OOP Project

-Robot Dancing-

Professor: Yap Chee Een

Class: 4th class(Ra)

20142397 You HeeDong

20142366 Seo YoungSun

1. Explain of the way that implemented.

To make Robot, We basically make BasicRobot. This robot is the basic frame of robot. So We can add some character by drawing on that robot. And to make Robot move, we want to store the data of the dancing, so we use json but do not use library, we make our own jsonparser.

1. Explain of the Basic class.

To implement the project, We divide the class as 4 part. Frame, robot, main. And more we make json package to use json for don’t using library. And we use 4 resource file, character, dance, image, music. Frame class has the information of the frame, robot class has the information of each robot for example muzi, Frodo, jayG, Neo etc. By modeling BasicRobot, We can make many character, dance very easy.

1. The hierarchy of classes.

1. Source Code

Frame, robot,main + json package

-Frame class

**package** sys;

**import** java.applet.Applet;

**import** java.applet.AudioClip;

**import** java.awt.BorderLayout;

**import** java.awt.Color;

**import** java.awt.Component;

**import** java.awt.FlowLayout;

**import** java.awt.GridBagConstraints;

**import** java.awt.GridBagLayout;

**import** java.awt.Image;

**import** java.awt.KeyEventDispatcher;

**import** java.awt.KeyboardFocusManager;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.awt.event.MouseMotionListener;

**import** java.io.File;

**import** java.util.ArrayList;

**import** javax.swing.BorderFactory;

**import** javax.swing.BoxLayout;

**import** javax.swing.ButtonGroup;

**import** javax.swing.ImageIcon;

**import** javax.swing.JButton;

**import** javax.swing.JComboBox;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** javax.swing.JRadioButton;

**import** javax.swing.JTextField;

**import** javax.swing.border.BevelBorder;

**import** myJSON.JSONObject;

**public** **class** frame {

**final** **int** frameWidth=1300;

**final** **int** frameHeight=760;

JSONObject test;

JSONObject dancetest;

JFrame f;

JLabel stage;

**final** String[] musicList={"I do I do","아름다운 밤","Let it be","Cute","Marry you","Treasure","Stop"};

**final** String[] danceList={"defalut","I do I do","아름다운 밤","Muzi","Neo","Con","Frodo","Tube","Apeach","JayG"};

JPanel ComboBoxPan;

JPanel musicPan;

JLabel musicLab;

JComboBox musicBox;

JPanel dancePan;

JLabel danceLab;

JComboBox danceBox;

JButton muziImgBtn;

JButton neoImgBtn;

JButton frodoImgBtn;

JButton conImgBtn;

JButton tubeImgBtn;

JButton jayGImgBtn;

JButton apeachImgBtn;

JPanel smallBtnGrp;

JButton playBtn;

JButton pauseBtn;

JButton trashBtn;

JRadioButton muziBtn;

JRadioButton neoBtn;

JRadioButton basicRobotBtn;

JRadioButton frodoBtn;

JRadioButton conBtn;

JRadioButton tubeBtn;

JRadioButton jayGBtn;

JRadioButton apeachBtn;

JRadioButton powerRangerBtn;

AudioClip clip;

**boolean** flag=**false**;

**boolean** dancingFlag=**true**;

BasicRobot dragbot=**null**;

**int** offsetX=0;

**int** offsetY=0;

String dancePath=**null**;

ArrayList<BasicThread> threadList=**new** ArrayList<BasicThread>();

**public** frame() {

f = **new** JFrame();

f.setTitle("KaKao Friends");

f.setSize(frameWidth, frameHeight);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

test = **new** JSONObject();

test.put("X", **new** Integer(100));

test.put("Y", **new** Integer(100));

test.put("armSize", **new** Integer(10));

test.put("legSize", **new** Integer(10));

test.put("width", **new** Integer(50));

test.put("height", **new** Integer(50));

test.put("neckLength", **new** Integer(0));

test.put("neckangle", **new** Double(0));

test.put("headangle", **new** Double(0));

test.put("leftArm", **new** Integer(5));

test.put("leftArm1angle", **new** Double(0));

test.put("leftArm1Length", **new** Integer(15));

test.put("leftArm2angle", **new** Double(90));

test.put("leftArm2Length", **new** Integer(15));

test.put("rightArm", **new** Integer(5));

test.put("rightArm1angle", **new** Double(0));

test.put("rightArm1Length", **new** Integer(15));

test.put("rightArm2angle", **new** Double(60));

test.put("rightArm2Length", **new** Integer(15));

test.put("leftLegStart", **new** Integer(5));

test.put("leftLeg1angle", **new** Double(0));

test.put("leftLeg1Length", **new** Integer(15));

test.put("leftLeg2angle", **new** Double(0));

test.put("leftLeg2Length", **new** Integer(15));

test.put("rightLegStart", **new** Integer(5));

test.put("rightLeg1angle", **new** Double(0));

test.put("rightLeg1Length", **new** Integer(15));

test.put("rightLeg2angle", **new** Double(0));

test.put("rightLeg2Length", **new** Integer(15));

// System.out.println(test.toJSONString());

test.put("headWidth", **new** Integer(50));

test.put("headHeight", **new** Integer(40));

test.put("faceWidth", **new** Integer(35));

test.put("faceHeight", **new** Integer(35));

test.put("earWidth", **new** Integer(10));

test.put("earHeight", **new** Integer(30));

ArrayList<JSONObject> dance = **new** ArrayList<JSONObject>();

JSONObject frame1 = **new** JSONObject();

frame1.put("leftLeg1angle", **new** Double(45));

frame1.put("leftLeg2angle", **new** Double(-45));

frame1.put("rightLeg1angle", **new** Double(-45));

frame1.put("rightLeg2angle", **new** Double(45));

dance.add(frame1);

JSONObject frame2 = **new** JSONObject();

frame2.put("leftLeg1angle", **new** Double(0));

frame2.put("leftLeg2angle", **new** Double(0));

frame2.put("rightLeg1angle", **new** Double(0));

frame2.put("rightLeg2angle", **new** Double(0));

dance.add(frame2);

JSONObject frame3 = **new** JSONObject();

frame3.put("leftLeg1angle", **new** Double(-45));

frame3.put("leftLeg2angle", **new** Double(45));

frame3.put("rightLeg1angle", **new** Double(45));

frame3.put("rightLeg2angle", **new** Double(-45));

dance.add(frame3);

JSONObject frame4 = **new** JSONObject();

frame4.put("leftLeg1angle", **new** Double(0));

frame4.put("leftLeg2angle", **new** Double(0));

frame4.put("rightLeg1angle", **new** Double(0));

frame4.put("rightLeg2angle", **new** Double(0));

dance.add(frame1);

dancetest = **new** JSONObject();

dancetest.put("frameNum", dance.size());

dancetest.put("delay", 100);

dancetest.put("dance", dance);

// System.out.println(dancetest.toJSONString());

// robot.setRobot(test.toJSONString());

// robot.setRobot(50, 50, Math.PI/3, 0, 30, 30, 0, 0, 30);

f.setLayout(**new** GridBagLayout());

// f.setBounds(0, 0, 1300, 760);

ImageIcon backgroundImage=**new** ImageIcon("res/image/talk.png");

Image img=backgroundImage.getImage();

backgroundImage=**new** ImageIcon(img.getScaledInstance(f.getWidth(), (**int**)(f.getHeight()\*4.0/5), Image.SCALE\_SMOOTH));

stage = **new** JLabel(backgroundImage);

GridBagConstraints stage\_GBC = **new** GridBagConstraints();

stage\_GBC.gridx=0;

stage\_GBC.gridy=0;

//stage\_GBC.ipadx = frameWidth;

//stage\_GBC.ipady = 100;

stage\_GBC.anchor=GridBagConstraints.CENTER;

//stage\_GBC.weightx=0.3;

stage.setLayout(**new** BorderLayout());

// stage.setBounds(0, 0, 1200, 760);

stage.setBorder(BorderFactory.createBevelBorder(BevelBorder.RAISED));

// stage.setBackground(new Color(0,0,0,0));

JLabel btn = **new** JLabel();

// btn.setLayout(new BoxLayout(btn, BoxLayout.LINE\_AXIS));

btn.setLayout(**new** FlowLayout());

// btn.setBounds(0, 0, 500, 760);

// btn.setBackground(Color.GREEN);

GridBagConstraints btn\_GBC = **new** GridBagConstraints();

// btn\_GBC.weightx = 0.1;

// btn\_GBC.gridheight=2;

btn\_GBC.gridx=0;

btn\_GBC.gridy=1;

// btn\_GBC.fill=GridBagConstraints.HORIZONTAL;

btn\_GBC.ipadx=frameWidth;

btn\_GBC.ipady=(**int**)(f.getHeight()\*1.0/5);

//btn\_GBC.anchor=GridBagConstraints.SOUTH;

JTextField hidden=**new** JTextField();

hidden.setVisible(**false**);

f.add(hidden);

// System.out.println("test");

KeyboardFocusManager manager = KeyboardFocusManager.getCurrentKeyboardFocusManager();

manager.addKeyEventDispatcher(**new** KeyEventDispatcher() {

**public** **boolean** dispatchKeyEvent(KeyEvent e) {

// **TODO** Auto-generated method stub

**if**(e.getID()==KeyEvent.KEY\_PRESSED){

// System.out.println(e.getKeyChar());

}

**return** **false**;

}

});

ComboBoxPan=**new** JPanel();

ComboBoxPan.setLayout(**new** BoxLayout(ComboBoxPan, BoxLayout.Y\_AXIS));

musicPan=**new** JPanel();

musicLab=**new** JLabel("music");

musicBox=**new** JComboBox(musicList);

musicPan.add(musicLab);

musicPan.add(musicBox);

musicBox.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

File file=**null**;

**if**(musicBox.getSelectedItem().equals("Cute"))

{

file = **new** File("res/music/cute.wav");

}

**else** **if**(musicBox.getSelectedItem().equals("Marry you"))

{

file = **new** File("res/music/marryyou.wav");

}

**else** **if**(musicBox.getSelectedItem().equals("I do I do"))

{

file = **new** File("res/music/idoido.wav");

}

**else** **if**(musicBox.getSelectedItem().equals("아름다운 밤"))

{

file = **new** File("res/music/abam.wav");

}

**else** **if**(musicBox.getSelectedItem().equals("Let it be"))

{

file = **new** File("res/music/letitbe.wav");

}

**else** **if**(musicBox.getSelectedItem().equals("Treasure"))

{

file = **new** File("res/music/treasure.wav");

}

clip.stop();

**try** {

clip = Applet.newAudioClip(file.toURL());

clip.loop();

}**catch**(Exception a){

}

}

});

dancePan=**new** JPanel();

danceLab=**new** JLabel("dance");

danceBox=**new** JComboBox(danceList);

danceBox.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if**(danceBox.getSelectedItem().equals("defalut"))

{

dancePath=**null**;

}

**else** **if**(danceBox.getSelectedItem().equals("I do I do"))

{

dancePath="IdoIdo.json";

}

**else** **if**(danceBox.getSelectedItem().equals("아름다운 밤"))

{

dancePath="abam.json";

}

**else** **if**(danceBox.getSelectedItem().equals("Muzi"))

{

dancePath="muziDance.json";

}

**else** **if**(danceBox.getSelectedItem().equals("Neo"))

{

dancePath="neoDance.json";

}

}

});

dancePan.add(danceLab);

dancePan.add(danceBox);

ComboBoxPan.add(musicPan);

ComboBoxPan.add(dancePan);

btn.add(ComboBoxPan);

ButtonGroup btnGroup = **new** ButtonGroup();

muziBtn = **new** JRadioButton("Muzi");

neoBtn = **new** JRadioButton("Neo");

conBtn = **new** JRadioButton("Con");

frodoBtn=**new** JRadioButton("Frodo");

tubeBtn = **new** JRadioButton("Tube");

apeachBtn = **new** JRadioButton("Apeach");

jayGBtn=**new** JRadioButton("JayG");

powerRangerBtn=**new** JRadioButton("PowerRanger");

basicRobotBtn = **new** JRadioButton("BasicRobot");

btnGroup.add(muziBtn);

btnGroup.add(neoBtn);

btnGroup.add(conBtn);

btnGroup.add(frodoBtn);

btnGroup.add(tubeBtn);

btnGroup.add(apeachBtn);

btnGroup.add(jayGBtn);

btnGroup.add(powerRangerBtn);

btnGroup.add(basicRobotBtn);

basicRobotBtn.setSelected(**true**);

muziImgBtn = **new** JButton(**new** ImageIcon("res/image/Muzi.png"));

muziImgBtn.setBackground(**new** Color(255,255,255,255));

muziImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

muziBtn.setSelected(**true**);

}

});

neoImgBtn = **new** JButton(**new** ImageIcon("res/image/Neo.png"));

neoImgBtn.setBackground(**new** Color(255,255,255,255));

neoImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

neoBtn.setSelected(**true**);

}

});

conImgBtn = **new** JButton(**new** ImageIcon("res/image/Con\_.png"));

conImgBtn.setBackground(**new** Color(255,255,255,255));

conImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

conBtn.setSelected(**true**);

}

});

frodoImgBtn = **new** JButton(**new** ImageIcon("res/image/Frodo.png"));

frodoImgBtn.setBackground(**new** Color(255,255,255,255));

frodoImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

frodoBtn.setSelected(**true**);

}

});

tubeImgBtn = **new** JButton(**new** ImageIcon("res/image/Tube.png"));

tubeImgBtn.setBackground(**new** Color(255,255,255,255));

tubeImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

tubeBtn.setSelected(**true**);

}

});

apeachImgBtn = **new** JButton(**new** ImageIcon("res/image/Apeach.png"));

apeachImgBtn.setBackground(**new** Color(255,255,255,255));

apeachImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

apeachBtn.setSelected(**true**);

}

});

jayGImgBtn = **new** JButton(**new** ImageIcon("res/image/Jay-G.png"));

jayGImgBtn.setBackground(**new** Color(255,255,255,255));

jayGImgBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

jayGBtn.setSelected(**true**);

}

});

btn.add(muziImgBtn);

btn.add(neoImgBtn);

btn.add(conImgBtn);

btn.add(frodoImgBtn);

btn.add(tubeImgBtn);

btn.add(apeachImgBtn);

btn.add(jayGImgBtn);

smallBtnGrp=**new** JPanel();

smallBtnGrp.setLayout(**new** BoxLayout(smallBtnGrp,BoxLayout.Y\_AXIS));

playBtn=**new** JButton(**new** ImageIcon("res/image/play.png"));

playBtn.setBackground(**new** Color(255,255,255,255));

playBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

dancingFlag=**true**;

**for**(**int** i=0;i<threadList.size();i++){

threadList.get(i).startdancing();

}

}

});

pauseBtn=**new** JButton(**new** ImageIcon("res/image/pause.png"));

pauseBtn.setBackground(**new** Color(255,255,255,255));

pauseBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

dancingFlag=**false**;

**for**(**int** i=0;i<threadList.size();i++){

threadList.get(i).stopdancing();

}

}

});

trashBtn=**new** JButton(**new** ImageIcon("res/image/trash.png"));

trashBtn.setBackground(**new** Color(255,255,255,255));

trashBtn.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**for**(**int** i=0;i<threadList.size();i++){

threadList.get(i).stop();

stage.remove(threadList.get(i).getComponent());

}

threadList.clear();

stage.repaint();

}

});

smallBtnGrp.add(playBtn);

smallBtnGrp.add(pauseBtn);

smallBtnGrp.add(trashBtn);

btn.add(smallBtnGrp);

// btn.add(muziBtn);

// btn.add(neoBtn);

// btn.add(conBtn);

// btn.add(frodoBtn);

// btn.add(tubeBtn);

// btn.add(apeachBtn);

// btn.add(jayGBtn);

// btn.add(powerRangerBtn);

// btn.add(basicRobotBtn);

// stage.setBackground(Color.GREEN);

// stage.add(robot.getComponent());

f.add(btn, btn\_GBC);

//f.setContentPane(new JLabel(new ImageIcon("res/back.png")));

f.add(stage, stage\_GBC);

stage.addMouseMotionListener(**new** MouseMotionListener() {

**public** **void** mouseMoved(MouseEvent e) {

// **TODO** Auto-generated method stub

}

**public** **void** mouseDragged(MouseEvent e) {

// **TODO** Auto-generated method stub

**if** (flag) {

dragbot.setLocation(e.getX()-offsetX, e.getY()-offsetY);

}

}

});

stage.addMouseListener(**new** MouseListener() {

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

flag=**false**;

dragbot=**null**;

}

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

Component[] array = stage.getComponents();

**for** (**int** i = 0; i < array.length; i++) {

**if** (BasicRobot.**class**.isInstance(array[i])

&& inner((BasicRobot) array[i], e.getX(),

e.getY())) {

flag = **true**;

dragbot = (BasicRobot) array[i];

offsetX=e.getX()-dragbot.X;

offsetY=e.getY()-dragbot.Y;

// System.out.println(array[i]);

**break**;

}

}

}

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

String CharName=**null**;

BasicThread robot = **null**;

**if** (muziBtn.isSelected()) {

robot = **new** BasicThread(**new** Muzi(BasicThread

.FileRead("res/character/Muzi.txt")));

CharName = "muzi";

}

**if** (conBtn.isSelected()) {

robot = **new** BasicThread(**new** ConClass(BasicThread

.FileRead("res/character/ConClass.txt")));

CharName = "con";

}

**if** (neoBtn.isSelected()) {

robot = **new** BasicThread(**new** Neo(BasicThread

.FileRead("res/character/Neo.txt")));

CharName = "neo";

}

**if** (basicRobotBtn.isSelected()) {

robot = **new** BasicThread(**new** BasicRobot(BasicThread

.FileRead("res/character/BasicRobot.txt")));

CharName = "muzi";

}

**if** (frodoBtn.isSelected()) {

robot = **new** BasicThread(**new** Frodo(BasicThread

.FileRead("res/character/Frodo.txt")));

CharName = "frodo";

}

**if** (tubeBtn.isSelected()) {

robot = **new** BasicThread(**new** Tube(BasicThread

.FileRead("res/character/Tube.txt")));

CharName = "tube";

}

**if** (apeachBtn.isSelected()) {

robot = **new** BasicThread(**new** Apeach(BasicThread

.FileRead("res/character/Apeach.txt")));

CharName = "apeach";

}

**if** (jayGBtn.isSelected()) {

robot = **new** BasicThread(**new** JayG(BasicThread

.FileRead("res/character/JayG.txt")));

CharName = "jayg";

}

**if**(dancePath!=**null**){

robot.setDance("res/dance/"+dancePath);

}

**else**{

robot.setDance("res/dance/"+CharName+"Dance.json");

}

robot.setLocation(e.getX(), e.getY());

robot.start();

**if**(dancingFlag){

robot.startdancing();

}

**else**{

robot.stopdancing();

}

threadList.add(robot);

// System.out.println("create Basic");

stage.add(robot.getComponent(), BorderLayout.CENTER);

clip.play();

f.setVisible(**true**);

}

});

// f.add(stage);

// f.pack();

f.setVisible(**true**);

// robot.startdancing();

// robot.start();

// BasicThread.panel=stage;

**try** {

File file = **new** File("res/music/idoido.wav");

clip = Applet.newAudioClip(file.toURL());

}**catch**(Exception e){

}

}

**public** **static** **boolean** inner(BasicRobot bot,**int** x,**int** y){

**int** width=bot.getWidthC();

**int** height=bot.getHeightC();

**int** X=bot.getXC()+width/2;

**int** Y=bot.getYC()+height/2;

**if**(x>X-(width/1.8) && x<X+(width/1.8) && y>Y-(height\*2) && y<Y+height){

**return** **true**;

}

**return** **false**;

}

}

Basic Robot class

**package** sys;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** java.util.Random;

**import** javax.swing.JComponent;

**import** myJSON.JSONObject;

**import** myJSON.JSONParser;

**public** **class** BasicRobot **extends** JComponent {

**class** point {

**public** **int** x;

**public** **int** y;

**public** point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

};

**protected** **int** originY=0;

**protected** **int** X = 0;

**protected** **int** Y = 0;

**protected** **int** offset = 0;

**protected** point neck;

**protected** point head;

**protected** **double** headangle = 0;

**protected** **int** headWidth = 0;

**protected** **int** headHeight = 0;

**protected** **int** width = 0;

**protected** **int** height = 0;

**protected** **int** armSize = 0;

**protected** **int** legSize = 0;

**protected** point leftShoulder;

**protected** point leftElbow;

**protected** point leftHand;

**protected** point rightShoulder;

**protected** point rightElbow;

**protected** point rightHand;

**protected** point leftLeg;

**protected** point leftKnee;

**protected** point leftFoot;

**protected** point rightLeg;

**protected** point rightKnee;

**protected** point rightFoot;

**protected** **int** neckLength;

**protected** **double** neckangle;

**protected** **int** leftArm;

**protected** **double** leftArm1angle;

**protected** **int** leftArm1Length;

**protected** **double** leftArm2angle;

**protected** **int** leftArm2Length;

**protected** **int** rightArm;

**protected** **double** rightArm1angle;

**protected** **int** rightArm1Length;

**protected** **double** rightArm2angle;

**protected** **int** rightArm2Length;

**protected** **int** leftLegStart;

**protected** **double** leftLeg1angle;

**protected** **int** leftLeg1Length;

**protected** **double** leftLeg2angle;

**protected** **int** leftLeg2Length;

**protected** **int** rightLegStart;

**protected** **double** rightLeg1angle;

**protected** **int** rightLeg1Length;

**protected** **double** rightLeg2angle;

**protected** **int** rightLeg2Length;

**protected** **boolean** gravity = **true**;

**protected** **boolean** diffFlag=**false**;

**protected** **double** diff;

**public** **void** setLocation(**int** x,**int** y){

X=x;

Y=y;

originY=y;

**this**.calcPoint();

}

**public** BasicRobot(String jsonString) {

Random rand=**new** Random();

JSONObject json = **null**;

json = (JSONObject) **new** JSONParser().parse(jsonString);

jsonRead(json);

**if** (gravity) {

**int** height = **this**.height;

**int** leftKnee = (**int**) (height + (leftLeg1Length \* Math

.cos(leftLeg1angle)));

**int** leftFoot = (**int**) (leftKnee + (leftLeg2Length \* Math

.cos(leftLeg2angle)));

**int** rightKnee = (**int**) (height + (rightLeg1Length \* Math

.cos(rightLeg1angle)));

**int** rightFoot = (**int**) (rightKnee + (rightLeg2Length \* Math

.cos(rightLeg2angle)));

offset = Math.max(

Math.max(Math.max(leftFoot, leftKnee),

Math.max(rightFoot, rightKnee)), height);

}

diff=rand.nextDouble()+0.5;

**if**(!diffFlag){

diff=1;

}

**this**.calcPoint();

}

**public** **void** calcPoint(){

**if** (gravity) {

**int** height = **this**.height;

**int** leftKnee = (**int**) (height + (leftLeg1Length \* Math

.cos(leftLeg1angle)));

**int** leftFoot = (**int**) (leftKnee + (leftLeg2Length \* Math

.cos(leftLeg2angle)));

**int** rightKnee = (**int**) (height + (rightLeg1Length \* Math

.cos(rightLeg1angle)));

**int** rightFoot = (**int**) (rightKnee + (rightLeg2Length \* Math

.cos(rightLeg2angle)));

Y =originY+offset

- Math.max(

Math.max(Math.max(leftFoot, leftKnee),

Math.max(rightFoot, rightKnee)), height);

}

**this**.leftShoulder = **new** point(X + 0, Y + armSize / 2 + leftArm);

// tempX=-l1L tempY=armSize/2

**this**.leftElbow = **new** point(

(**int**) (**this**.leftShoulder.x + ((-leftArm1Length \* Math

.cos(leftArm1angle)))),

(**int**) (**this**.leftShoulder.y + ((-leftArm1Length \* Math

.sin(leftArm1angle)))));

**this**.leftHand = **new** point(

(**int**) (**this**.leftElbow.x + ((-leftArm2Length \* Math

.cos(leftArm2angle)))),

(**int**) (**this**.leftElbow.y + ((-leftArm2Length \* Math

.sin(leftArm2angle)))));

**this**.rightShoulder = **new** point(X + **this**.width, Y + armSize / 2

+ rightArm);

// tempX=width+l1L tempY=armSize/2

**this**.rightElbow = **new** point(

(**int**) (**this**.rightShoulder.x + (((rightArm1Length) \* Math

.cos(rightArm1angle)))),

(**int**) (**this**.rightShoulder.y + (((rightArm1Length) \* Math

.sin(rightArm1angle)))));

**this**.rightHand = **new** point(

(**int**) (**this**.rightElbow.x + ((rightArm2Length \* Math

.cos(rightArm2angle)))),

(**int**) (**this**.rightElbow.y + ((rightArm2Length \* Math

.sin(rightArm2angle)))));

**this**.leftLeg = **new** point(X + (legSize / 2) + leftLegStart, Y + height);

**this**.leftKnee = **new** point(

(**int**) (**this**.leftLeg.x + (-leftLeg1Length \* Math

.sin(leftLeg1angle))),

(**int**) (**this**.leftLeg.y + (leftLeg1Length \* Math

.cos(leftLeg1angle))));

**this**.leftFoot = **new** point(

(**int**) (**this**.leftKnee.x + (-leftLeg2Length \* Math

.sin(leftLeg2angle))),

(**int**) (**this**.leftKnee.y + (leftLeg2Length \* Math

.cos(leftLeg2angle))));

**this**.rightLeg = **new** point(X + width - (legSize / 2) - rightLegStart, Y

+ height);

**this**.rightKnee = **new** point(

(**int**) (**this**.rightLeg.x + (-rightLeg1Length \* Math

.sin(rightLeg1angle))),

(**int**) (**this**.rightLeg.y + (rightLeg1Length \* Math

.cos(rightLeg1angle))));

**this**.rightFoot = **new** point(

(**int**) (**this**.rightKnee.x + (-rightLeg2Length \* Math

.sin(rightLeg2angle))),

(**int**) (**this**.rightKnee.y + (rightLeg2Length \* Math

.cos(rightLeg2angle))));

**this**.neck = **new** point(X + (width / 2), Y);

**this**.head = **new** point(

(**int**) (**this**.neck.x + (neckLength \* Math.sin(neckangle))),

(**int**) (**this**.neck.y + (-neckLength \* Math.cos(neckangle))));

}

**public** **void** setRobot(String jsonString) {

JSONObject json = **null**;

json =JSONParser.parse(jsonString);

**this**.setRobot(json);

}

**public** **void** setRobot(JSONObject json) {

**this**.jsonRead(json);

**this**.calcPoint();

}

**public** **void** jsonRead(JSONObject json){

**try** {

X = ((Long) json.get("X")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

Y = ((Long) json.get("Y")).intValue();

originY=Y;

} **catch** (NullPointerException e) {

}

**try** {

armSize = ((Long) json.get("armSize")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

legSize = ((Long) json.get("legSize")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

width = ((Long) json.get("width")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

height = ((Long) json.get("height")).intValue();

} **catch** (NullPointerException e) {

}

**try**{

headWidth = ((Long) json.get("headWidth")).intValue();

}**catch**(NullPointerException e){}

**try**{

headHeight = ((Long) json.get("headHeight")).intValue();

}**catch**(NullPointerException e){}

**try** {

headangle = Math.toRadians(((Double) json.get("headangle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

neckLength = ((Long) json.get("neckLength")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

neckangle = Math.toRadians(((Double) json.get("neckangle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm = ((Long) json.get("leftArm")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftArm1angle = Math.toRadians(((Double) json.get("leftArm1angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm1Length = ((Long) json.get("leftArm1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftArm2angle = Math.toRadians(((Double) json.get("leftArm2angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm2Length = ((Long) json.get("leftArm2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm = ((Long) json.get("rightArm")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm1angle = Math.toRadians(((Double) json

.get("rightArm1angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightArm1Length = ((Long) json.get("rightArm1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm2angle = Math.toRadians(((Double) json

.get("rightArm2angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightArm2Length = ((Long) json.get("rightArm2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLegStart = ((Long) json.get("leftLegStart")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLeg1angle = Math.toRadians(((Double) json.get("leftLeg1angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftLeg1Length = ((Long) json.get("leftLeg1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLeg2angle = Math.toRadians(((Double) json.get("leftLeg2angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftLeg2Length = ((Long) json.get("leftLeg2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLegStart = ((Long) json.get("rightLegStart")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLeg1angle = Math.toRadians(((Double) json

.get("rightLeg1angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightLeg1Length = ((Long) json.get("rightLeg1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLeg2angle = Math.toRadians(((Double) json

.get("rightLeg2angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightLeg2Length = ((Long) json.get("rightLeg2Length")).intValue();

} **catch** (NullPointerException e) {

}

}

**public** **void** dancing(JSONObject json){

**int** FPS=0;

**int** time=0;

**int** X = 0;

**int** Y = 0;

**double** headangle = 0;

**int** headWidth = 0;

**int** headHeight = 0;

**int** width = 0;

**int** height = 0;

**int** armSize = 0;

**int** legSize = 0;

**int** neckLength=0;

**double** neckangle=0;

**int** leftArm=0;

**double** leftArm1angle=0;

**int** leftArm1Length=0;

**double** leftArm2angle=0;

**int** leftArm2Length=0;

**int** rightArm=0;

**double** rightArm1angle=0;

**int** rightArm1Length=0;

**double** rightArm2angle=0;

**int** rightArm2Length=0;

**int** leftLegStart=0;

**double** leftLeg1angle=0;

**int** leftLeg1Length=0;

**double** leftLeg2angle=0;

**int** leftLeg2Length=0;

**int** rightLegStart=0;

**double** rightLeg1angle=0;

**int** rightLeg1Length=0;

**double** rightLeg2angle=0;

**int** rightLeg2Length=0;

**try**{

time=((Long)json.get("time")).intValue();

}**catch**(NullPointerException e){

}

**try**{

FPS=((Long)json.get("FPS")).intValue();

}**catch**(NullPointerException e){

}

**try** {

X = ((Long) json.get("X")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

Y = ((Long) json.get("Y")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

armSize = ((Long) json.get("armSize")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

legSize = ((Long) json.get("legSize")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

width = ((Long) json.get("width")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

height = ((Long) json.get("height")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

headangle = Math.toRadians(((Double) json.get("headangle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

neckLength = ((Long) json.get("neckLength")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

neckangle = Math.toRadians(((Double) json.get("neckangle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm = ((Long) json.get("leftArm")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftArm1angle = Math.toRadians(((Double) json.get("leftArm1angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm1Length = ((Long) json.get("leftArm1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftArm2angle = Math.toRadians(((Double) json.get("leftArm2angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftArm2Length = ((Long) json.get("leftArm2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm = ((Long) json.get("rightArm")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm1angle = Math.toRadians(((Double) json

.get("rightArm1angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightArm1Length = ((Long) json.get("rightArm1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightArm2angle = Math.toRadians(((Double) json

.get("rightArm2angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightArm2Length = ((Long) json.get("rightArm2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLegStart = ((Long) json.get("leftLegStart")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLeg1angle = Math.toRadians(((Double) json.get("leftLeg1angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftLeg1Length = ((Long) json.get("leftLeg1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

leftLeg2angle = Math.toRadians(((Double) json.get("leftLeg2angle"))

.doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

leftLeg2Length = ((Long) json.get("leftLeg2Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLegStart = ((Long) json.get("rightLeg")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLeg1angle = Math.toRadians(((Double) json

.get("rightLeg1angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightLeg1Length = ((Long) json.get("rightLeg1Length")).intValue();

} **catch** (NullPointerException e) {

}

**try** {

rightLeg2angle = Math.toRadians(((Double) json

.get("rightLeg2angle")).doubleValue());

} **catch** (NullPointerException e) {

}

**try** {

rightLeg2Length = ((Long) json.get("rightLeg2Length")).intValue();

} **catch** (NullPointerException e) {

}

**for**(**int** i=0;i<time/(1000/FPS);i++){

**this**.X += X/FPS;

**this**.Y += Y/FPS;

**this**.headangle += headangle/FPS;

**this**.headWidth += headWidth/FPS;

**this**.headHeight += headHeight/FPS;

**this**.width += width/FPS;

**this**.height += height/FPS;

**this**.armSize += armSize/FPS;

**this**.legSize += legSize/FPS;

**this**.neckLength+=neckLength/FPS;

**this**.neckangle+=neckangle/FPS;

**this**.leftArm+=leftArm/FPS;

**this**.leftArm1angle+=leftArm1angle/FPS;

**this**.leftArm1Length+=leftArm1Length/FPS;

**this**.leftArm2angle+=leftArm2angle/FPS;

**this**.leftArm2Length+=leftArm2Length/FPS;

**this**.rightArm+=rightArm/FPS;

**this**.rightArm1angle+=rightArm1angle/FPS;

**this**.rightArm1Length+=rightArm1Length/FPS;

**this**.rightArm2angle+=rightArm2angle/FPS;

**this**.rightArm2Length+=rightArm2Length/FPS;

**this**.leftLegStart+=leftLegStart/FPS;

**this**.leftLeg1angle+=leftLeg1angle/FPS;

**this**.leftLeg1Length+=leftLeg1Length/FPS;

**this**.leftLeg2angle+=leftLeg2angle/FPS;

**this**.leftLeg2Length+=leftLeg2Length/FPS;

**this**.rightLegStart+=rightLegStart/FPS;

**this**.rightLeg1angle+=rightLeg1angle/FPS;

**this**.rightLeg1Length+=rightLeg1Length/FPS;

**this**.rightLeg2angle+=rightLeg2angle/FPS;

**this**.rightLeg2Length+=rightLeg2Length/FPS;

**this**.calcPoint();

**this**.repaint();

**try**{

Thread.sleep((**int**)((1000/FPS)\*diff));

}**catch**(Exception e){

}

}

}

**protected** **void** paintComponent(Graphics g) {

// **TODO** Auto-generated method stub

Graphics2D g2 = (Graphics2D) g;

g2.drawRect(X, Y, width, height);

// System.out.println(leftShoulder.x+" "+leftShoulder.y);

// System.out.println(leftElbow.y+" "+leftElbow.y);

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x,

rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.drawLine(neck.x, neck.y, head.x, head.y);

g2.rotate(headangle, head.x, head.y);

g2.drawRect(head.x - (headWidth / 2), head.y - headHeight, headWidth, headHeight);

}

**public** **int** getWidthC(){

**return** **this**.width;

}

**public** **int** getHeightC(){

**return** **this**.height;

}

**public** **int** getXC(){

**return** **this**.X;

}

**public** **int** getYC(){

**return** **this**.Y;

}

}

Basic Thread class

**package** sys;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.text.ParseException;

**import** java.util.ArrayList;

**import** javax.swing.JPanel;

**import** myJSON.JSONArray;

**import** myJSON.JSONObject;

**import** myJSON.JSONParser;

**public** **class** BasicThread **extends** Thread **implements** dancingrobots {

**private** BasicRobot robot;

**private** JSONObject begin;

**private** **boolean** dancing = **true**;

**public** **static** JPanel panel=**null**;

**private** ArrayList<JSONObject> dance = **new** ArrayList<JSONObject>();

**private** **int** delay = 0;

**public** BasicThread(String FileName) {

String jsonText = FileRead(FileName);

robot = **new** Muzi(jsonText);

}

**public** BasicThread(BasicRobot robot) {

**this**.robot = robot;

}

**public** **static** String FileRead(String FileName){

String jsonText="";

**try** {

BufferedReader in = **new** BufferedReader(**new** FileReader(FileName));

String s;

**while** ((s = in.readLine()) != **null**) {

jsonText += s + "\n";

}

in.close();

} **catch** (IOException e) {

}

**return** jsonText;

}

**public** **void** setDance(String jsonString) {

JSONObject json = **null**;

**try**{

json=(JSONObject) **new** JSONParser().parse(FileRead(jsonString));

}**catch**(Exception e2){

json = (JSONObject) **new** JSONParser().parse(jsonString);

}

// delay = ((Long) json.get("delay")).intValue();

begin=(JSONObject)json.get("set");

JSONArray array = (JSONArray) json.get("dance");

**for** (**int** i = 0; i < array.size(); i++) {

dance.add((JSONObject) array.get(i));

}

}

**public** **boolean** getDance() {

**return** dancing;

}

**public** **void** run() {

**while** (**true**) {

robot.setRobot(begin);

// System.out.println("t");

**int** size = dance.size();

**for** (**int** i = 0; i < size; i++) {

robot.dancing(dance.get(i));

// robot.setRobot(dance.get(i));

robot.repaint();

// try {

// Thread.sleep(delay);

// } catch (Exception e) {

// }

}

}

}

**public** **void** setLocation(**int** x,**int** y){

robot.setLocation(x, y);

}

**public** BasicRobot getComponent() {

**return** robot;

}

//@SuppressWarnings("deprecation")

**public** **void** startdancing() {

// **TODO** Auto-generated method stub

dancing = **true**;

**this**.resume();

}

//@SuppressWarnings("deprecation")

**public** **void** stopdancing() {

// **TODO** Auto-generated method stub

dancing = **false**;

**this**.suspend();

}

}

**Dancing Robot interface**

**package** sys;

**public** **interface** dancingrobots {

**void** startdancing();

**void** stopdancing();

}

**Main class**

**package** sys;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

frame f=**new** frame();

}

}

**Each character class**

package sys;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class Apeach extends BasicRobot {

public Apeach(String jsonString) {

super(jsonString);

// TODO Auto-generated constructor stub

}

protected void paintComponent(Graphics g) {

// TODO Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(new Color(254,254,254));

g2.fillRect(X, Y, width, height);

g2.setColor(new Color(0,0,0));

g2.drawRect(X, Y, width, height);

g2.setColor(new Color(254,254,254));

g2.setStroke(new BasicStroke(armSize));

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setColor(new Color(254,254,254));

g2.setStroke(new BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

// 발그리기

// g2.setColor(new Color(250,187,0));

// g2.fillOval(rightFoot.x-5, rightFoot.y+3, 15, 8);

// g2.fillOval(leftFoot.x-5, leftFoot.y+3, 15, 8);

g2.setColor(new Color(249,208,214));

g2.rotate(headangle, head.x, head.y);

//g2.fillOval(head.x-(int)(headWidth/1.9), head.y-headHeight+25, headWidth+5, headHeight);

// g2.drawRoundRect(head.x-(int)(headWidth/1.9), head.y-headHeight+25, headWidth+5, headHeight, 60, 60);

//머리형태그리기

g2.fillRoundRect(head.x-(int)(headWidth/1.9), head.y-headHeight+25, headWidth+5, headHeight-20, 60, 60);

int px[] = new int[3];

int py[] = new int[3];

px[0] = head.x-25;

px[1] = head.x;

px[2] = head.x+25;

py[0] = head.y-48;

py[1] = head.y-70;

py[2] = head.y-48;

g2.fillPolygon(px,py,3);

g2.setColor(new Color(0,0,0));

g2.setStroke(new BasicStroke(2));

//홍조그리기

g2.setColor(new Color(239,133,135));

g2.fillOval(head.x-27,head.y-23,17,7);

g2.fillOval(head.x+13,head.y-23,17,7);

//입술그리기

g2.setColor(new Color(43,27,27));

g2.drawArc(head.x-5, head.y-10, 10, 5, 235, 90);

//눈 흰자그리기

g2.setColor(new Color(248,250,247));

g2.fillArc(head.x-27, head.y-34, 20, 7, 0, 360);

g2.fillArc(head.x+10, head.y-34, 20, 7, 0, 360);

g2.setColor(new Color(43,27,27));

g2.setStroke(new BasicStroke(2));

//눈 테두리

g2.drawArc(head.x-27, head.y-34, 20, 7, 0, 360);

g2.drawArc(head.x+10, head.y-34, 20, 7, 0, 360);

//눈동자

g2.fillOval(head.x-20, head.y-32, 5, 5);

g2.fillOval(head.x+17, head.y-32, 5, 5);

g2.setStroke(new BasicStroke(2));

//눈썹은 춤모션에서 추가 하고 싶은경우에 추가할것 \_apeach에 한함.

// //왼쪽눈썹

// g2.drawLine(head.x-20, head.y-36, head.x-20, head.y-38);

// g2.drawLine(head.x-25, head.y-36, head.x-26, head.y-38);

// g2.drawLine(head.x-15, head.y-36, head.x-14, head.y-38);

// //오른쪽눈썹

// g2.drawLine(head.x+20, head.y-36, head.x+20, head.y-38);

// g2.drawLine(head.x+25, head.y-36, head.x+26, head.y-38);

// g2.drawLine(head.x+15, head.y-36, head.x+14, head.y-38);

}

}

package sys;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class ConClass extends BasicRobot {

public ConClass(String jsonString) {

super(jsonString);

// TODO Auto-generated constructor stub

}

protected void paintComponent(Graphics g) {

// TODO Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(new Color(46,153,59));

g2.fillRect(X, Y, width, height);

g2.setStroke(new BasicStroke(armSize));

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setStroke(new BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.rotate(headangle, head.x, head.y);

g2.fillOval(head.x-(int)(headWidth/1.7), head.y-headHeight, headWidth, headHeight);

g2.fillOval(head.x-headWidth,head.y-(headHeight/2),headWidth,headHeight/2);

g2.setStroke(new BasicStroke(0));

g2.setColor(Color.YELLOW);

g2.fillRoundRect(head.x-(int)(headWidth/3.4), head.y-(int)(headHeight/1.4), headWidth/3, headHeight/3, 100, 2);

g2.setColor(Color.BLACK);

g2.drawRoundRect(head.x-(int)(headWidth/3.4), head.y-(int)(headHeight/1.4), headWidth/3, headHeight/3, 100, 2);

g2.drawLine(head.x-(int)(headWidth/6.2),head.y-(int)(headHeight/1.4),head.x-(int)(headWidth/6.2),head.y-(int)(headHeight/2));

g2.drawLine(head.x-headWidth, head.y-(headHeight/4), head.x-(headWidth/2), head.y-(headHeight/4));

g2.setColor(Color.WHITE);

g2.drawLine(head.x-(int)(headWidth/1.5), head.y-(headHeight/4), head.x-(int)(headWidth/1.5), head.y-(headHeight/3));

}

}

package sys;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class Frodo extends BasicRobot {

public Frodo(String jsonString) {

super(jsonString);

// TODO Auto-generated constructor stub

}

protected void paintComponent(Graphics g) {

// TODO Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(new Color(181,136,51));

g2.fillRect(X, Y, width, height);

g2.setColor(new Color(0,0,0));

g2.drawRect(X, Y, width, height);

g2.setColor(new Color(181,136,51));

g2.setStroke(new BasicStroke(armSize));

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

//빨간 목걸이

g2.setColor(new Color(192,27,21));

g2.drawLine(head.x-20, head.y+5, head.x+20, head.y+5);

//회색 방울

g2.setColor(new Color(157,156,152));

g2.fillOval(head.x-5, head.y+10, 8, 8);

g2.setColor(new Color(181,136,51));

g2.setStroke(new BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setColor(new Color(181,136,51));

g2.rotate(headangle, head.x, head.y);

g2.fillRoundRect(head.x-(int)(headWidth/1.9), head.y-headHeight+5, headWidth, headHeight, 45, 45);

g2.fillRoundRect(head.x+8, head.y-headHeight, 15, 15, 40, 40);

g2.fillRoundRect(head.x-27, head.y-headHeight, 15, 15, 40, 40);

g2.setColor(new Color(0,0,0));

g2.setStroke(new BasicStroke(2));

// 입

g2.setColor(Color.white);

g2.fillArc(head.x-20, head.y-26,40,20,180,180);

g2.setColor(Color.black);

g2.drawArc(head.x-20, head.y-26,40,20,180,180);

g2.drawLine(head.x-20, head.y-16, head.x+20, head.y-16);

g2.drawLine(head.x-18, head.y-11, head.x+18, head.y-11);

g2.drawLine(head.x, head.y-16, head.x, head.y-6);

g2.drawLine(head.x-10,head.y-16,head.x-10,head.y-9);

g2.drawLine(head.x+10,head.y-16,head.x+10,head.y-9);

//g2.drawLine(head.x-5, head.y-2,head.x+5, head.y-2);

// 코

g2.fillOval(head.x-7, head.y-26, 11, 6);

// // 아래는 눈 테두리용

// g2.fillOval(head.x-19, head.y-31, 15, 6);

// g2.fillOval(head.x+5, head.y-31, 15, 6);

// // 아래는 눈 흰자용

g2.setColor(Color.white);

// g2.fillOval(head.x-19, head.y-30, 13, 4);

// g2.fillOval(head.x+5, head.y-30, 13, 4);

g2.fillArc(head.x-25, head.y-38, 18, 20, 0, 180);

g2.fillArc(head.x+5, head.y-38, 18, 20, 0, 180);

// // 아래는 눈 검은자위용

// g2.setColor(Color.black);

// g2.setStroke(new BasicStroke(3));

g2.setColor(Color.black);

g2.fillOval(head.x-18, head.y-35, 4, 4);

g2.fillOval(head.x+12, head.y-35, 4, 4);

// g2.drawLine(head.x-14, head.y-30, head.x-14, head.y-28);

// g2.drawLine(head.x+10, head.y-30, head.x+10, head.y-28);

}

}

package sys;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class JayG extends BasicRobot {

public JayG(String jsonString) {

super(jsonString);

// TODO Auto-generated constructor stub

}

protected void paintComponent(Graphics g) {

// TODO Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(new Color(103,83,29));

g2.fillOval(leftHand.x-(armSize/2)-(int)(Math.cos(leftArm2angle)\*armSize/2), leftHand.y-(armSize/2)-(int)(Math.sin(leftArm2angle)\*armSize/2), armSize, armSize);

g2.fillOval(rightHand.x-(armSize/2)+(int)(Math.cos(rightArm2angle)\*armSize/2), rightHand.y-(armSize/2)+(int)(Math.sin(rightArm2angle)\*armSize/2), armSize, armSize);

//

g2.setColor(Color.BLACK);

g2.fillRect(X, Y, width, height);

int[] shirtX={X+(width/2)-(width/6),X+(width/2),X+(width/2)+(width/6)};

int[] shirtY={Y,Y+(height/3),Y};

g2.setColor(Color.WHITE);

g2.fillPolygon(shirtX, shirtY, shirtX.length);

g2.setColor(Color.BLACK);

int[] neckTiX={X+(width/2)-(width/12),X+(width/2)-1,X+(width/2)-5,X+(width/2),X+(width/2)+5,X+(width/2)+1,X+(width/2)+(width/12)};

int[] neckTiY={Y,Y+(height/6)-1,Y+(height/3),Y+(height/2),Y+(height/3),Y+(height/6)-1,Y};

g2.fillPolygon(neckTiX, neckTiY, neckTiX.length);

g2.setStroke(new BasicStroke(2));

g2.drawLine(X+(width/2), Y, X+(width/2)-(width/6), Y+(height/4));

g2.drawLine(X+(width/2), Y, X+(width/2)+(width/6), Y+(height/4));

g2.setStroke(new BasicStroke(armSize));

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setColor(new Color(192,27,21));

// g2.drawLine(head.x-20, head.y+5, head.x+20, head.y+5);

// g2.setColor(new Color(157,156,152));

// g2.fillOval(head.x-5, head.y+10, 8, 8);

// g2.setColor(new Color(181,136,51));

g2.setColor(Color.BLACK);

g2.setStroke(new BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setStroke(new BasicStroke(0));

if (leftLeg2angle < 0) {

int leftshoesX[] = { leftFoot.x + 3, leftFoot.x + 3,

leftFoot.x , leftFoot.x , leftFoot.x - 8,

leftFoot.x - 3 };

int leftshoesY[] = { leftFoot.y + 1,

leftFoot.y + (int) (legSize / 1),

leftFoot.y + (int) (legSize / 1),

leftFoot.y + (int) (legSize / 1.8),

leftFoot.y + (int) (legSize / 1), leftFoot.y + 1 };

g2.fillPolygon(leftshoesX, leftshoesY, leftshoesX.length);

}

else {

int leftshoesX[] = { leftFoot.x - 5, leftFoot.x - 5,

leftFoot.x-2 , leftFoot.x-2 , leftFoot.x + 6,

leftFoot.x + 1 };

int leftshoesY[] = { leftFoot.y + 1,

leftFoot.y + (int) (legSize / 1),

leftFoot.y + (int) (legSize / 1),

leftFoot.y + (int) (legSize / 1.8),

leftFoot.y + (int) (legSize / 1), leftFoot.y + 1 };

g2.fillPolygon(leftshoesX, leftshoesY, leftshoesX.length);

}

if (rightLeg2angle < 0) {

int rightshoesX[] = { rightFoot.x + 3, rightFoot.x + 3,

rightFoot.x , rightFoot.x , rightFoot.x - 8,

rightFoot.x - 3 };

int rightshoesY[] = { rightFoot.y + 1,

rightFoot.y + (int) (legSize / 1),

rightFoot.y + (int) (legSize / 1),

rightFoot.y + (int) (legSize / 1.8),

rightFoot.y + (int) (legSize / 1), rightFoot.y + 1 };

g2.fillPolygon(rightshoesX, rightshoesY, rightshoesX.length);

}

else {

int rightshoesX[] = { rightFoot.x - 5, rightFoot.x - 5,

rightFoot.x-2 , rightFoot.x-2 , rightFoot.x + 6,

rightFoot.x + 1 };

int rightshoesY[] = { rightFoot.y + 1,

rightFoot.y + (int) (legSize / 1),

rightFoot.y + (int) (legSize / 1),

rightFoot.y + (int) (legSize / 1.8),

rightFoot.y + (int) (legSize / 1), rightFoot.y + 1 };

g2.fillPolygon(rightshoesX, rightshoesY, rightshoesX.length);

}

g2.rotate(headangle, head.x, head.y);

g2.setColor(new Color(254,213,1));

g2.fillOval(head.x-(int)(headWidth\*1.6)/2, head.y-(int)(headHeight\*1.6), (int)(headWidth\*1.6), (int)(headHeight\*1.6));

g2.setColor(new Color(103,83,29));

g2.fillOval(head.x-(headWidth/2), head.y-headHeight+5, headWidth, headHeight);

g2.setColor(Color.WHITE);

g2.fillOval(head.x - 9, head.y - (int) (headHeight / 3.5), 8, 6);

g2.fillOval(head.x, head.y - (int) (headHeight / 3.5), 8, 6);

g2.setColor(Color.BLACK);

g2.fillOval(head.x - 3, head.y - (int) (headHeight / 3.5) -2, 6, 6);

g2.setStroke(new BasicStroke(0));

g2.drawOval(head.x - 9, head.y - (int) (headHeight / 3.5), 8, 6);

g2.drawOval(head.x, head.y - (int) (headHeight / 3.5), 8, 6);

g2.fillRoundRect(head.x-25, head.y-(int)((headHeight)/1.7), 22, 17,5,5);

g2.fillRoundRect(head.x+3, head.y-(int)((headHeight)/1.7), 22, 17,5,5);

g2.setStroke(new BasicStroke(2));

g2.drawLine(head.x-10, head.y-(int)((headHeight)/1.7)+8, head.x+13, head.y-(int)((headHeight)/1.7)+8);

// g2.setColor(Color.WHITE);

// g2.fillArc(head.x-22, head.y-(int)(headHeight/2.5)-12, 18, 9, 210, 110);

// g2.fillArc(head.x-22, head.y-(int)(headHeight/2.5)-7, 18, 9, 30, 110);

//

// g2.fillArc(head.x+4, head.y-(int)(headHeight/2.5)-12, 18, 9, 210, 110);

// g2.fillArc(head.x+4, head.y-(int)(headHeight/2.5)-7, 18, 9, 30, 110);

//

////

// g2.setColor(Color.BLACK);

// g2.drawArc(head.x-22, head.y-(int)(headHeight/2.5)-12, 18, 9, 210, 110);

// g2.drawArc(head.x-22, head.y-(int)(headHeight/2.5)-7, 18, 9, 30, 110);

//

// g2.drawArc(head.x+4, head.y-(int)(headHeight/2.5)-12, 18, 9, 210, 110);

// g2.drawArc(head.x+4, head.y-(int)(headHeight/2.5)-7, 18, 9, 30, 110);

g2.setColor(new Color(254,213,1));

int num=28;

g2.rotate(Math.toRadians(-145),head.x,head.y-(int)(headHeight\*1.6)/2);

for(int i=0;i<num;i++){

g2.rotate(Math.toRadians(10),head.x,head.y-(int)(headHeight\*1.6)/2);

g2.fillOval(head.x-5,head.y-(int)(headHeight\*1.6)-5,10,10);

}

// g2.fillOval(head.x-(int)(headWidth\*1.6)/2, head.y-(int)(headHeight\*1.6), (int)(headWidth\*1.6), (int)(headHeight\*1.6));

}

}

**package** sys;

**import** java.awt.BasicStroke;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** myJSON.JSONObject;

**import** myJSON.JSONParser;

**public** **class** Muzi **extends** BasicRobot {

**private** **int** faceWidth = 30;

**private** **int** faceHeight = 30;

**private** point leftEar;

**private** point rightEar;

**private** **int** earWidth;

**private** **int** earHeight;

**private** **int** look=0;

**public** Muzi(String jsonString) {

**super**(jsonString);

// System.out.println(jsonString);

JSONObject json = **null**;

json = (JSONObject) **new** JSONParser().parse(jsonString);

**this**.jsonRead(json);

**this**.calcPoint();

}

**public** **void** jsonRead(JSONObject json){

**super**.jsonRead(json);

// System.out.println(json);

**try**{

faceWidth = ((Long) json.get("faceWidth")).intValue();

}**catch**(NullPointerException e){}

**try**{

faceHeight = ((Long) json.get("faceHeight")).intValue();

}**catch**(NullPointerException e){}

**try**{

earWidth = ((Long) json.get("earWidth")).intValue();

}**catch**(NullPointerException e){}

**try**{

earHeight = ((Long) json.get("earHeight")).intValue();

}**catch**(NullPointerException e){}

**try**{

look=((Long)json.get("look")).intValue();

}**catch**(NullPointerException e){}

}

**public** **void** calcPoint(){

**super**.calcPoint();

leftEar = **new** point(

(**int**) (head.x - ((headWidth / 2) - (earWidth \* 1.3))), head.y

- headHeight);

rightEar = **new** point(

(**int**) (head.x + ((headWidth / 2) - (earWidth \* 1.3))), head.y

- headHeight);

}

**public** **void** setRobot(String jsonString) {

JSONObject json = **null**;

json = (JSONObject) **new** JSONParser().parse(jsonString);

setRobot(json);

}

**public** **void** setRobot(JSONObject json) {

**super**.setRobot(json);

jsonRead(json);

calcPoint();

}

**protected** **void** paintComponent(Graphics g) {

// **TODO** Auto-generated method stub

Graphics2D g2 = (Graphics2D) g;

g2.setColor(Color.BLACK);

g2.setStroke(**new** BasicStroke(legSize+2));

g2.fillOval(leftLeg.x - (legSize / 2), leftLeg.y - (legSize / 2),

legSize, legSize);

g2.drawLine(leftLeg.x, leftLeg.y , leftKnee.x,

leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.fillOval(rightLeg.x - (legSize / 2), rightLeg.y - (legSize / 2),

legSize, legSize);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x,

rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setStroke(**new** BasicStroke(legSize));

g2.setColor(Color.YELLOW);

g2.fillOval(leftLeg.x - (legSize / 2), leftLeg.y - (legSize / 2),

legSize, legSize);

g2.drawLine(leftLeg.x, leftLeg.y , leftKnee.x,

leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.fillOval(rightLeg.x - (legSize / 2), rightLeg.y - (legSize / 2),

legSize, legSize);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x,

rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setColor(Color.WHITE);

g2.fillRoundRect(X, Y, width, height, 10, 10);

g2.setColor(Color.BLACK);

g2.setStroke(**new** BasicStroke(0));

g2.drawRoundRect(X, Y, width, height, 10, 10);

g2.setColor(Color.YELLOW);

// System.out.println(leftShoulder.x+" "+leftShoulder.y);

// System.out.println(leftElbow.y+" "+leftElbow.y);

g2.fillOval(leftShoulder.x - (armSize / 2), leftShoulder.y

- (armSize / 2), armSize, armSize);

g2.setColor(Color.BLACK);

g2.drawOval(leftShoulder.x - (armSize / 2), leftShoulder.y

- (armSize / 2), armSize, armSize);

g2.fillOval(rightShoulder.x - (armSize / 2), rightShoulder.y

- (armSize / 2), armSize, armSize);

g2.setColor(Color.BLACK);

g2.setStroke(**new** BasicStroke(armSize+2));

g2.drawLine(leftShoulder.x , leftShoulder.y,

leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y,

rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setStroke(**new** BasicStroke(armSize));

g2.setColor(Color.YELLOW);

g2.drawLine(leftShoulder.x , leftShoulder.y,

leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.setStroke(**new** BasicStroke(armSize));

g2.setColor(Color.YELLOW);

g2.fillOval(rightShoulder.x - (armSize / 2), rightShoulder.y

- (armSize / 2), armSize, armSize);

g2.drawLine(rightShoulder.x, rightShoulder.y,

rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setStroke(**new** BasicStroke(legSize));

// g2.drawLine(neck.x,neck.y,head.x,head.y);

g2.rotate(headangle, head.x, head.y);

g2.setColor(Color.WHITE);

g2.fillOval(head.x - (headWidth / 2), head.y - (headHeight),

headWidth, headHeight + 10);

g2.setColor(Color.YELLOW);

g2.fillOval(head.x - (faceWidth / 2), head.y - (faceHeight) + 2,

faceWidth, faceHeight);

// g2.setStroke(new BasicStroke(0));

// g2.setColor(Color.BLACK);

// g2.drawOval(head.x - (headWidth / 2), head.y - (headHeight),

// headWidth, headHeight + 10);

// g2.drawOval(head.x - (faceWidth / 2), head.y - (faceHeight) + 2,

// faceWidth, faceHeight);

g2.setColor(Color.WHITE);

g2.fillRoundRect(leftEar.x - (earWidth / 2), leftEar.y - earHeight + 10,

earWidth, earHeight, 20, 10);

g2.fillRoundRect(rightEar.x - (earWidth / 2), rightEar.y - earHeight

+ 10, earWidth, earHeight, 20, 10);

g2.setColor(Color.PINK);

g2.fillRoundRect(leftEar.x - (**int**) (earWidth / 2.6), leftEar.y

- earHeight + 15, (**int**) (earWidth \* 0.7),

(**int**) (earHeight \* 0.7), 20, 10);

g2.fillRoundRect(rightEar.x - (**int**) (earWidth / 2.6), rightEar.y

- earHeight + 15, (**int**) (earWidth \* 0.7),

(**int**) (earHeight \* 0.7), 20, 10);

g2.setColor(Color.BLACK);

// g2.drawRoundRect(leftEar.x - (earWidth / 2), leftEar.y - earHeight + 5,

// earWidth, earHeight, 20, 10);

// g2.drawRoundRect(rightEar.x - (earWidth / 2), rightEar.y - earHeight

// + 5, earWidth, earHeight, 20, 10);

// g2.drawRoundRect(leftEar.x - (int) (earWidth / 2.6), leftEar.y

// - earHeight + 10, (int) (earWidth \* 0.7),

// (int) (earHeight \* 0.7), 20, 10);

// g2.drawRoundRect(rightEar.x - (int) (earWidth / 2.6), rightEar.y

// - earHeight + 10, (int) (earWidth \* 0.7),

// (int) (earHeight \* 0.7), 20, 10);

**if** (look == 0) {

g2.setColor(Color.BLACK);

g2.fillOval(head.x - 3, head.y - (**int**) (headHeight / 2.5) +1, 6, 6);

g2.setStroke(**new** BasicStroke(0));

g2.drawArc(head.x - 9, head.y - (**int**) (headHeight / 2.5)+3, 8, 8,

180, 180);

g2.drawArc(head.x, head.y - (**int**) (headHeight / 2.5)+3, 8, 8, 180,

180);

g2.setColor(Color.WHITE);

g2.fillOval(head.x - 20, head.y - (**int**) (headHeight / 2.5) - 10, 16,

12);

g2.fillOval(head.x + 4, head.y - (**int**) (headHeight / 2.5) - 10, 16, 12);

g2.setColor(Color.BLACK);

g2.drawOval(head.x - 20, head.y - (**int**) (headHeight / 2.5) - 10, 16,

12);

g2.drawOval(head.x + 4, head.y - (**int**) (headHeight / 2.5) - 10, 16, 12);

g2.fillOval(head.x - 13, head.y - (**int**) (headHeight / 2.5) - 5, 3, 4);

g2.fillOval(head.x + 11, head.y - (**int**) (headHeight / 2.5) - 5, 3, 4);

}

**else** **if**(look == 1){

g2.setColor(Color.BLACK);

g2.fillOval(head.x - 3, head.y - (**int**) (headHeight / 2.5) +1, 6, 6);

g2.setStroke(**new** BasicStroke(0));

g2.drawArc(head.x - 9, head.y - (**int**) (headHeight / 2.5)+3, 8, 8,

180, 180);

g2.drawArc(head.x, head.y - (**int**) (headHeight / 2.5)+3, 8, 8, 180,

180);

g2.drawLine(head.x - 5, head.y - (**int**) (headHeight / 2.1), head.x-17, head.y - (**int**) (headHeight / 2.1));

g2.drawLine(head.x - 5, head.y - (**int**) (headHeight / 2.1), head.x-17, head.y - (**int**) (headHeight / 1.8));

g2.drawLine(head.x - 5, head.y - (**int**) (headHeight / 2.1), head.x-17, head.y - (**int**) (headHeight / 2.6));

g2.drawLine(head.x + 5, head.y - (**int**) (headHeight / 2.1), head.x+17, head.y - (**int**) (headHeight / 2.1));

g2.drawLine(head.x + 5, head.y - (**int**) (headHeight / 2.1), head.x+17, head.y - (**int**) (headHeight / 1.8));

g2.drawLine(head.x + 5, head.y - (**int**) (headHeight / 2.1), head.x+17, head.y - (**int**) (headHeight / 2.6));

// g2.setColor(Color.BLACK);

// g2.drawOval(head.x - 10, head.y - (int) (headHeight / 2.5) - 7, 6,

// 4);

// g2.drawOval(head.x + 4, head.y - (int) (headHeight / 2.5) - 7, 6, 4);

//

// g2.drawOval(head.x - 8, head.y - (int) (headHeight / 2.5) - 6, 1, 1);

// g2.drawOval(head.x + 6, head.y - (int) (headHeight / 2.5) - 6, 1, 1);

}

}

**public** **void** dancing(JSONObject json) {

// **TODO** Auto-generated method stub

**int** look=0;

**try**{

look=((Long)json.get("look")).intValue();

}**catch**(NullPointerException e){}

**this**.look=look;

**super**.dancing(json);

}

}

**package** sys;

**import** java.awt.BasicStroke;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** java.text.ParseException;

**import** myJSON.JSONObject;

**import** myJSON.JSONParser;

**public** **class** Neo **extends** BasicRobot {

//

// private int headWidth = 50;

// private int headHeight = 40;

**private** **int** faceWidth = 30;

**private** **int** faceHeight = 30;

**private** point leftEar;

**private** point rightEar;

**private** **int** earWidth;

**private** **int** earHeight;

**public** Neo(String jsonString) {

**super**(jsonString);

// System.out.println(jsonString);

JSONObject json = **null**;

json = (JSONObject) **new** JSONParser().parse(jsonString);

**this**.jsonRead(json);

**this**.calcPoint();

}

**public** **void** jsonRead(JSONObject json){

**super**.jsonRead(json);

// System.out.println(json);

**try**{

earWidth = ((Long) json.get("earWidth")).intValue();

}**catch**(NullPointerException e){}

**try**{

earHeight = ((Long) json.get("earHeight")).intValue();

}**catch**(NullPointerException e){}

}

**public** **void** calcPoint(){

**super**.calcPoint();

leftEar = **new** point(

(**int**) (head.x - ((headWidth / 2) - (earWidth \* 1.3))), head.y

- headHeight);

rightEar = **new** point(

(**int**) (head.x + ((headWidth / 2) - (earWidth \* 1.3))), head.y

- headHeight);

}

**public** **void** setRobot(String jsonString) {

JSONObject json = **null**;

json = (JSONObject) **new** JSONParser().parse(jsonString);

setRobot(json);

}

**public** **void** setRobot(JSONObject json) {

**super**.setRobot(json);

jsonRead(json);

calcPoint();

}

**protected** **void** paintComponent(Graphics g) {

// **TODO** Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(**new** Color(116,150,187));

g2.fillRoundRect(X, Y, width, height, 5, 5);

g2.setStroke(**new** BasicStroke(armSize));

g2.fillOval(leftShoulder.x - (armSize / 2), leftShoulder.y

- (armSize / 2), armSize, armSize);

g2.drawLine(leftShoulder.x , leftShoulder.y,

leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.fillOval(rightShoulder.x - (armSize / 2), rightShoulder.y

- (armSize / 2), armSize, armSize);

g2.drawLine(rightShoulder.x , rightShoulder.y,

rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

g2.setStroke(**new** BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x,

leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x,

rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setColor(Color.WHITE);

g2.fillRoundRect(X+3, Y+10, width-6, height-15, 30, 40);

g2.fillOval(leftHand.x-(armSize), leftHand.y-(armSize/2), armSize, armSize);

g2.fillOval(rightHand.x, rightHand.y-(armSize/2), armSize, armSize);

g2.fillOval(leftFoot.x-(legSize/2), leftFoot.y, legSize, legSize);

g2.fillOval(rightFoot.x-(legSize/2), rightFoot.y, legSize, legSize);

g2.rotate(headangle, head.x, head.y);

g2.setColor(**new** Color(116,150,187));

g2.fillRoundRect(head.x - (headWidth / 2), head.y - (headHeight),

headWidth, headHeight + 10, 50, 100);

g2.setColor(Color.BLACK);

g2.fillArc(head.x- (**int**)(headWidth\*0.6), head.y-(**int**)(headHeight\*1.2), (**int**)(headWidth\*1.2), (**int**)(headHeight\*1.2), 0, 180);

g2.fillRect(head.x- (**int**)(headWidth\*0.6), head.y-(**int**)(headHeight\*0.6), (**int**)(headWidth\*0.3), (**int**)(headHeight\*0.4));

g2.fillRect(head.x+ (**int**)(headWidth\*0.3), head.y-(**int**)(headHeight\*0.6), (**int**)(headWidth\*0.3), (**int**)(headHeight\*0.4));

g2.setColor(**new** Color(70,70,70));

g2.setStroke(**new** BasicStroke(2));

**int**[] leftNX={head.x- (**int**)(headWidth\*0.5),head.x- (**int**)(headWidth\*0.5),head.x- (**int**)(headWidth\*0.4),head.x- (**int**)(headWidth\*0.4)};

**int**[] leftNY={head.y-(**int**)(headHeight\*0.4),head.y-(**int**)(headHeight\*0.5),head.y-(**int**)(headHeight\*0.4),head.y-(**int**)(headHeight\*0.5)};

**int**[] rightNX={head.x+ (**int**)(headWidth\*0.4),head.x+ (**int**)(headWidth\*0.4),head.x+ (**int**)(headWidth\*0.5),head.x+ (**int**)(headWidth\*0.5)};

**int**[] rightNY={head.y-(**int**)(headHeight\*0.4),head.y-(**int**)(headHeight\*0.5),head.y-(**int**)(headHeight\*0.4),head.y-(**int**)(headHeight\*0.5)};

**int**[] mtX={head.x- (**int**)(headWidth\*0.3),head.x- (**int**)(headWidth\*0.246),head.x- (**int**)(headWidth\*0.192),head.x- (**int**)(headWidth\*0.138),head.x- (**int**)(headWidth\*0.084),head.x- (**int**)(headWidth\*0.03),head.x+ (**int**)(headWidth\*0.024),head.x+ (**int**)(headWidth\*0.078),head.x+ (**int**)(headWidth\*0.132),head.x+ (**int**)(headWidth\*0.186),head.x+ (**int**)(headWidth\*0.24),head.x+ (**int**)(headWidth\*0.3)};

**int**[] mtY={head.y-(**int**)(headHeight\*0.75),head.y-(**int**)(headHeight\*0.8),head.y-(**int**)(headHeight\*0.75),head.y-(**int**)(headHeight\*0.85),head.y-(**int**)(headHeight\*0.75),head.y-(**int**)(headHeight\*0.9),head.y-(**int**)(headHeight\*0.75),head.y-(**int**)(headHeight\*0.85),head.y-(**int**)(headHeight\*0.75),head.y-(**int**)(headHeight\*0.8),head.y-(**int**)(headHeight\*0.75)};

g2.drawPolyline(leftNX, leftNY, leftNX.length);

g2.drawPolyline(rightNX, rightNY, rightNX.length);

g2.drawPolyline(mtX, mtY, mtY.length);

g2.setColor(**new** Color(116,150,187));

**int**[] leftEarPointx={leftEar.x-9,leftEar.x-8,leftEar.x+5};

**int**[] leftEarPointy={leftEar.y+3,leftEar.y-11,leftEar.y-3};

g2.fillPolygon(leftEarPointx, leftEarPointy, 3);

**int**[] rightEarPointx={rightEar.x+9,rightEar.x+8,rightEar.x-5};

**int**[] rightEarPointy={rightEar.y+3,rightEar.y-11,rightEar.y-3};

g2.fillPolygon(rightEarPointx, rightEarPointy, 3);

g2.setColor(**new** Color(200,145,152));

**int**[] leftEar2Pointx={leftEar.x-5,leftEar.x-5,leftEar.x+3};

**int**[] leftEar2Pointy={leftEar.y+1,leftEar.y-7,leftEar.y-2};

g2.fillPolygon(leftEar2Pointx, leftEar2Pointy, 3);

**int**[] rightEar2Pointx={rightEar.x+6,rightEar.x+6,rightEar.x-2};

**int**[] rightEar2Pointy={rightEar.y+1,rightEar.y-7,rightEar.y-2};

g2.fillPolygon(rightEar2Pointx, rightEar2Pointy, 3);

// g2.setColor(Color.WHITE);

// g2.fillRoundRect(leftEar.x - (earWidth / 2), leftEar.y - earHeight + 5,

// earWidth, earHeight, 20, 10);

// g2.fillRoundRect(rightEar.x - (earWidth / 2), rightEar.y - earHeight

// + 5, earWidth, earHeight, 20, 10);

//

// g2.setColor(Color.PINK);

// g2.fillRoundRect(leftEar.x - (int) (earWidth / 2.6), leftEar.y

// - earHeight + 10, (int) (earWidth \* 0.7),

// (int) (earHeight \* 0.7), 20, 10);

// g2.fillRoundRect(rightEar.x - (int) (earWidth / 2.6), rightEar.y

// - earHeight + 10, (int) (earWidth \* 0.7),

// (int) (earHeight \* 0.7), 20, 10);

g2.setColor(Color.WHITE);

g2.fillOval(head.x - 9, head.y - (**int**) (headHeight / 2.5), 8, 6);

g2.fillOval(head.x, head.y - (**int**) (headHeight / 2.5), 8, 6);

g2.setColor(Color.BLACK);

g2.fillOval(head.x - 3, head.y - (**int**) (headHeight / 2.5) -2, 6, 6);

g2.setStroke(**new** BasicStroke(0));

g2.drawOval(head.x - 9, head.y - (**int**) (headHeight / 2.5), 8, 6);

g2.drawOval(head.x, head.y - (**int**) (headHeight / 2.5), 8, 6);

g2.setColor(Color.WHITE);

g2.fillArc(head.x-22, head.y-(**int**)(headHeight/2.5)-12, 18, 9, 210, 110);

g2.fillArc(head.x-22, head.y-(**int**)(headHeight/2.5)-7, 18, 9, 30, 110);

g2.fillArc(head.x+4, head.y-(**int**)(headHeight/2.5)-12, 18, 9, 210, 110);

g2.fillArc(head.x+4, head.y-(**int**)(headHeight/2.5)-7, 18, 9, 30, 110);

//

g2.setColor(Color.BLACK);

g2.drawArc(head.x-22, head.y-(**int**)(headHeight/2.5)-12, 18, 9, 210, 110);

g2.drawArc(head.x-22, head.y-(**int**)(headHeight/2.5)-7, 18, 9, 30, 110);

g2.drawArc(head.x+4, head.y-(**int**)(headHeight/2.5)-12, 18, 9, 210, 110);

g2.drawArc(head.x+4, head.y-(**int**)(headHeight/2.5)-7, 18, 9, 30, 110);

g2.setStroke(**new** BasicStroke(1));

g2.drawLine(head.x - 13, head.y - (**int**) (headHeight / 2.5) - 6,

head.x - 13, head.y - (**int**) (headHeight / 2.5) - 10);

g2.drawLine(head.x - 10, head.y - (**int**) (headHeight / 2.5) - 7,

head.x - 7, head.y - (**int**) (headHeight / 2.5) - 10);

g2.drawLine(head.x - 16, head.y - (**int**) (headHeight / 2.5) - 6,

head.x - 19, head.y - (**int**) (headHeight / 2.5) - 10);

g2.drawLine(head.x + 13, head.y - (**int**) (headHeight / 2.5) - 6,

head.x + 13, head.y - (**int**) (headHeight / 2.5) - 10);

g2.drawLine(head.x + 10, head.y - (**int**) (headHeight / 2.5) - 7,

head.x + 7, head.y - (**int**) (headHeight / 2.5) - 10);

g2.drawLine(head.x + 16, head.y - (**int**) (headHeight / 2.5) - 6,

head.x + 19, head.y - (**int**) (headHeight / 2.5) - 10);

g2.setStroke(**new** BasicStroke(2));

g2.drawLine(head.x - 13, head.y - (**int**) (headHeight / 2.5) - 6, head.x - 13, head.y - (**int**) (headHeight / 2.5) - 3);

g2.drawLine(head.x + 13, head.y - (**int**) (headHeight / 2.5) - 6, head.x + 13, head.y - (**int**) (headHeight / 2.5) - 3);

// g2.drawOval(head.x - 14, head.y - (int) (headHeight / 2.5) - 7, 1, 2);

// g2.drawOval(head.x + 13, head.y - (int) (headHeight / 2.5) - 7, 1, 2);

}

}

package sys;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

public class Tube extends BasicRobot {

public Tube(String jsonString) {

super(jsonString);

// TODO Auto-generated constructor stub

}

protected void paintComponent(Graphics g) {

// TODO Auto-generated method stub

Graphics2D g2=(Graphics2D)g;

g2.setColor(new Color(254,254,254));

g2.fillRect(X, Y, width, height);

g2.setColor(new Color(0,0,0));

g2.drawRect(X, Y, width, height);

g2.setColor(new Color(254,254,254));

g2.setStroke(new BasicStroke(armSize));

g2.drawLine(leftShoulder.x, leftShoulder.y, leftElbow.x, leftElbow.y);

g2.drawLine(leftElbow.x, leftElbow.y, leftHand.x, leftHand.y);

g2.drawLine(rightShoulder.x, rightShoulder.y, rightElbow.x, rightElbow.y);

g2.drawLine(rightElbow.x, rightElbow.y, rightHand.x, rightHand.y);

// g2.setColor(new Color(192,27,21));

// g2.drawLine(head.x-20, head.y+5, head.x+20, head.y+5);

// g2.setColor(new Color(157,156,152));

// g2.fillOval(head.x-5, head.y+10, 8, 8);

g2.setColor(new Color(254,254,254));

g2.setStroke(new BasicStroke(legSize));

g2.drawLine(leftLeg.x, leftLeg.y, leftKnee.x, leftKnee.y);

g2.drawLine(leftKnee.x, leftKnee.y, leftFoot.x, leftFoot.y);

g2.drawLine(rightLeg.x, rightLeg.y, rightKnee.x, rightKnee.y);

g2.drawLine(rightKnee.x, rightKnee.y, rightFoot.x, rightFoot.y);

g2.setColor(new Color(254,254,254));

g2.rotate(headangle, head.x, head.y);

g2.fillOval(head.x-(int)(headWidth/1.9), head.y-headHeight+20, headWidth+10, headHeight);

// g2.fillOval(head.x+5, head.y-headHeight, 15, 15);

// g2.fillOval(head.x-(int)(headWidth/2.5), head.y-headHeight, 15, 15);

g2.setColor(new Color(0,0,0));

g2.setStroke(new BasicStroke(2));

// // 입

// g2.drawLine(head.x-5, head.y-2,head.x+5, head.y-2);

// // 코

// g2.fillOval(head.x-7, head.y-20, 11, 6);

// // 아래는 눈 테두리용

// g2.fillOval(head.x-19, head.y-31, 15, 6);

// g2.fillOval(head.x+5, head.y-31, 15, 6);

// // 아래는 눈 흰자용

// g2.setColor(Color.white);

// g2.fillOval(head.x-19, head.y-30, 13, 4);

// g2.fillOval(head.x+5, head.y-30, 13, 4);

// 아래는 눈 검은자위용

//코 부분 노란색용

g2.setColor(new Color(250,187,0));

g2.setStroke(new BasicStroke(10));

g2.fillOval(head.x-8, head.y-22, 25, 20);

g2.fillRoundRect(head.x-40, head.y-15, 80, 30, 35, 35);

g2.setColor(Color.black);

//입술부분

g2.fillOval(head.x-6, head.y-18, 5, 5);

g2.fillOval(head.x+6, head.y-18, 5, 5);

g2.setStroke(new BasicStroke(2));

g2.drawArc(head.x-35, head.y, 60, 5, 180, 180);

//눈 부분

g2.fillOval(head.x-20, head.y-25, 6, 6);

g2.fillOval(head.x+18, head.y-25, 6, 6);

g2.setStroke(new BasicStroke(5));

//눈썹 부분

g2.drawArc(head.x-20, head.y-37, 10, 10, 45, 90);

g2.drawArc(head.x+18, head.y-37, 10, 10, 45, 90);

//g2.fillOval(head.x-headWidth,head.y-(headHeight/2),headWidth,headHeight/2);

}

}

**Json Package**

**package** myJSON;

**import** java.util.HashMap;

**public** **class** JSONObject **extends** HashMap<String,Object>{

}

**package** myJSON;

**import** java.util.ArrayList;

**public** **class** JSONArray **extends** ArrayList<Object> {

}

**package** myJSON;

**import** java.util.Stack;

**public** **class** JSONParser {

**private** **static** String *jsonString*;

**private** **static** **int** *i*;

**public** **static** JSONObject parse(String jsonText){

Stack<JSONObject> stack=**new** Stack<JSONObject>();

*jsonString*=jsonText.replaceAll("[ \n\r\t]","");

// System.out.println(jsonString);

*i*=0;

**return** *parseJsonObject*();

}

**private** **static** JSONObject parseJsonObject(){

JSONObject jsonObject=**new** JSONObject();

*i*=*jsonString*.indexOf("{",*i*)+1;

**while**(*jsonString*.charAt(*i*)!='}'){

String key;

Object value;

*i*=*jsonString*.indexOf("\"",*i*)+1;

key=*jsonString*.substring(*i*, *jsonString*.indexOf("\"",*i*));

*i*=*jsonString*.indexOf(":",*i*)+1;

value=*parseObject*();

// System.out.println(key+" "+value);

jsonObject.put(key, value);

}

*i*++;

**return** jsonObject;

}

**private** **static** JSONArray parseJsonList(){

JSONArray array=**new** JSONArray();

*i*=*jsonString*.indexOf("[",*i*)+1;

**while**(*jsonString*.charAt(*i*)!=']'){

Object value;

value=*parseObject*();

**int** index=0x7fffffff;

**int** index1=*jsonString*.indexOf(",",*i*)+1;

**int** index2=*jsonString*.indexOf("]",*i*);

**if**(index>index1 && index1!=0){

index=index1;

}

**if**(index>index2 && index2!=-1){

index=index2;

}

*i*=index;

array.add(value);

}

*i*++;

**return** array;

}

**private** **static** Object parseObject(){

Object value=**null**;

**if**(*jsonString*.charAt(*i*)=='{'){

value=*parseJsonObject*();

}

**else** **if**(*jsonString*.charAt(*i*)=='['){

value=*parseJsonList*();

}

**else** **if**(*jsonString*.charAt(*i*)=='"'){

*i*=*jsonString*.indexOf("\"",*i*)+1;

value=*jsonString*.substring(*i*, *jsonString*.indexOf("\"",*i*));

*i*=*jsonString*.indexOf("\"",*i*)+1;

}

**else**{

**int** index=0x7fffffff;

**int** index1=*jsonString*.indexOf(",",*i*);

**int** index2=*jsonString*.indexOf("}",*i*);

**int** index3=*jsonString*.indexOf("]",*i*);

**if**(index>index1 && index1!=-1){

index=index1;

}

**if**(index>index2 && index2!=-1){

index=index2;

}

**if**(index>index3 && index3!=-1){

index=index3;

}

value=*jsonString*.substring(*i*, index);

*i*=index;

**try**{

value=Long.*parseLong*((String)value);

}**catch**(NumberFormatException e){

value=Double.*parseDouble*((String)value);

}

}

**return** value;

}

}

1. Sample ScreenShot.



This is the sample screenshot of the program.

I will explain leftside to rightside.

First, We make 2 JComboBox. Upside box can select music. Downside box can select dance. Each character has own dance. If you do not choose dance, each characters dancing their own dance. Next 7 button is JRadioButton We just put image in that. If you check one button and clik the frame Then character will appear in the frame. Last three button is option button. Play button’s function is start dancing Pause button’s function is stop dancing. And trash can button’s function is clear the frame.