Errors during importing KST to SunriseWorkbench

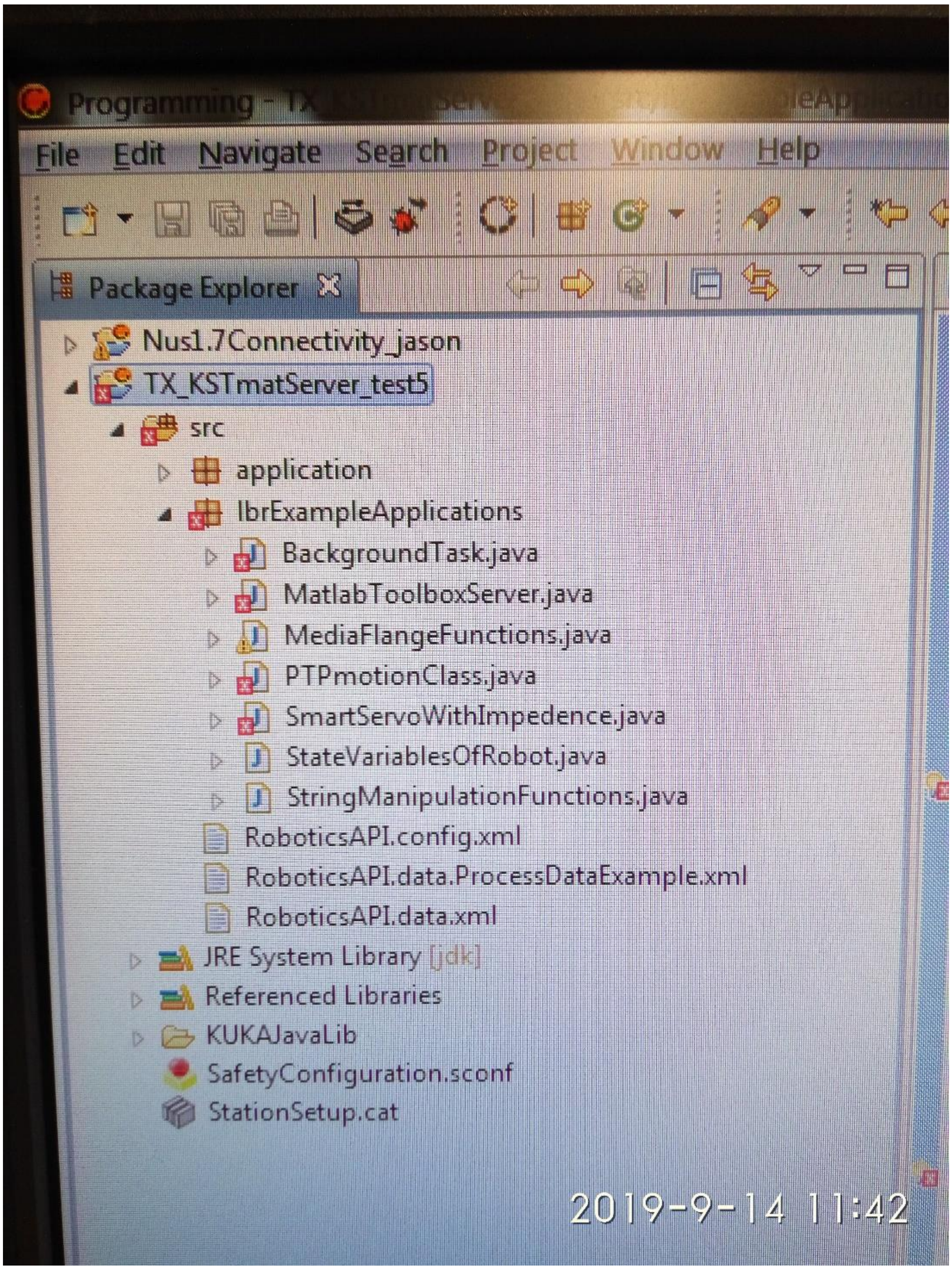
Dear Dr. Safeea,

I'm following your KST tutorials and I encounter some problems.

I'm now trying to import KST to SunriseWorkbench (Version 1.7). But there are some error icons near the copied java files. I have added the Direct Servo Motion Extension and Smart Servo Motion Extension in the StationSetup.cat, however, the errors still exist. The specific error messages are listed in the below pictures.

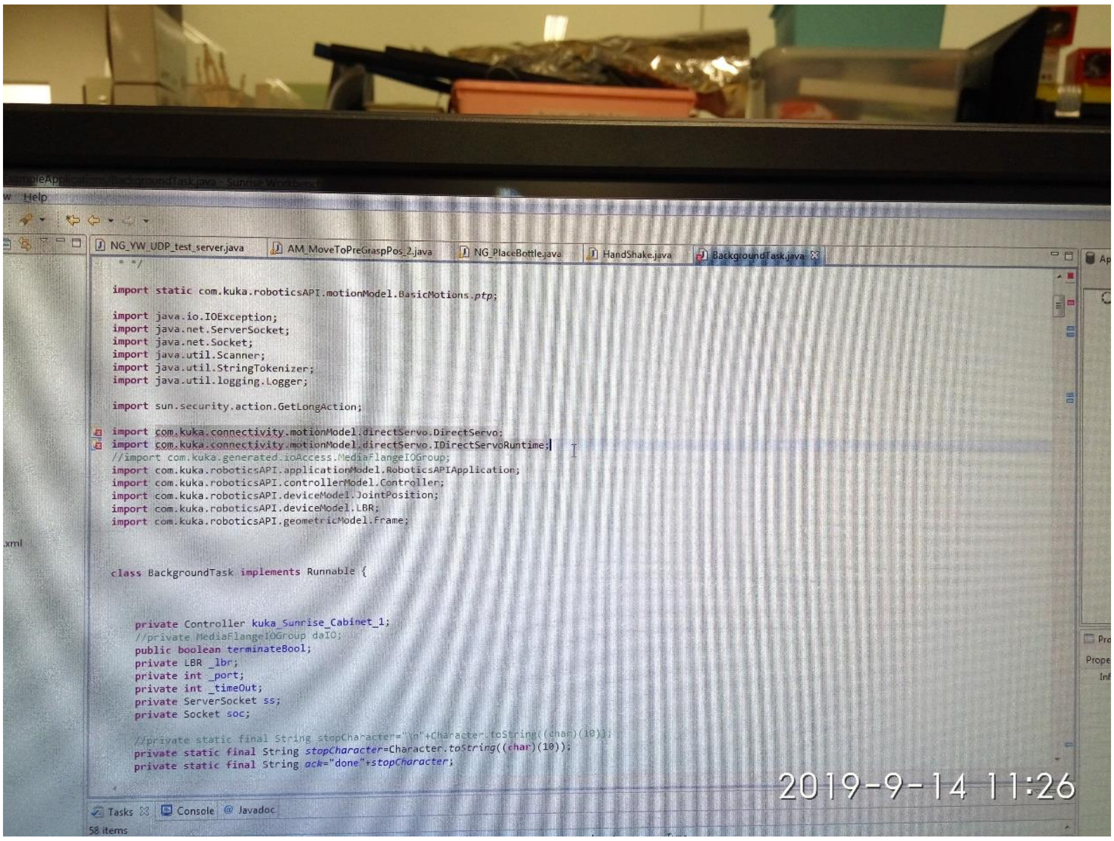
I have tried to select different media flange when establishing a new project and to use different KST version from KST_1.6 to KST_1.7, but they are no use. Could you help me analyze the reasons of the problems that I have? I really don't know how to deal with it. Thanks a lot.

Tian Xu.

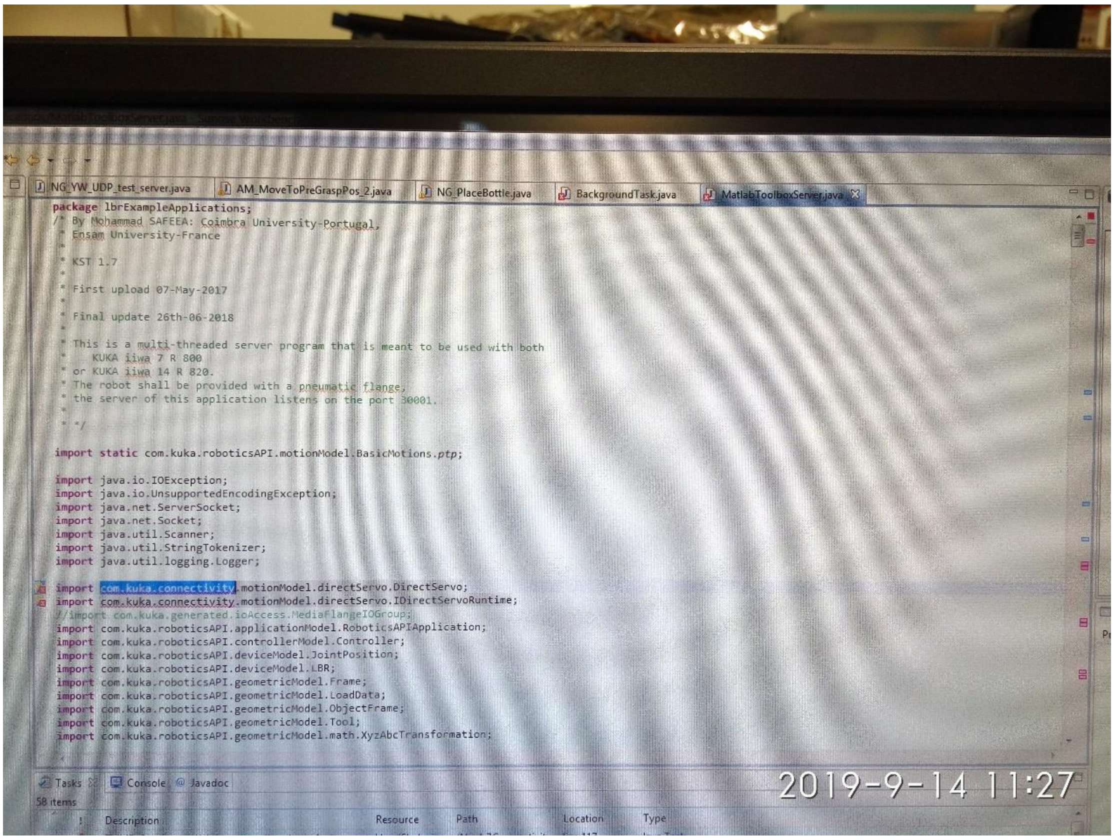


```
▷ Nus1.7Connectivity_jason
◢ TX_KSTmatServer_test5
    ◢ src
        ▷ application
        ◢ lbrExampleApplications
            ▷ BackgroundTask.java
            ▷ MatlabToolboxServer.java
            ▷ MediaFlangeFunctions.java
            ▷ PTPmotionClass.java
            ▷ SmartServoWithImpedence.java
            ▷ StateVariablesOfRobot.java
            ▷ StringManipulationFunctions.java
          RoboticsAPI.config.xml
          RoboticsAPI.data.ProcessDataExample.xml
          RoboticsAPI.data.xml
    ▷ JRE System Library [jdk]
    ▷ Referenced Libraries
    ▷ KUKAJavaLib
      SafetyConfiguration.sconf
      StationSetup.cat
```

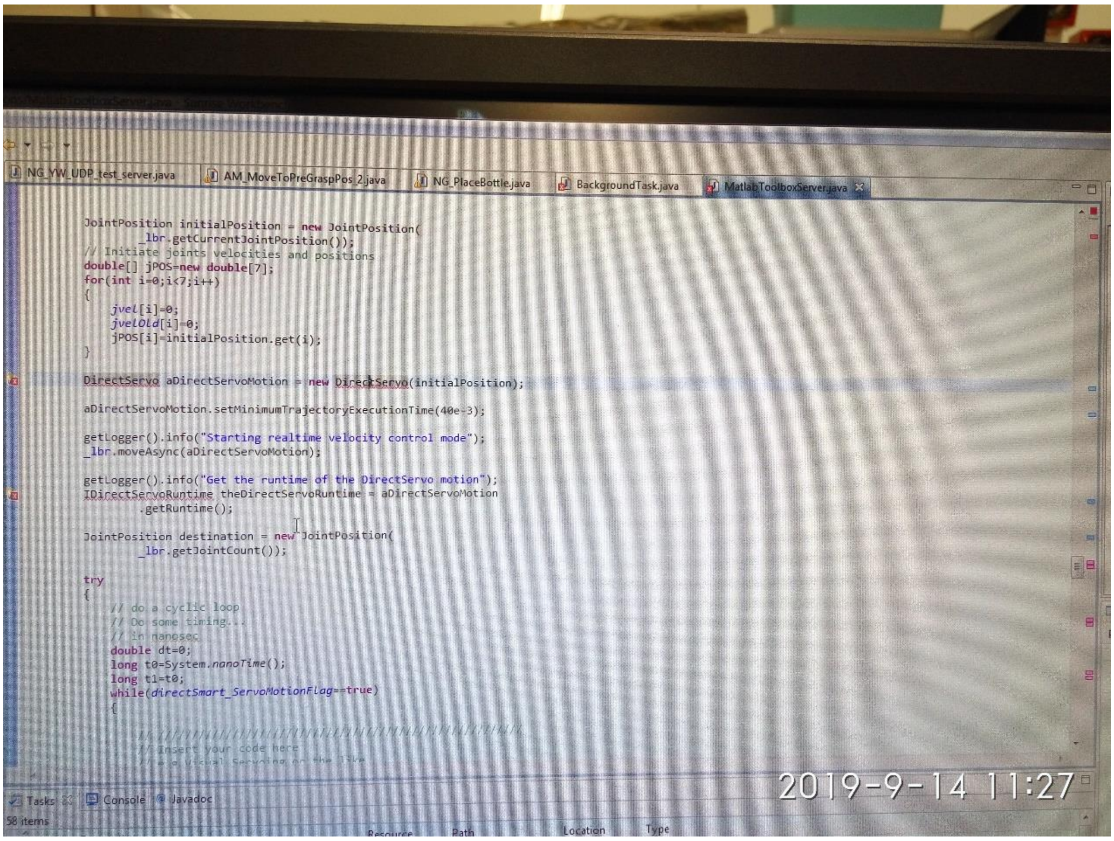
2019-9-14 11:42



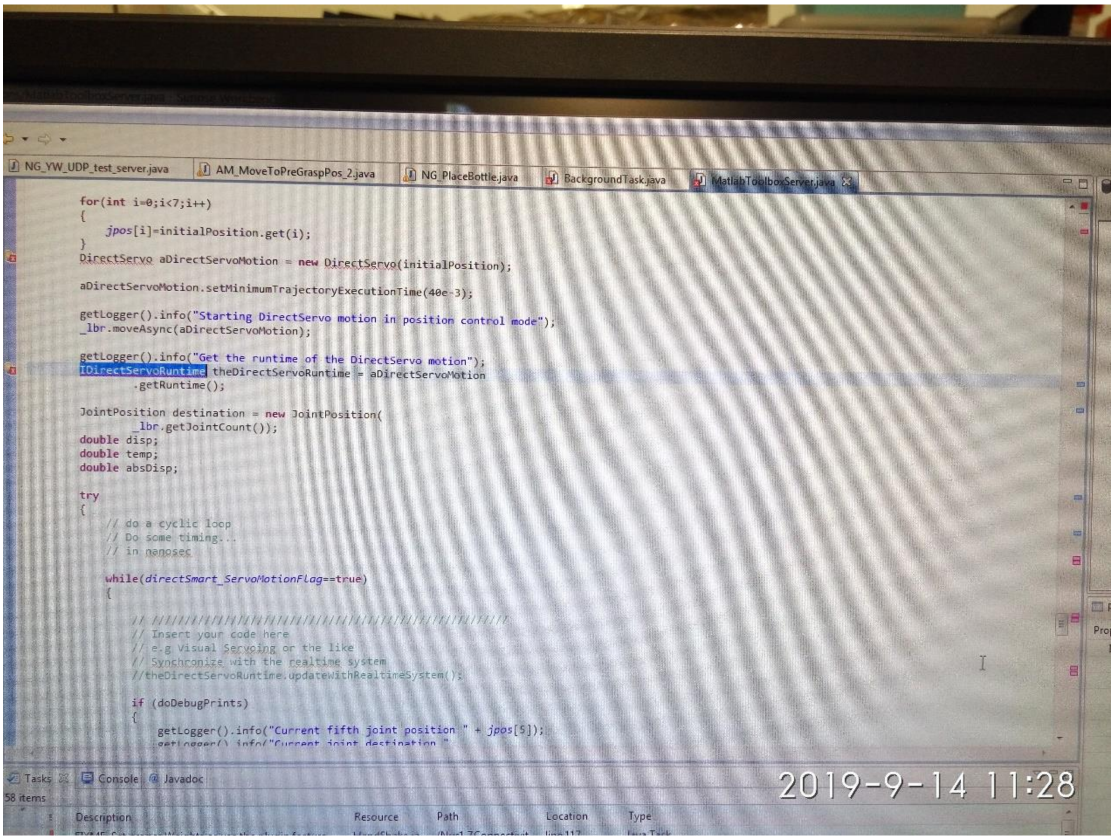
```
 * */

import static com.kuka.roboticsAPI.motionModel.BasicMotions.ptp;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.logging.Logger;

import sun.security.action.GetLongAction;

import com.kuka.connectivity.motionModel.directServo.DirectServo;
import com.kuka.connectivity.motionModel.directServo.IDirectServoRuntime;
//import com.kuka.generated.ioAccess.MediaFlangeIOGroup;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.JointPosition;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.Frame;


class BackgroundTask implements Runnable {



    private Controller kuka_Sunrise_Cabinet_1;
    //private MediaFlangeIOGroup daIO;
    public boolean terminateBool;
    private LBR _lbr;
    private int _port;
    private int _timeOut;
    private ServerSocket ss;
    private Socket soc;

    //private static final String stopCharacter="\n"+Character.toString((char)(10));
    private static final String stopCharacter=Character.toString((char)(10));
    private static final String ack="done"+stopCharacter;
```



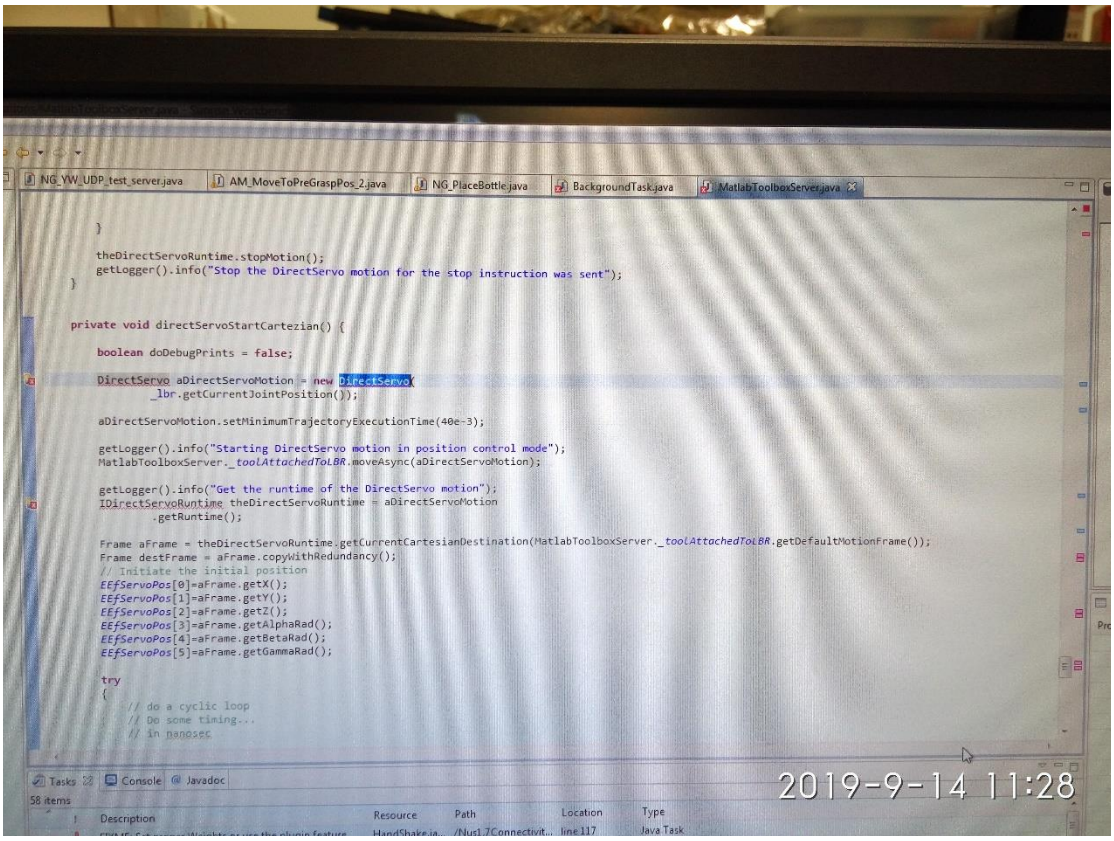
```
package lbrExampleApplications;
/* By Mohammad SAFEEA: Coimbra University-Portugal,
 * Ensam University-France
 *
 * KST 1.7
 *
 * First upload 07-May-2017
 *
 * Final update 26th-06-2018
 *
 * This is a multi-threaded server program that is meant to be used with both
 *    KUKA iiwa 7 R 800
 * or KUKA iiwa 14 R 820.
 * The robot shall be provided with a pneumatic flange,
 * the server of this application listens on the port 30001.
 *
 * */

import static com.kuka.roboticsAPI.motionModel.BasicMotions.ptp;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.logging.Logger;

import com.kuka.connectivity.motionModel.directServo.DirectServo;
import com.kuka.connectivity.motionModel.directServo.IDirectServoRuntime;
//import com.kuka.generated.ioAccess.MediaFlangeIOGroup;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.JointPosition;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.Frame;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.ObjectFrame;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.math.XyzAbcTransformation;
```



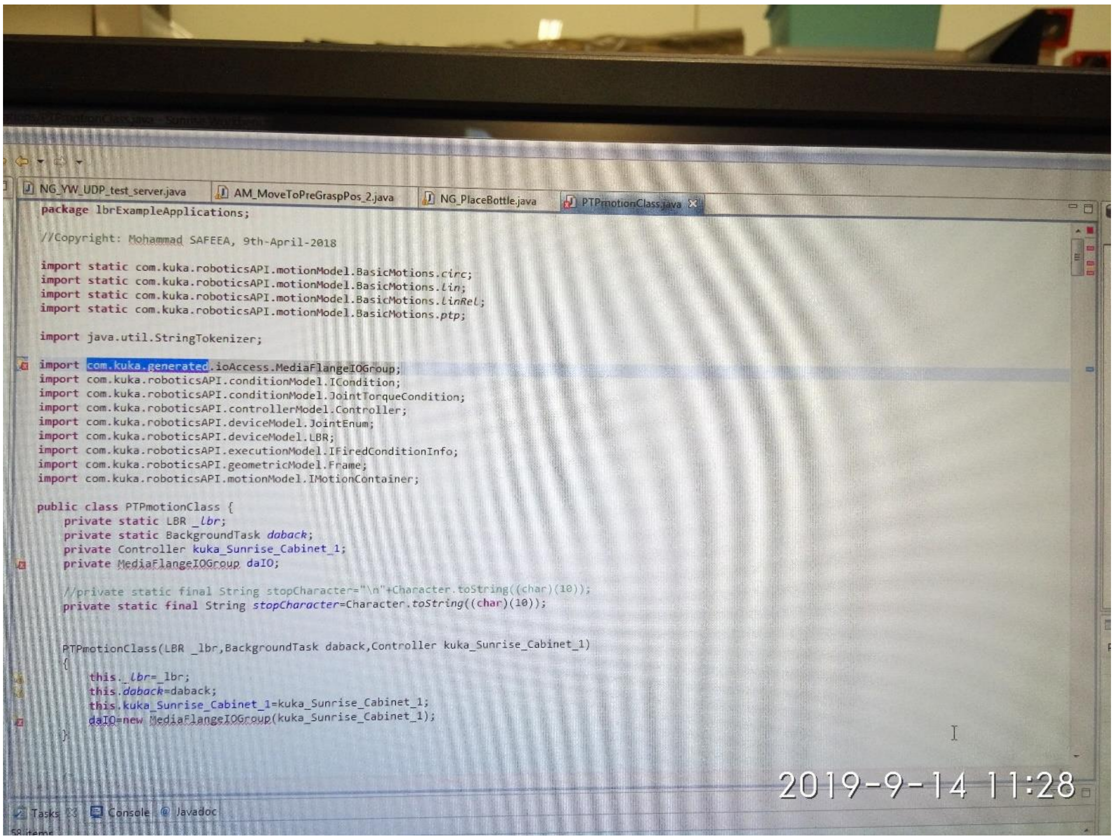
```
JointPosition initialPosition = new JointPosition(
        _lbr.getCurrentJointPosition());
// Initiate joints velocities and positions
double[] jPOS=new double[7];
for(int i=0;i<7;i++)
{
    jvel[i]=0;
    jvelOld[i]=0;
    jPOS[i]=initialPosition.get(i);
}

DirectServo aDirectServoMotion = new DirectServo(initialPosition);

aDirectServoMotion.setMinimumTrajectoryExecutionTime(40e-3);

getLogger().info("Starting realtime velocity control mode");
_lbr.moveAsync(aDirectServoMotion);

getLogger().info("Get the runtime of the DirectServo motion");
IDirectServoRuntime theDirectServoRuntime = aDirectServoMotion
        .getRuntime();

JointPosition destination = new JointPosition(
        _lbr.getJointCount());

try
{
    // do a cyclic loop
    // Do some timing...
    // in nanosec
    double dt=0;
    long t0=System.nanoTime();
    long t1=t0;
    while(directSmart_ServoMotionFlag==true)
    {

        // ///////////////////////////////////////////////////////
        // Insert your code here
        // e.g Visual Servoing or the like
```



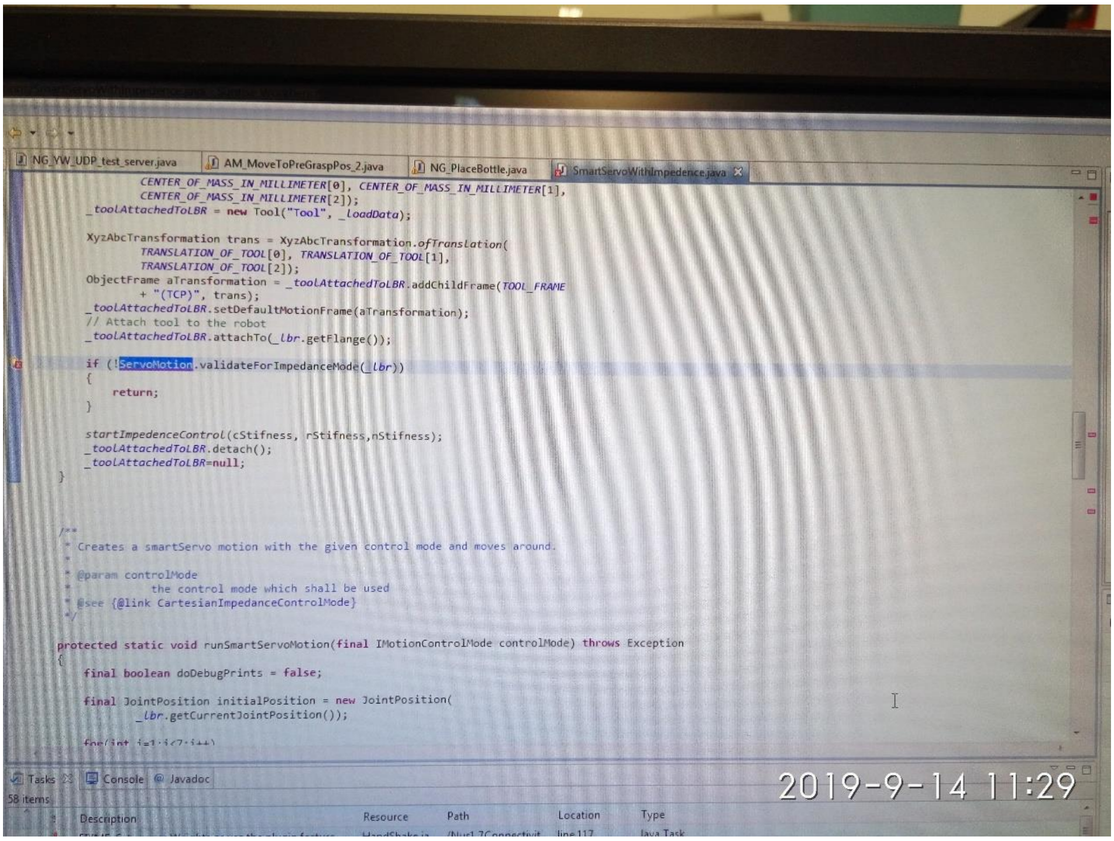
```
for(int i=0;i<7;i++)
{
    jpos[i]=initialPosition.get(i);
}
DirectServo aDirectServoMotion = new DirectServo(initialPosition);

aDirectServoMotion.setMinimumTrajectoryExecutionTime(40e-3);

getLogger().info("Starting DirectServo motion in position control mode");
_lbr.moveAsync(aDirectServoMotion);

getLogger().info("Get the runtime of the DirectServo motion");
IDirectServoRuntime theDirectServoRuntime = aDirectServoMotion
        .getRuntime();

JointPosition destination = new JointPosition(
        _lbr.getJointCount());
double disp;
double temp;
double absDisp;

try
{
    // do a cyclic loop
    // Do some timing...
    // in nanosec

    while(directSmart_ServoMotionFlag==true)
    {

        // ///////////////////////////////////////////////////////
        // Insert your code here
        // e.g Visual Servoing or the like
        // Synchronize with the realtime system
        //theDirectServoRuntime.updateWithRealtimeSystem();

        if (doDebugPrints)
        {
            getLogger().info("Current fifth joint position " + jpos[5]);
            getLogger().info("Current joint destination "
```



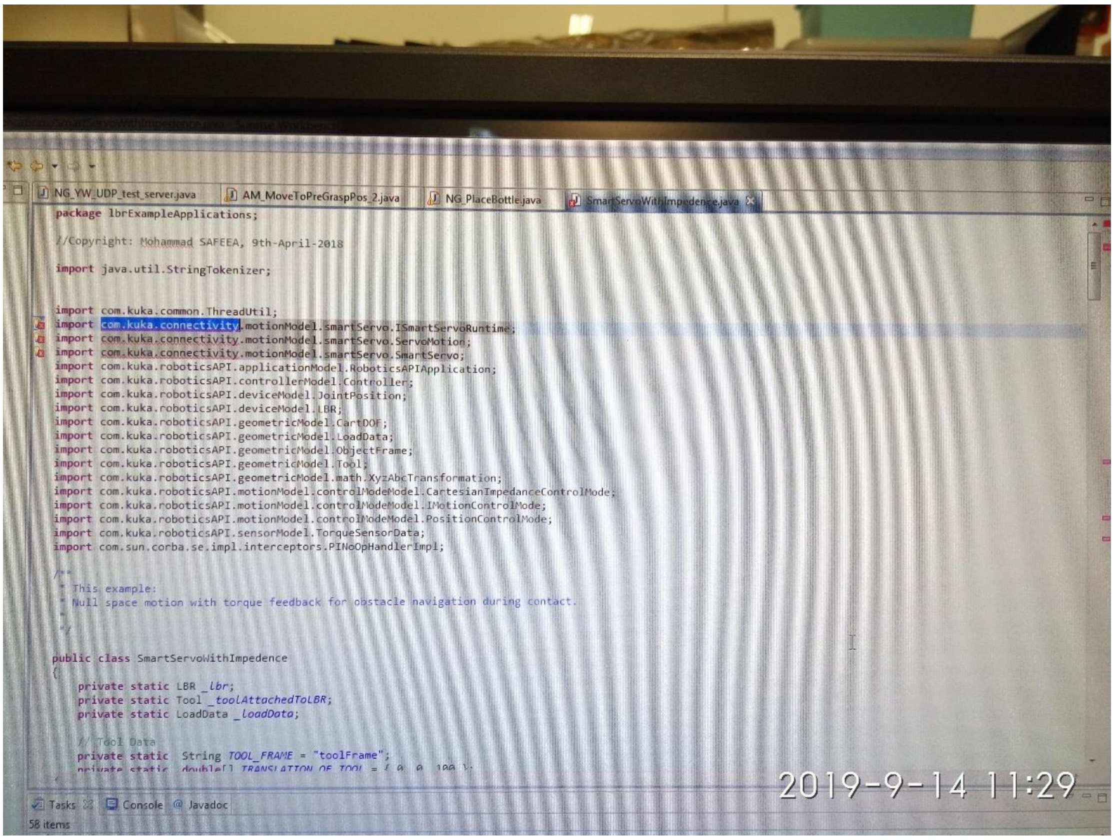
```
        }

        theDirectServoRuntime.stopMotion();
        getLogger().info("Stop the DirectServo motion for the stop instruction was sent");
    }


    private void directServoStartCartezian() {

        boolean doDebugPrints = false;

        DirectServo aDirectServoMotion = new DirectServo(
                _lbr.getCurrentJointPosition());

        aDirectServoMotion.setMinimumTrajectoryExecutionTime(40e-3);

        getLogger().info("Starting DirectServo motion in position control mode");
        MatlabToolboxServer._toolAttachedToLBR.moveAsync(aDirectServoMotion);

        getLogger().info("Get the runtime of the DirectServo motion");
        IDirectServoRuntime theDirectServoRuntime = aDirectServoMotion
                .getRuntime();

        Frame aFrame = theDirectServoRuntime.getCurrentCartesianDestination(MatlabToolboxServer._toolAttachedToLBR.getDefaultMotionFrame());
        Frame destFrame = aFrame.copyWithRedundancy();
        // Initiate the initial position
        EEfServoPos[0]=aFrame.getX();
        EEfServoPos[1]=aFrame.getY();
        EEfServoPos[2]=aFrame.getZ();
        EEfServoPos[3]=aFrame.getAlphaRad();
        EEfServoPos[4]=aFrame.getBetaRad();
        EEfServoPos[5]=aFrame.getGammaRad();

        try
        {
            // do a cyclic loop
            // Do some timing...
            // in nanosec
```



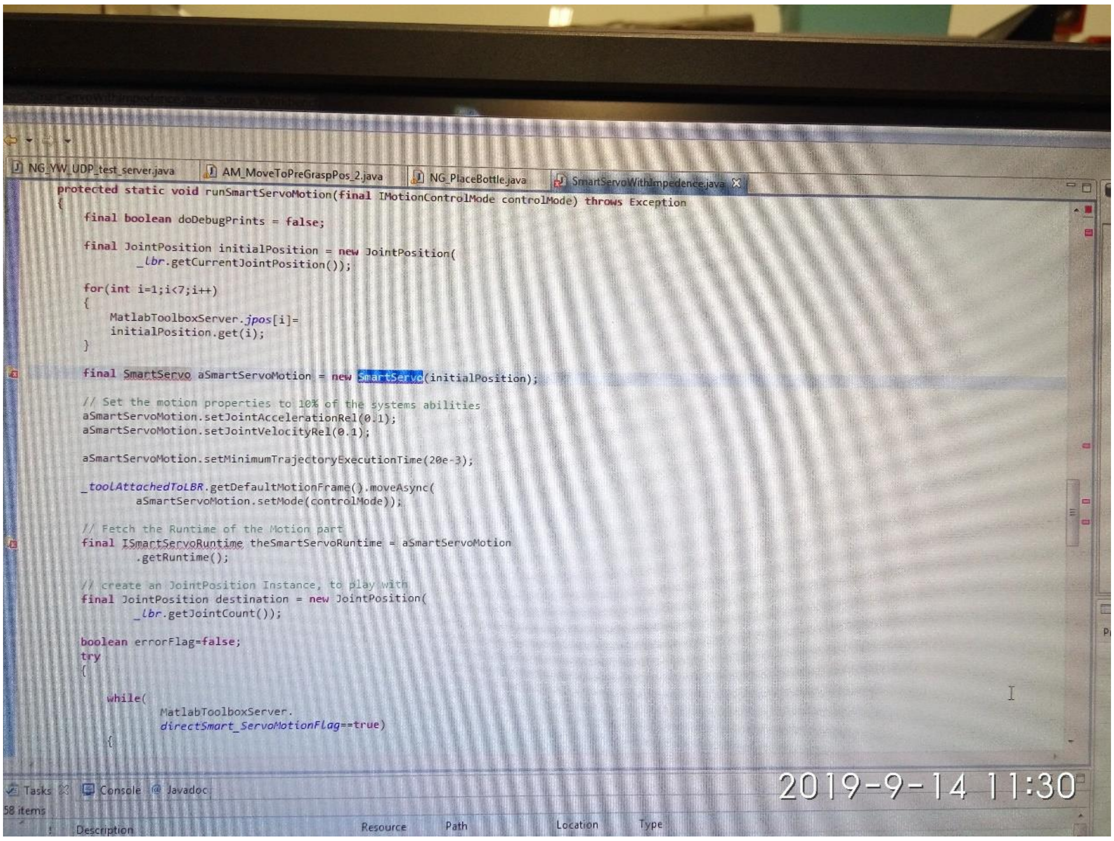
```
package lbrExampleApplications;

//Copyright: Mohammad SAFEEA, 9th-April-2018

import static com.kuka.roboticsAPI.motionModel.BasicMotions.circ;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.lin;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.linRel;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.ptp;

import java.util.StringTokenizer;

import com.kuka.generated.ioAccess.MediaFlangeIOGroup;
import com.kuka.roboticsAPI.conditionModel.ICondition;
import com.kuka.roboticsAPI.conditionModel.JointTorqueCondition;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.JointEnum;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.executionModel.IFiredConditionInfo;
import com.kuka.roboticsAPI.geometricModel.Frame;
import com.kuka.roboticsAPI.motionModel.IMotionContainer;

public class PTPmotionClass {
    private static LBR _lbr;
    private static BackgroundTask daback;
    private Controller kuka_Sunrise_Cabinet_1;
    private MediaFlangeIOGroup daIO;

    //private static final String stopCharacter="\n"+Character.toString((char)(10));
    private static final String stopCharacter=Character.toString((char)(10));


    PTPmotionClass(LBR _lbr,BackgroundTask daback,Controller kuka_Sunrise_Cabinet_1)
    {
        this._lbr=_lbr;
        this.daback=daback;
        this.kuka_Sunrise_Cabinet_1=kuka_Sunrise_Cabinet_1;
        daIO=new MediaFlangeIOGroup(kuka_Sunrise_Cabinet_1);
    }
```



```java
                CENTER_OF_MASS_IN_MILLIMETER[0], CENTER_OF_MASS_IN_MILLIMETER[1],
                CENTER_OF_MASS_IN_MILLIMETER[2]);
        _toolAttachedToLBR = new Tool("Tool", _loadData);

        XyzAbcTransformation trans = XyzAbcTransformation.ofTranslation(
                TRANSLATION_OF_TOOL[0], TRANSLATION_OF_TOOL[1],
                TRANSLATION_OF_TOOL[2]);
        ObjectFrame aTransformation = _toolAttachedToLBR.addChildFrame(TOOL_FRAME
                + "(TCP)", trans);
        _toolAttachedToLBR.setDefaultMotionFrame(aTransformation);
        // Attach tool to the robot
        _toolAttachedToLBR.attachTo(_lbr.getFlange());

        if (!ServoMotion.validateForImpedanceMode(_lbr))
        {
            return;
        }

        startImpedenceControl(cStifness, rStifness,nStifness);
        _toolAttachedToLBR.detach();
        _toolAttachedToLBR=null;
    }


    /**
     * Creates a smartServo motion with the given control mode and moves around.
     *
     * @param controlMode
     *            the control mode which shall be used
     * @see {@link CartesianImpedanceControlMode}
     */

    protected static void runSmartServoMotion(final IMotionControlMode controlMode) throws Exception
    {
        final boolean doDebugPrints = false;

        final JointPosition initialPosition = new JointPosition(
                _lbr.getCurrentJointPosition());

        for(int i=1;i<7;i++)
```


NG_YW_UDP_test_server.java
AM_MoveToPreGraspPos_2.java
NG_PlaceBottle.java
SmartServoWithImpedence.java
Tasks
Console
Javadoc
58 items
2019-9-14 11:29

```java
package lbrExampleApplications;

//Copyright: Mohammad SAFEEA, 9th-April-2018

import java.util.StringTokenizer;


import com.kuka.common.ThreadUtil;
import com.kuka.connectivity.motionModel.smartServo.ISmartServoRuntime;
import com.kuka.connectivity.motionModel.smartServo.ServoMotion;
import com.kuka.connectivity.motionModel.smartServo.SmartServo;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.JointPosition;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.CartDOF;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.ObjectFrame;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.math.XyzAbcTransformation;
import com.kuka.roboticsAPI.motionModel.controlModeModel.CartesianImpedanceControlMode;
import com.kuka.roboticsAPI.motionModel.controlModeModel.IMotionControlMode;
import com.kuka.roboticsAPI.motionModel.controlModeModel.PositionControlMode;
import com.kuka.roboticsAPI.sensorModel.TorqueSensorData;
import com.sun.corba.se.impl.interceptors.PINoOpHandlerImpl;

/**
 * This example:
 * Null space motion with torque feedback for obstacle navigation during contact.
 *
 */

public class SmartServoWithImpedence
{
    private static LBR _lbr;
    private static Tool _toolAttachedToLBR;
    private static LoadData _loadData;

    // Tool Data
    private static  String TOOL_FRAME = "toolFrame";
    private static  double[] TRANSLATION_OF_TOOL = { 0, 0, 100 };
```



NG_YW_UDP_test_server.java
AM_MoveToPreGraspPos_2.java
NG_PlaceBottle.java
SmartServoWithImpedence.java

```java
protected static void runSmartServoMotion(final IMotionControlMode controlMode) throws Exception
{
    final boolean doDebugPrints = false;

    final JointPosition initialPosition = new JointPosition(
            _lbr.getCurrentJointPosition());

    for(int i=1;i<7;i++)
    {
        MatlabToolboxServer.jpos[i]=
        initialPosition.get(i);
    }

    final SmartServo aSmartServoMotion = new SmartServo(initialPosition);

    // Set the motion properties to 10% of the systems abilities
    aSmartServoMotion.setJointAccelerationRel(0.1);
    aSmartServoMotion.setJointVelocityRel(0.1);

    aSmartServoMotion.setMinimumTrajectoryExecutionTime(20e-3);

    _toolAttachedToLBR.getDefaultMotionFrame().moveAsync(
            aSmartServoMotion.setMode(controlMode));

    // Fetch the Runtime of the Motion part
    final ISmartServoRuntime theSmartServoRuntime = aSmartServoMotion
            .getRuntime();

    // create an JointPosition Instance, to play with
    final JointPosition destination = new JointPosition(
            _lbr.getJointCount());

    boolean errorFlag=false;
    try
    {

        while(
                MatlabToolboxServer.
                directSmart_ServoMotionFlag==true)
        {
```

2019-9-14 11:30

2019-9-14 11:24