, and determine commonalities among all these systems.
3. Describe expected input/output and important problems that your software should resolve.

Exercise 2

Write a program that allows an operator to send messages to robots. Each robot is specialized in one of the operations mentioned above (which gives the robot's type). Each robot has a unique identification number. Interaction with the operator is done by standard input/output.

The following interaction pattern should be satisfied:
- At startup, the operator sees a list of available robot types.

- To power up a robot, the operator sends messages of the form:
`PowerOn ROBOT_TYPE`

- To see the robots' status, the operator sends the message:
`GetStatus`

- Consequently, the following information is displayed on the console:
    - o Which types of robots are operational
    - o The identification numbers of the operational robots
    - o For each robot type, the messages that can be sent to that type and their parameters.

- The operator types in messages for robots. Each message is of the form:
`ROBOT_ID.message(param1, param2,… )`

- The robot's answer to the message is displayed on the console.

- To shut down the robots, the operator sends the message:
`ShutDown`

You are free to make design choices regarding robot types (properties, messages, etc.). Your choices should allow an intuitive operation of the system. If a new robot type is added to the system, your code should suffer minimal changes.

Hints:
- Define a class for each type of robot
- Use the Java Reflection API

The checking interface:

```
package checking;
public interface CheckerInterface {
/*
* Starts operation of the system.
*/
public void start();
/*
* Terminates system operation: releases all resources.
* Make sure that this method returns (no System.exit here).
*/
public void stop();
/*
* Returns the student numbers of the authors. Each number is
* represented as a String
*/
public String[] getStudentIds();
}
```