

Action.java

```
1 package com.eimacs.lab05;
2 import com.eimacs.lab05gui.Turtle;
3 import java.awt.Graphics;
4 /**
5  *
6  * @author IMACS Curriculum Development Group
7  * @version 2.0 January 14, 2015
8  */
9 public abstract class Action
10 {
11     /**
12      * Class constructor
13      */
14     public Action()
15     {
16     }
17     public abstract void execute( Turtle t, Graphics g );
18 }
19
```

APPoint.java

```
1 package com.eimacs.lab05;
2
3 /**
4  * Models a point in a plane
5  *
6  * @author Naomi Spargo
7  * @version 1.0 February 18, 2017
8  */
9 public class APPoint
10 {
11     private double myX;
12     private double myY;
13
14     public APPoint( double x, double y )
15     {
16         myX = x;
17         myY = y;
18     }
19
20     public double getX() { return myX; }
21     public double getY() { return myY; }
22     public void setX( double x ) { myX = x; }
23     public void setY( double y ) { myY = y; }
24 }
25
```

MoveBack.java

```
1 package com.eimacs.lab05;
2 import com.eimacs.lab05gui.Turtle;
3 import java.awt.Graphics;
4 /**
5  *
6  * @author Naomi Spargo
7  * @version 1.0 February 18, 2017
8  */
9 public class MoveBack extends Action
10 {
11     private int mySteps;
12     public MoveBack(int step)
13     {
14         mySteps=step;
15     }
16     public String toString()
17     {
18         return "Back "+mySteps+"";
19     }
20     public void execute( Turtle t, Graphics g )
21     {
22         APPoint p = t.getPosition();
23         double h = Math.toRadians( t.getHeading() );
24
25         APPoint newPoint = new APPoint( p.getX() - mySteps * Math.sin( h ),
26                                         p.getY() + mySteps * Math.cos( h ) );
27
28         t.lineTo( newPoint, g );
29     }
30 }
31 }
```

MoveForward.java

```
1 package com.eimacs.lab05;
2 import com.eimacs.lab05gui.Turtle;
3 import java.awt.Graphics;
4
5 /**
6  *
7  * @author Naomi Spargo
8  * @version 1.0 February 18, 2017
9  */
10 public class MoveForward extends Action
11 {
12     private int mySteps;
13     public MoveForward(int step)
14     {
15
16         mySteps=step;
17     }
18     public String toString()
19     {
20         return "Forward "+mySteps+"";
21     }
22
23     public void execute( Turtle t, Graphics g )
24     {
25         APPoint p = t.getPosition();
26         double h = Math.toRadians( t.getHeading() );
27
28         APPoint newPoint = new APPoint( p.getX() + mySteps * Math.sin( h ),
29                                         p.getY() - mySteps * Math.cos( h ) );
30
31         t.lineTo( newPoint, g );
32 }}
```

RepeatAction.java

```
1 package com.eimacs.lab05;
2
3 import java.awt.Graphics;
4
5 import com.eimacs.lab05gui.Turtle;
6
7 public class RepeatAction extends Action {
8     private int nRepeats;
9     private TurtleProgram myTurtleProgram;
10
11     public RepeatAction(int n, TurtleProgram t) {
12         nRepeats = n;
13         myTurtleProgram = t;
14     }
15
16     public void execute(Turtle t, Graphics g) {
17         for (int i = 0; i < nRepeats; i++)
18             myTurtleProgram.execute(t, g);
19     }
20
21     public String toString() {
22         String n = "Repeat " + nRepeats + " ";
23         n += "\n";
24         n += "[";
25         n += myTurtleProgram.toString();
26         n += "\n";
27         n += "];";
28         return n;
29     }
30 }
31
```

TurnLeft.java

```
1 package com.eimacs.lab05;
2 import com.eimacs.lab05gui.Turtle;
3 import java.awt.Graphics;
4
5 /**
6  *
7  * @author Naomi Spargo
8  * @version 1.0 February 18, 2017
9  */
10 public class TurnLeft extends Action
11 {
12     private double myAngle;
13     public TurnLeft(double angle)
14     {
15         myAngle=angle;
16     }
17     public String toString()
18     {
19         return "Left "+myAngle+"";
20     }
21     public void execute( Turtle t, Graphics g )
22     {
23         t.setHeading( t.getHeading() - myAngle );
24     }
25 }
```

TurnRight.java

```
1 package com.eimacs.lab05;
2 import com.eimacs.lab05gui.Turtle;
3 import java.awt.Graphics;
4 /**
5  *
6  * @author Naomi Spargo
7  * @version 1.0 February 18, 2017
8  */
9 public class TurnRight extends Action
10 {
11     private double myAngle;
12     public TurnRight(double angle)
13     {
14         myAngle=angle;
15     }
16     public String toString()
17     {
18         return "Right "+myAngle+"";
19     }
20     public void execute( Turtle t, Graphics g )
21     {
22         t.setHeading( t.getHeading() + myAngle );
23     }
24 }
```

TurtleProgram.java

```
1 package com.eimacs.lab05;
2
3 import java.util.ArrayList;
4 import com.eimacs.lab05gui.Turtle;
5 import java.awt.Graphics;
6 /**
7  *
8  * @author Naomi Spargo
9  * @version 1.0 February 18, 2017
10 */
11 public class TurtleProgram
12 { private ArrayList<Action> myActions;
13   private boolean isValid;
14   public TurtleProgram()
15   {
16       myActions=new ArrayList<Action>();
17       isValid=false;
18   }
19   public void setIsValid(boolean b)
20   {
21       isValid=b;
22   }
23   public void addAction(Action a)
24   {
25       myActions.add(a);
26       isValid=false;
27   }
28   public String toString()
29   {
30       String ans="";
31       if(myActions.size()==0)
32           return ans;
33       ans+= myActions.get(0);
34       for(int i=1; i<myActions.size();i++)
35       {
36           ans+="\n";
37           ans+=myActions.get(i);
38       }
39       return ans;
40   }
41   public void execute(Turtle t, Graphics g)
42   {
43       if (isValid)
44           {for(Action a: myActions)
45             {
46                 a.execute(t, g);
47             }}
48   }
```


TurtleProgram.java

```
49 public void showTurtle( Turtle t, Graphics g )
50 {
51     int[] xCoords=new int[3];
52     int[] yCoords=new int[3];
53     APPoint p = t.getPosition();
54     double h = Math.toRadians( t.getHeading());
55     APPoint bl=new APPoint(p.getX() - 30 * Math.sin( h+Math.toRadians(15)),
56                             p.getY() + 30 * Math.cos( h+Math.toRadians(15) ) );
57     APPoint br=new APPoint(p.getX() - 30 * Math.sin( h -Math.toRadians(15)),
58                             p.getY() + 30 * Math.cos( h -Math.toRadians(15) ) );
59
60     xCoords[0]=(int)p.getX();
61     yCoords[0]=(int)p.getY();
62     xCoords[1]=(int)bl.getX();
63     yCoords[1]=(int)bl.getY();
64     xCoords[2]=(int)br.getX();
65     yCoords[2]=(int)br.getY();
66     g.drawPolygon(xCoords, yCoords, 3);
67 }
68 }
69
```

Lab05Runner.java

```
1 package com.eimacs.lab05gui;
2
3 import com.eimacs.lab05.*;
4
5 /**
6  *
7  * @author Naomi Spargo
8  * @version 1.0 February 18, 2017
9  */
10 public class Lab05Runner
11 {
12     private static TurtleWindow theTurtleWindow;
13     /**
14      * The main method
15      *
16      * @param args the command line arguments
17      */
18     public static TurtleWindow getTurtleWindow()
19     {
20         return theTurtleWindow;
21     }
22     public static void main( String[] args )
23     {
24
25         theTurtleWindow = new TurtleWindow();
26     }
27
28 }
29
```

Turtle.java

```
1 package com.eimacs.lab05gui;
2
3 import com.eimacs.lab05.APPoint;
4 import java.awt.Graphics;
5
6 public class Turtle {
7     private APoint myPosition;
8     private double myHeading;
9
10    public Turtle() {
11        myPosition = new APoint(0, 0);
12        myHeading = 0;
13    }
14
15    public APoint getPosition() {
16        return myPosition;
17    }
18
19    public double getHeading() {
20        return myHeading;
21    }
22
23    public void setHeading(double d) {
24        myHeading = d;
25    }
26
27    public void lineTo(APoint newPoint, Graphics g) {
28        g.drawLine((int) myPosition.getX(), (int) myPosition.getY(),
29                (int) newPoint.getX(), (int) newPoint.getY());
30        myPosition = newPoint;
31    }
32 }
33
```

TurtleController.java

```
1 package com.eimacs.lab05gui;
2
3 import java.awt.event.ActionEvent;
4
5 import javax.swing.JOptionPane;
6
7 /**
8  *
9  * @author IMACS Curriculum Development Group
10 * @version 2.0 January 14, 2015
11 */
12 public class TurtleController extends TurtleProgrammer
13 {
14
15     private TurtlePlane myTurtlePlane;
16     /**
17      * Class constructor
18      */
19     public TurtleController(TurtlePlane turtleplane)
20     {
21
22         myTurtlePlane=turtleplane;
23         myTurtlePlane.setTurtleController(this);
24         initialize();
25     }
26
27     /**
28      * Gets this TurtleController's program
29      *
30      * @return this TurtleController's program
31      */
32     private String getInput( String prompt )
33     {
34         return JOptionPane.showInputDialog( this, prompt );
35     }
36
37     /**
38      * Overrides ActionListener's actionPerformed method
39      *
40      * @param e the event provoking an action to be performed
41      */
42     public void actionPerformed( ActionEvent e )
43     {
44
45         String actionName = e.getActionCommand();
46
47         if ( "Execute".equals( actionName ) )
48             getTurtleProgram().setIsValid( true );
```

TurtleController.java

```
49         else
50             super.actionPerformed( e );
51
52         executeProgram();
53
54     }
55
56
57     /**
58      * The class initializer
59      */
60     private void initialize()
61     {
62         //add control buttons
63         addButton( "Execute" );
64         addButton( "Reset" );
65     }
66
67     /**
68      * Adds a button to this TurtleController
69      *
70      * @param buttonName the name (and action command) of the button
71      */
72
73
74     public void executeProgram()
75     {
76         myTurtlePlane.drawPlane();
77     }
78 }
79
```

TurtlePlane.java

```
1 package com.eimacs.lab05gui;
2
3 import com.eimacs.lab05.APPoint;
4 import java.awt.Color;
5 import java.awt.Dimension;
6 import java.awt.Graphics;
7 import javax.swing.BorderFactory;
8 import javax.swing.JPanel;
9
10 public class TurtlePlane extends JPanel
11 {
12     private TurtleController myTurtleController;
13
14     public TurtlePlane()
15     {
16         setBorder( BorderFactory.createLoweredBevelBorder() );
17         setBackground( Color.decode( "0xEDFFED" ) );
18         setPreferredSize( new Dimension( 300, 400 ) );
19     }
20
21     public void drawPlane()
22     {
23         repaint();
24     }
25
26     public void paintMe( Graphics g )
27     {
28         Turtle t = new Turtle();
29
30         APPoint startPoint = t.getPosition();
31         startPoint.setX( 150 );
32         startPoint.setY( 200 );
33
34         myTurtleController.getTurtleProgram().execute( t, g );
35         myTurtleController.getTurtleProgram().showTurtle(t,g);
36     }
37
38     public void paintComponent( Graphics g )
39     {
40         super.paintComponent( g );
41         paintMe( g );
42     }
43
44     public void setTurtleController( TurtleController tc )
45     {
46         myTurtleController = tc;
47     }
48 }
```

TurtleProgrammer.java

```
1 package com.eimacs.lab05gui;
2
3 import com.eimacs.lab05.*;
4 import com.eimacs.lab05.TurtleProgram;
5 import java.awt.Color;
6 import java.awt.Dimension;
7 import java.awt.FlowLayout;
8 import java.awt.event.ActionEvent;
9 import java.awt.event.ActionListener;
10 import javax.swing.BorderFactory;
11 import javax.swing.JButton;
12 import javax.swing.JOptionPane;
13 import javax.swing.JPanel;
14 import javax.swing.JScrollPane;
15 import javax.swing.JTextArea;
16
17 /**
18  *
19  * @author IMACS Curriculum Development Group
20  * @version 2.0 January 14, 2015
21  */
22 public class TurtleProgrammer extends JPanel implements ActionListener {
23     /** This TurtleProgrammer's program display area */
24     private JTextArea myProgramDisplay;
25     /** This TurtleProgrammer's program */
26     private TurtleProgram myTurtleProgram;
27     private TurtlePlane myTurtlePlane;
28
29     /**
30      * Class constructor
31      */
32     public TurtleProgrammer() {
33         setLayout(new FlowLayout());
34         setBorder(BorderFactory.createRaisedBevelBorder());
35         setBackground(Color.gray);
36         setPreferredSize(new Dimension(190, 350));
37         initialize();
38     }
39
40     /**
41      * Gets this TurtleProgrammer's program
42      *
43      * @return this TurtleProgrammer's program
44      */
45     private String getInput(String prompt) {
46         return JOptionPane.showInputDialog(this, prompt);
47     }
48 }
```

TurtleProgrammer.java

```

49  public TurtleProgram getTurtleProgram() {
50      return myTurtleProgram;
51  }
52
53  /**
54   * Overrides ActionListener's actionPerformed method
55   *
56   * @param e
57   *         the event provoking an action to be performed
58   */
59  public void actionPerformed(ActionEvent e) {
60      String actionName = e.getActionCommand();
61      if ("Forward".equals(actionName)) {
62          String input = getInput("How many steps?");
63          if (input != null && !input.trim().equals("")) {
64              int steps = Integer.parseInt(input);
65              myTurtleProgram.addAction(new MoveForward(steps));
66          }
67      } else if ("Back".equals(actionName)) {
68          String input = getInput("How many steps?");
69          if (input != null && !input.trim().equals("")) {
70              int steps = Integer.parseInt(input);
71              myTurtleProgram.addAction(new MoveBack(steps));
72          }
73      } else if ("Left".equals(actionName)) {
74          String input = getInput("How many degrees?");
75          if (input != null && !input.trim().equals("")) {
76              double degrees = Double.parseDouble(input);
77              myTurtleProgram.addAction(new TurnLeft(degrees));
78          }
79      } else if ("Right".equals(actionName)) {
80          String input = getInput("How many degrees?");
81          if (input != null && !input.trim().equals("")) {
82              double degrees = Double.parseDouble(input);
83              myTurtleProgram.addAction(new TurnRight(degrees));
84          }
85      } else if ("Reset".equals(actionName)) {
86          myTurtleProgram = new TurtleProgram();
87      }
88      else if ("Repeat".equals(actionName)) {
89          String input = getInput("How many times?");
90          Lab05Runner.getTurtleWindow().incNDepth();
91          if (input != null && !input.trim().equals("")) {
92              int repeats = Integer.parseInt(input);
93              new TurtleRepeaterDialog(repeats, myTurtleProgram);
94          }
95      }
96      else {

```


TurtleProgrammer.java

```

97         JOptionPane.showMessageDialog(this, actionName);
98     }
99
100     displayProgram();
101
102 }
103
104 /**
105  * The class initializer
106  */
107 private void initialize() {
108     myTurtleProgram = new TurtleProgram();
109
110     // add action buttons
111     String[] buttons = { "Forward", "Back", "Left", "Right", "Repeat" };
112     for (String bName : buttons)
113         addButton(bName);
114
115     // add text area for displaying the program
116     myProgramDisplay = new JTextArea(12, 10);
117     myProgramDisplay.setEditable(false);
118     JScrollPane areaScrollPane = new JScrollPane(myProgramDisplay);
119     areaScrollPane.setVerticalScrollBarPolicy
120 (JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
121     add(areaScrollPane);
122
123     // add control buttons
124 }
125
126 /**
127  * Adds a button to this TurtleProgrammer
128  *
129  * @param buttonName
130  *         the name (and action command) of the button
131  */
132 public void addButton(String buttonName) {
133     JButton newButton = new JButton(buttonName);
134     newButton.setActionCommand(buttonName);
135     newButton.addActionListener(this);
136     add(newButton);
137 }
138
139 public void displayProgram() {
140     myProgramDisplay.setText(myTurtleProgram.toString());
141 }
142
143 }

```


TurtleRepeater.java

```
1 package com.eimacs.lab05gui;
2
3 import java.awt.event.ActionEvent;
4
5 import com.eimacs.lab05.Action;
6 import com.eimacs.lab05.RepeatAction;
7 import com.eimacs.lab05.TurtleProgram;
8
9 public class TurtleRepeater extends TurtleProgrammer {
10     private TurtleRepeaterDialog myDialog;
11     private TurtleProgram parentProgram;
12     int nRepeats;
13
14     public TurtleRepeater(TurtleRepeaterDialog trd, TurtleProgram tp, int n) {
15         myDialog = trd;
16         parentProgram = tp;
17         nRepeats = n;
18         this.addButton("Done");
19         this.addButton("Reset");
20     }
21
22     public void saveRepeat() {
23         TurtleProgram tp = this.getTurtleProgram();
24         tp.setIsValid(true);
25         Action a = new RepeatAction(nRepeats, tp);
26         parentProgram.addAction(a);
27     }
28
29     public void actionPerformed(ActionEvent e) {
30
31         String actionName = e.getActionCommand();
32
33         if ("Done".equals(actionName)) {
34             this.saveRepeat();
35             Lab05Runner.getTurtleWindow().decNDepth();
36             myDialog.dispose();
37         } else
38             super.actionPerformed(e);
39     }
40 }
41
```

TurtleRepeaterDialog.java

```
1 package com.eimacs.lab05gui;
2
3 import com.eimacs.lab05.TurtleProgram;
4 import java.awt.FlowLayout;
5 import javax.swing.JDialog;
6 import javax.swing.JPanel;
7 import static javax.swing.WindowConstants.DISPOSE_ON_CLOSE;
8
9 public class TurtleRepeaterDialog extends JDialog {
10     private TurtleRepeater myTurtleRepeater;
11
12     public TurtleRepeaterDialog(int nTimes, TurtleProgram tp) {
13         super(Lab05Runner.getTurtleWindow(), "Repeater!", true);
14         myTurtleRepeater = new TurtleRepeater(this, tp, nTimes);
15         initialize();
16     }
17
18     private void initialize() {
19         JPanel layoutPanel = new JPanel();
20         layoutPanel.setLayout(new FlowLayout());
21         layoutPanel.add(myTurtleRepeater);
22         getContentPane().add(layoutPanel);
23         setDefaultCloseOperation(DISPOSE_ON_CLOSE);
24         pack();
25         setSize(200, 390);
26         int nd = Lab05Runner.getTurtleWindow().getNDepth();
27         setLocation(40 + 25 * nd, 40 + 25 * nd);
28         setVisible(true);
29     }
30 }
31 }
```

TurtleWindow.java

```
1 package com.eimacs.lab05gui;
2
3 import java.awt.FlowLayout;
4 import javax.swing.JFrame;
5 import javax.swing.JPanel;
6
7 /**
8  *
9  * @author IMACS Curriculum Development Group
10 * @version 2.0 January 14, 2015
11 */
12 public class TurtleWindow extends JFrame
13 {
14     /** This TurtleWindow's TurtleController */
15     private TurtleController myTurtleController;
16     private TurtlePlane myTurtlePlane;
17     private int nDepth;
18     /**
19      * Class constructor
20      */
21     public TurtleWindow()
22     {
23         super( "AP Lab 05" );
24         myTurtlePlane= new TurtlePlane();
25         myTurtleController = new TurtleController(myTurtlePlane);
26         nDepth=-1;
27         initialize();
28     }
29
30     /**
31      * The class initializer
32      */
33     private void initialize()
34     {
35         JPanel layoutPanel = new JPanel();
36         layoutPanel.setLayout( new FlowLayout() );
37         layoutPanel.add( myTurtlePlane );
38         layoutPanel.add( myTurtleController );
39
40         getContentPane().add( layoutPanel );
41
42         setDefaultCloseOperation( EXIT_ON_CLOSE );
43         pack();
44         setSize( 500, 440 );
45         setLocationRelativeTo( null );
46         setVisible( true );
47     }
48     public int getNDepth()
```

TurtleWindow.java

```
49     {  
50         return nDepth;  
51     }  
52     public void incNDepth()  
53     {  
54         nDepth+=1;  
55     }  
56     public void decNDepth()  
57     {  
58         nDepth-=1;  
59     }  
60 }  
61
```