

Java Applications

CS4000 Section 003

Term: Summer 2, 2002

Project #2: CORBA wrapper for RoboCode

Introduction

You are required to build a CORBA wrapper for the robots of the RoboCode simulator. The Web Page of the simulator is

<http://robocode.alphaworks.ibm.com/home/home.html>.

You can download the simulator from

<http://robocode.alphaworks.ibm.com/install/Install.html>.

The CORBA wrapper consists of a CORBA server and a CORBA client. The CORBA server runs as a normal robot inside the RoboCode simulator. The CORBA client is a Java program possibly running on a different computer, that contains the real brain of the robot. In order to take part to the battle, the client connects to the CORBA server and executes methods from the IDL interface.

Description

The system is composed of two parts:

1. a GUI (and CORBA client) that allows users to type commands for the robot to execute, and sends the commands to the simulation;
2. a Robot instance that runs inside the simulation and acts as a CORBA server, accepting commands from the GUI.

The minimum requirement for the GUI is to provide a window containing a text box where the user can type commands, a button that sends the

command to the simulation, and a text area where the results, that the methods return, are shown. The syntax of commands can be chosen by the programmer, but there must be one command for each available method.

The IDL interface must provide methods ahead, back, doNothing, fire, getBattleFieldHeight, getBattleFieldWidth, getEnergy, getGunHeading, getHeading, getHeight, getName, getOthers, getRadarHeading, getVelocity, getWidth, getX, getY, turnGunLeft, turnGunRight, turnLeft, turnRight. Every time the client invokes one of the above methods, the server must call the corresponding method of the Robot class (see RoboCode API documentation), and return to the client the result that the method invocation returns.

The following program shows the skeleton for the Robot object. If you are interested, other programs can be found among the samples that come with the simulation.

```
import robocode.*;

// other imports are probably needed here

public class CORBARobot extends Robot
{
    /**
     * run: Robot's main run function
     */
    public void run() {
        CORBAServer c=new CORBAServer(this);
        while (true)
        {
            // do nothing: just wait for client's requests
        }
    }
}

class CORBAServer ... // treat it like a normal CORBA server class
{
    public CORBAServer(CORBARobot r)
    {
        // store r in an instance variable. Invocations of methods defined by the simulation will
```

```
    // have to be done using this reference. For example, if you want to call  
    // method fire(), you will execute robot.fire() – assuming that robot is the name  
    // of your instance variable.  
  }  
}
```