**Project 7 – Logo Interactors**

**(10 points)**

**Name: \_\_\_\_Ozaner Hansha\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Due Date: Monday, January 11, 2016**

**Description:**

In this short project, we will experiment with a few of the Java Swing Interactors such as **JButton**, **JSlider** and **JRadioButtons**. These classes provide “widgets” for common facilities for builing GUIs. The ACM Graphics library also provides a simple border layout that makes it easy to structure the components of a GUI into a center canvas region, and regions along the borders known as “North”, “South”, East”, and “West”.

In this project, we will use the “Target logo” example that I gave you earlier in an animation with controls to reset the animation, to control its speed, and to set the size of the bouncing logo. Be sure to include Javadoc and write clean, commented, nicely formatted code.

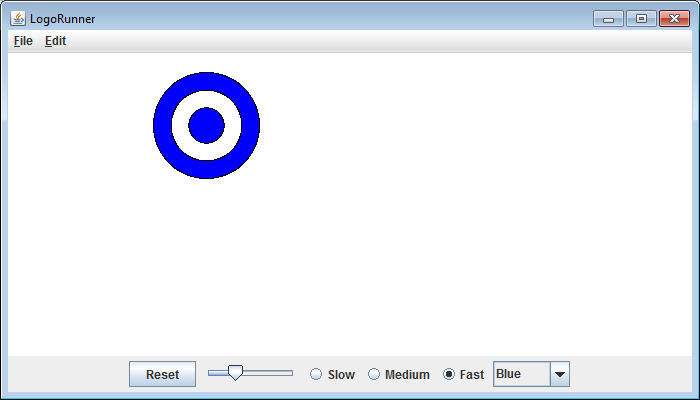
Carry out the following steps:

1. Download the archived project APCS2015Proj07LogoInteractors.
2. Do **not** modify the GTargetLogo class. Run LogoRunner to see what it does. Click the mouse to see what it does. Note how the run() method, advanceOneTimeStep() method and the mouse handler is coded.
3. In the init method of the LogoRunner class, add a JButton “Reset”. Look at the DrawStarMap\_ClearButton class in the APCS2015ExamplesACMSwingInteractors project for an example of using a JButton. Add code in the ActionPerformed method as required.
4. In the init method of the LogoRunner class, add a JSlider interactor to control the size of the logo. The new logo size will take effect after the next reset. Look at the DrawStarMap\_SizeSlider class in the APCS2015ExamplesACMSwingInteractors project for an example of using a JSlider. Modify the getCurrentPercentage method as required.
5. In the init method of the LogoRunner class, add a set of three JRadioButton interactors to control the speed of the animation. Since the speed is taken into account on each time step of the animation, it will immediately have an effect. Look at the DrawStarMap\_RadioButtons class in the APCS2015ExamplesACMSwingInteractors project for an example of using a JRadioButton. Modify the getCurrentSpeed method as required.

To complete the project,

* submit this Word document to your shared Google Drive folder
* fill in your name above
* include the source listing for the LogoRunner class
* a screen shot of the app after using the interactors
* Extensions: Add a JComboBox to change the color of the logo. You will need to adapt the GTargetLogo class to permit the color change.

***Screenshot:***



***LogoRunner Class:***

**package** unit6.logoProject;

**import** java.awt.Color;

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

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

**import** java.util.HashMap;

**import** javax.swing.ButtonGroup;

**import** javax.swing.JButton;

**import** javax.swing.JComboBox;

**import** javax.swing.JRadioButton;

**import** javax.swing.JSlider;

**import** acm.program.GraphicsProgram;

/\*\*

\* Demonstrates animation and Swing interactors such as JButtons, JSliders

\* and JRadioButtons.<br><br>

\*

\* Ozaner Hansha<br>

\* AP Computer Science<br>

\* **@author** Dr. Mark Jones

\* January 11th, 2016

\*/

@SuppressWarnings({"serial", "unchecked", "rawtypes"})

**public** **class** LogoRunner **extends** GraphicsProgram {

// application constants

**private** **static** **final** **int** ***INITIAL\_WIDTH*** = 700;

**private** **static** **final** **int** ***INITIAL\_HEIGHT*** = 400;

**private** **static** **final** **long** ***PAUSE\_TIME*** = 10; /\* animation pause time \*/

// logo sizes as percentages of the screen size

**private** **static** **final** **int** ***MIN\_PERCENTAGE*** = 10;

**private** **static** **final** **int** ***MAX\_PERCENTAGE*** = 90;

**private** **static** **final** **int** ***DEFAULT\_PERCENTAGE*** = 50;

// animation speed factors

**private** **static** **final** **double** ***SLOW\_SPEED*** = 1;

**private** **static** **final** **double** ***MEDIUM\_SPEED*** = 2;

**private** **static** **final** **double** ***FAST\_SPEED*** = 5;

// instance variables

**private** GTargetLogo logo; /\* the logo object \*/

**private** **double** dx; /\* velocity delta in the x direction \*/

**private** **double** dy; /\* velocity delta in the y direction \*/

**private** **boolean** suspend; /\* whether to suspend the animation \*/

//A map of the name of colors and the actual colors.

**private** **static** HashMap<String, Color> *colors* = **new** HashMap<String, Color>();

**static**

{

*colors*.put("Red", Color.***RED***);

*colors*.put("Blue", Color.***BLUE***);

*colors*.put("Cyan", Color.***CYAN***);

*colors*.put("Gray", Color.***GRAY***);

*colors*.put("Green", Color.***GREEN***);

*colors*.put("Magenta", Color.***MAGENTA***);

*colors*.put("Yellow", Color.***YELLOW***);

}

//Swing objects.

**private** JButton reset = **new** JButton("Reset");

**private** JSlider size = **new** JSlider(***MIN\_PERCENTAGE***,***MAX\_PERCENTAGE***,***DEFAULT\_PERCENTAGE***);

**private** JComboBox colorBox = **new** JComboBox(*colors*.keySet().toArray());

**private** JRadioButton[] speeds = {**new** JRadioButton("Slow"),

**new** JRadioButton("Medium"),

**new** JRadioButton("Fast")};

**private** ButtonGroup speedGroup = **new** ButtonGroup();

/\*\*

\* Animates a Target logo.

\* **@param** args none expected

\*/

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

**new** LogoRunner().start(args);

}

/\*\* Makes a bouncing Target logo and the GUI to control it. \*/

**public** **void** init() {

setSize(***INITIAL\_WIDTH***, ***INITIAL\_HEIGHT***);

// add a JButton to reset the animation starting at the center

// at the current size and speed

add(reset, ***SOUTH***);

// add a JSlider for the size of the logo

add(size, ***SOUTH***);

// add radio buttons for the speed of the animation

// (you control the speed by varying the pause interval in the animation)

**for**(JRadioButton b: speeds)

{

speedGroup.add(b);

add(b, ***SOUTH***);

}

speeds[1].setSelected(**true**);

//JComboBox for color changing.

add(colorBox, ***SOUTH***);

colorBox.addActionListener(**this**);

addActionListeners();

addMouseListeners();

addKeyListeners();

reset();

}

/\*\*

\* A space press pauses/resmums the animation

\* **@param** e a key event

\*/

**public** **void** keyPressed(KeyEvent e){

**if**(e.getKeyCode() == e.***VK\_SPACE***)

suspend = !suspend;

}

/\*\*

\* A mouse click suspends/resumes the animation.

\* **@param** e a mouse event

\*/

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

suspend = !suspend;

}

/\*\*

\* ActionEvent handler for button presses, etc.

\*/

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

**switch**(e.getActionCommand()){

**case** "comboBoxChanged":

logo.setColor(*colors*.get(((JComboBox)e.getSource()).getSelectedItem()));

**break**;

**case** "Reset": reset();

}

}

/\*\*

\* Resets the animation to start at the center at the current size and speed.

\*/

**public** **void** reset() {

removeAll();

**double** size = getCurrentPercentage() \* Math.*min*(getWidth(), getHeight());

logo = **new** GTargetLogo(getWidth() / 2., getHeight() / 2., size);

System.***out***.println(logo);

add(logo);

dx = 2; dy = 1;

suspend = **false**;

colorBox.setSelectedIndex(0);

}

/\*\* Simple animation method that shifts an object. \*/

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

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

**if** (!suspend) advanceOneTimeStep();

pause(***PAUSE\_TIME*** / getCurrentSpeed());

}

}

/\*\* Check for bounces and advance the ball \*/

**private** **void** advanceOneTimeStep() {

**double** x = logo.getBounds().getX();

**double** y = logo.getBounds().getY();

**if** (x + dx < 0 || x + logo.getWidth() + dx > getWidth())

dx = -dx;

**if** (y + dy < 0 || y + logo.getHeight() + dy > getHeight())

dy = -dy;

logo.move(dx, dy);

}

/\*\*

\* Computes the current percentage of the window size to use for the logo size.

\* **@return** the percentage

\*/

**public** **double** getCurrentPercentage(){

**return** size.getValue() / 100.;

}

/\*\*

\* Computes the current speed factor for the animation.

\* **@return** the current speed.

\*/

**private** **double** getCurrentSpeed(){

**if**(speeds[0].isSelected()) **return** ***SLOW\_SPEED***;

**if**(speeds[2].isSelected()) **return** ***FAST\_SPEED***;

**return** ***MEDIUM\_SPEED***;

}

}

***GTargetLogo Class:***

**package** unit6.logoProject;

**import** java.awt.Color;

**import** acm.graphics.\*;

/\*\*

\* Makes the Target logo.

\* The origin of the GCompound is in the center of the logo.

\*

\* **@author** Dr. Mark Jones

\*/

@SuppressWarnings("serial")

**public** **class** GTargetLogo **extends** GCompound {

**private** **double** size;

**private** GOval outerCircle, middleCircle, innerCircle;

/\*\*

\* Makes a target logo object of a given size.

\* **@param** size size of the logo

\*/

**public** GTargetLogo(**double** size) {

**this**.size = size;

**double** outerRedCircleSize = size;

outerCircle = **new** GOval(outerRedCircleSize, outerRedCircleSize);

outerCircle.setFilled(**true**);

outerCircle.setFillColor(Color.RED);

outerCircle.setLocation(-outerRedCircleSize / 2, -outerRedCircleSize / 2);

add(outerCircle);

// make inner white circle

**double** whiteCircleSize = 2./3.\*size;

middleCircle = **new** GOval(whiteCircleSize, whiteCircleSize);

middleCircle.setFilled(**true**);

middleCircle.setFillColor(Color.WHITE);

middleCircle.setLocation(-whiteCircleSize / 2, -whiteCircleSize / 2);

add(middleCircle);

// make small inner red circle

**double** innerRedCircleSize = 1./3.\*size;

innerCircle = **new** GOval(innerRedCircleSize, innerRedCircleSize);

innerCircle.setFilled(**true**);

innerCircle.setFillColor(Color.RED);

innerCircle.setLocation(-innerRedCircleSize / 2, -innerRedCircleSize / 2);

add(innerCircle);

}

/\*\*

\* Makes a target logo object at a given canvas location of a given size.

\* **@param** x x-coordinate of the logo

\* **@param** y y-coordinate of the logo

\* **@param** size size of the logo

\*/

**public** GTargetLogo(**double** x, **double** y, **double** size) {

**this**(size);

setLocation(x, y);

}

/\*\*

\* Sets the color of the logo's inner and outer circles.

\*/

**public** **void** setColor(Color c)

{

outerCircle.setFillColor(c);

innerCircle.setFillColor(c);

}

**public** String toString() {

String s = **super**.toString();

**return** s.substring(0, s.length()-1) + ", size=" + size +"]";

}

}