Java OOP: Modifications to the Turtle and SimpleTurtle Classes*

R.G. (Dick) Baldwin

Abstract

Learn how to modify Ericson's Turtle and SimpleTurtle classes for a variety of purposes.

1 Table of Contents

2 Preface

This module is one of a series of modules designed to teach you about Object-Oriented Programming (OOP) using Java.

The program described in this module requires the use of the Guzdial-Ericson multimedia class library. You will find download, installation, and usage instructions for the library at Java OOP: The Guzdial-Ericson Multimedia Class Library [1] .

2.1 Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

*Version 1.1: Aug 19, 2012 1:23 pm -0500
†http://creativecommons.org/licenses/by/3.0/
[1]http://cnx.org/content/m44148/latest/

2.1.1 Figures

2.1.2 Listings

3 Preview

Program specifications

Write a program named **Prob02** that uses the class definition shown in Listing 4 (p. 7) and Ericson's media library to produce the graphic output image shown in Figure 1 (p. 3) .

Required screen output.

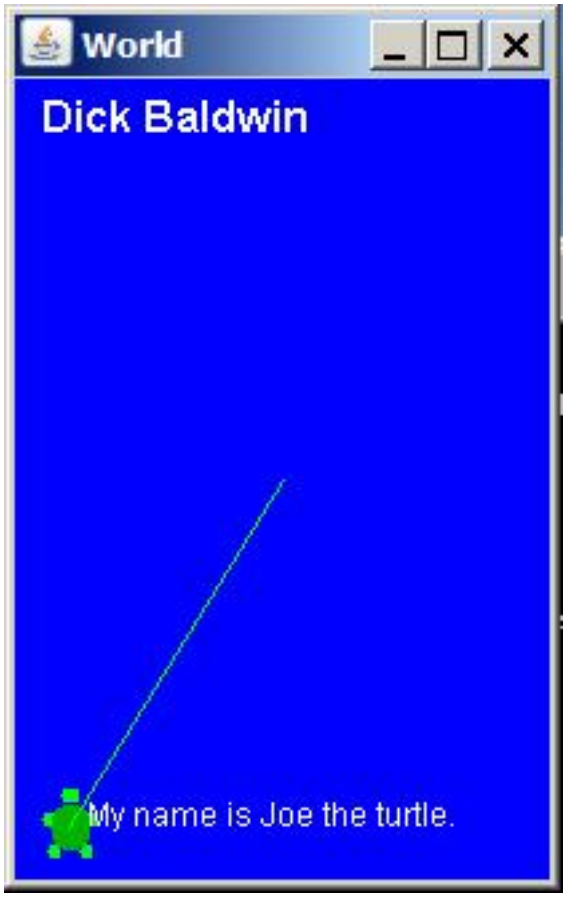

Figure 1: Required screen output.

No new classes allowed

You may not define any new classes to cause your program to behave as required, and you may not modify the class definition for the class named **Prob02** given in Listing 4 (p. 7) .

Files in your folder

You must copy and modify *(as necessary)* the media classes named **Turtle.java** and **SimpleTurtle.java** from Ericson's library to cause your program to produce the required output.

Your folder must contain only the following class files and source-code files:

- Prob02.class
- Prob02.java
- SimpleTurtle.class
- SimpleTurtle.java
- Turtle.class
- Turtle.java

Output text

In addition to the output image described above, your program must produce the text output shown in Figure 2 (p. 4) on the command- line screen

Required text output.

```
    Dick Baldwin
My name is Joe the turtle.
```

Figure 2: Required text output.

Required modifications

By comparing the default behavior of the **Turtle** and **SimpleTurtle** classes with the requirements of this program, it can be determined that the following modifications to the **Turtle** and **SimpleTurtle** classes are required to meet the specifications.

1. Modify the **Turtle** class to cause the student's name to be displayed on the command line.
2. Modify the **Turtle** and **SimpleTurtle** classes to accept and save a **String** parameter in addition to the **World** parameter when the Turtle object is constructed.
3. Modify the **SimpleTurtle** class to cause the default background of the world to be BLUE.
4. Modify the **SimpleTurtle** class to cause the student's name to be displayed near the top of the **World** image.
5. Modify the **toString** method in the **SimpleTurtle** class to cause it to return the value of the **String** parameter whenever the **toString** method is called. This causes the **drawInfoString** method to display the string in place of its normal behavior. It also causes the last statement in Listing 4 (p. 7) to display the turtle's name.

4 Discussion and sample code

4.1 Modifications to the Turtle class

Ericson's **Turtle** class was modified according to the first two items listed above under required modifications [2] .

A complete listing of the modified **Turtle** class is provided in Listing 5 (p. 7) near the end of the module.

Modification to the Turtle constructor

The **Turtle** class has several overloaded constructors. One of the constructors was modified to accept a **String** parameter in addition to the **World** parameter and pass the new parameter along to the superclass constructor. The code is shown in Listing 1 (p. 4) .

Listing 1: Modified Turtle constructor.

[2] http://cnx.org/content/m44348/latest/Lecture02.htm#Required_modifications

```
    public Turtle (ModelDisplay modelDisplay,
                   String turtleName){
  // let the parent constructor handle it
  super(modelDisplay,turtleName);
  System.out.println("Dick Baldwin");
}
```

A **println** statement was also added to the modified constructor to cause it to display the student's name on the command line screen when the **Turtle** object is constructed as shown in Figure 2.

4.2 Modifications to the SimpleTurtle class

A complete listing of the modified **SimpleTurtle** class is shown in Listing 6 (p. 9) near the end of the module.

The superclass of the Turtle class

The **SimpleTurtle** class is the superclass of the **Turtle** class. Therefore, the **SimpleTurtle** class must be modified to accept the **String** parameter passed to the superclass in Listing 1 (p. 4) . This was accomplished by modifying one of the constructors of the **SimpleTurtle** class as shown in Listing 2 (p. 5) .

Listing 2: Modified SimpleTurtle constructor.

```
    String turtleName = null;

public SimpleTurtle(ModelDisplay display,
                    String turtleName){

  // call constructor that takes x and y
  this((int) (display.getWidth() / 2),
       (int) (display.getHeight() / 2));
  modelDisplay = display;
  display.addModel(this);

  //THIS IS THE MODIFICATION
  this.turtleName = turtleName;
  Picture picture = ((World)(display)).getPicture();
  picture.setAllPixelsToAColor(Color.BLUE);
  picture.addMessage("Dick Baldwin",10,20);
}
```

The modification is shown in the last four statements in Listing 2 (p. 5) . This modification satisfies items 2, 3, and 4 listed earlier under required modifications [3] .

Modified toString method

Listing 3 (p. 5) shows the modified **toString** method that satisfies item 5 listed above under required modifications [4] .

Listing 3: Modified toString method.

[3] http://cnx.org/content/m44348/latest/Lecture02.htm#Required_modifications
[4] http://cnx.org/content/m44348/latest/Lecture02.htm#Required_modifications

```
    public String toString(){
  //return this.name + " turtle at " + this.xPos + ", " +
  //      this.yPos + " heading " + this.heading + ".";
  return "My name is " + turtleName + " the turtle.";
}//end toString
```

The original code was preserved as comments in Listing 3 (p. 5) , and the new modified code is shown below those comments.

5 Run the program

I encourage you to copy the code from Listing 4 (p. 7) , Listing 5 (p. 7) , and Listing 6 (p. 9) . Compile the code and execute it. Experiment with the code, making changes, and observing the results of your changes. Make certain that you can explain why your changes behave as they do.

6 Summary

You learned how to:

1. Modify the **Turtle** class to cause the student's name to be displayed on the command line.
2. Modify the **Turtle** and **SimpleTurtle** classes to accept and save a **String** parameter in addition to the **World** parameter when the **Turtle** object is constructed.
3. Modify the **SimpleTurtle** class to cause the default background of the world to be BLUE.
4. Modify the **SimpleTurtle** class to cause the student's name to be displayed near the top of the **World** image.
5. Modify the **toString** method in the **SimpleTurtle** class to cause it to return the value of the **String** parameter whenever the **toString** method is called. This causes the **drawInfoString** method to display the string in place of its normal behavior. It also causes the last statement in Listing 4 (p. 7) to display the turtle's name.

7 What's next?

In the next module, you will learn how to incorporate GUI components into a **World** object. In particular, you will learn how to add a **JButton** object to a **World** object and register an action listener on the button to control the behavior of the program.

8 Miscellaneous

This section contains a variety of miscellaneous information.

NOTE: **Housekeeping material**

- Module name: Java OOP: Modifications to the Turtle and SimpleTurtle Classes
- File: Java3104.htm
- Revised: 08/19/12

NOTE: **Disclaimers: Financial** : Although the Connexions site makes it possible for you to download a PDF file for this module at no charge, and also makes it possible for you to purchase a pre-printed version of the PDF file, you should be aware that some of the HTML elements in this module may not translate well into PDF.

I also want you to know that, I receive no financial compensation from the Connexions website even if you purchase the PDF version of the module.

In the past, unknown individuals have copied my modules from cnx.org, converted them to Kindle books, and placed them for sale on Amazon.com showing me as the author. I neither receive compensation for those sales nor do I know who does receive compensation. If you purchase such a book, please be aware that it is a copy of a module that is freely available on cnx.org and that it was made and published without my prior knowledge.

Affiliation : I am a professor of Computer Information Technology at Austin Community College in Austin, TX.

9 Complete program listing

Complete listings of the programs discussed in this module are shown below.

Listing 4: Source code for the class named Prob02.

```
    import java.awt.Color;

public class Prob02{
  public static void main(String[] args){
    World mars = new World(200,300);
    Turtle joe = new Turtle(mars,"Joe");
    joe.moveTo(20,280);
    joe.setInfoColor(Color.WHITE);
    joe.setShowInfo(true);
    System.out.println(joe);
  }//end main method
}//end class Prob02
```

Listing 5: Modified Turtle class.

```
    /*12/23/0812/23/08 This class and the class named
SimpleTurtle were modified to:

Accept and save a String parameter in addition to the
World parameter when the Turtle object is constructed.

Modify the toString method to cause it to return the
value of the String parameter whenever the toString
method is called. This causes the drawInfoString method
to display the string in place of its normal behavior.

Cause the default background of the world to be BLUE.

Cause the student's name to be displayed near the top of
the World image.

Cause the student's name as well as the turtle's name to
be displayed on the command line.
```

```
*/

/**
 * Class that represents a turtle which is similar to a Logo turtle.
 * This class inherts from SimpleTurtle and is for students
 * to add methods to.
 *
 * Copyright Georgia Institute of Technology 2004
 * @author Barb Ericson ericson@cc.gatech.edu
 */
public class Turtle extends SimpleTurtle
{
  ////////////////// constructors ///////////////////////

  /** Constructor that takes the x and y and a picture to
   * draw on
   * @param x the starting x position
   * @param y the starting y position
   * @param picture the picture to draw on
   */
  public Turtle (int x, int y, Picture picture)
  {
    // let the parent constructor handle it
    super(x,y,picture);
  }

  /** Constructor that takes the x and y and a model
   * display to draw it on
   * @param x the starting x position
   * @param y the starting y position
   * @param modelDisplayer the thing that displays the model
   */
  public Turtle (int x, int y,
                 ModelDisplay modelDisplayer)
  {
    // let the parent constructor handle it
    super(x,y,modelDisplayer);
  }

  //THIS IS A MODIFICATION
  //The following constructor was modified to accept and
  // save a String parameter and pass it to the superclass
  // constructor.
  /** Constructor that takes the model display
   * @param modelDisplay the thing that displays the model
   */
  public Turtle (ModelDisplay modelDisplay,
                 String turtleName){
    // let the parent constructor handle it
    super(modelDisplay,turtleName);
    System.out.println("Dick Baldwin");
```

```
  }

  /**
   * Constructor that takes a picture to draw on
   * @param p the picture to draw on
   */
  public Turtle (Picture p)
  {
    // let the parent constructor handle it
    super(p);
  }

  ////////////////// methods /////////////////////


} // this } is the end of class Turtle, put all new methods before this
```

Listing 6: Modified SimpleTurtle class.

```
    import javax.swing.*;
import java.awt.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.util.Observer;
import java.util.Random;

/*12/23/08 This class and the class named Turtle were
modified to:
Accept and save a String parameter in addition to the
World parameter when the Turtle object is constructed.

Modify the toString method to cause it to return the
value of the String parameter whenever the toString
method is called. This causes the drawInfoString method
to display the string in place of its normal behavior.

Cause the default background of the world to be BLUE.

Cause the student's name to be displayed near the top of
the World image.

Cause the student's name as well as the turtle's name to
be displayed on the command line.
 */

/**
 * Class that represents a Logo-style turtle.  The turtle
 * starts off facing north.
 * A turtle can have a name, has a starting x and y position,
 * has a heading, has a width, has a height, has a visible
```

```
 * flag, has a body color, can have a shell color, and has a pen.
 * The turtle will not go beyond the model display or picture
 * boundaries.
 *
 * You can display this turtle in either a picture or in
 * a class that implements ModelDisplay.
 *
 * Copyright Georgia Institute of Technology 2004
 * @author Barb Ericson ericson@cc.gatech.edu
 */
public class SimpleTurtle
{
  ///////////////// fields ////////////////////////

  /** count of the number of turtles created */
  private static int numTurtles = 0;

  /** array of colors to use for the turtles */
  private static Color[] colorArray = { Color.green, Color.cyan, new Color(204,0,204), Color.gray};

  /** who to notify about changes to this turtle */
  private ModelDisplay modelDisplay = null;

  /** picture to draw this turtle on */
  private Picture picture = null;

  /** width of turtle in pixels */
  private int width = 15;

  /** height of turtle in pixels */
  private int height = 18;

  /** current location in x (center) */
  private int xPos = 0;

  /** current location in y (center) */
  private int yPos = 0;

  /** heading angle */
  private double heading = 0;  // default is facing north

  /** pen to use for this turtle */
  private Pen pen = new Pen();

  /** color to draw the body in */
  private Color bodyColor = null;

  /** color to draw the shell in */
  private Color shellColor = null;

  /** color of information string */
```

```
  private Color infoColor = Color.black;

  /** flag to say if this turtle is visible */
  private boolean visible = true;

  /** flag to say if should show turtle info */
  private boolean showInfo = false;

  /** the name of this turtle */
  private String name = "No name";

  ////////////////// constructors ////////////////////

  /**
   * Constructor that takes the x and y position for the
   * turtle
   * @param x the x pos
   * @param y the y pos
   */
  public SimpleTurtle(int x, int y)
  {
    xPos = x;
    yPos = y;
    bodyColor = colorArray[numTurtles % colorArray.length];
    setPenColor(bodyColor);
    numTurtles++;
  }

  /**
   * Constructor that takes the x and y position and the
   * model displayer
   * @param x the x pos
   * @param y the y pos
   * @param display the model display
   */
  public SimpleTurtle(int x, int y, ModelDisplay display)
  {
    this(x,y); // call constructor that takes x and y
    modelDisplay = display;
    display.addModel(this);
  }
  //THIS IS A MODIFICATION
  //The following constructor was modified to accept and
  // save a String parameter.
  String turtleName = null;
  /**
   * Constructor that takes a model display and adds
   * a turtle in the middle of it
   * @param display the model display
   */
  public SimpleTurtle(ModelDisplay display,
```

```
                    String turtleName){

    // call constructor that takes x and y
    this((int) (display.getWidth() / 2),
         (int) (display.getHeight() / 2));
    modelDisplay = display;
    display.addModel(this);
    this.turtleName = turtleName;
    Picture picture = ((World)(display)).getPicture();
    //THIS IS A MODIFICATION
    picture.setAllPixelsToAColor(Color.BLUE);
    picture.addMessage("Dick Baldwin",10,20);
  }

  /**
   * Constructor that takes the x and y position and the
   * picture to draw on
   * @param x the x pos
   * @param y the y pos
   * @param picture the picture to draw on
   */
  public SimpleTurtle(int x, int y, Picture picture)
  {
    this(x,y); // call constructor that takes x and y
    this.picture = picture;
    this.visible = false; // default is not to see the turtle
  }

  /**
   * Constructor that takes the
   * picture to draw on and will appear in the middle
   * @param picture the picture to draw on
   */
  public SimpleTurtle(Picture picture)
  {
    // call constructor that takes x and y
    this((int) (picture.getWidth() / 2),
         (int) (picture.getHeight() / 2));
    this.picture = picture;
    this.visible = false; // default is not to see the turtle
  }

  ////////////////////// methods ///////////////////////

  /**
   * Get the distance from the passed x and y location
   * @param x the x location
   * @param y the y location
   */
  public double getDistance(int x, int y)
  {
```

```
    int xDiff = x - xPos;
    int yDiff = y - yPos;
    return (Math.sqrt((xDiff * xDiff) + (yDiff * yDiff)));
  }

  /**
   * Method to turn to face another simple turtle
   */
  public void turnToFace(SimpleTurtle turtle)
  {
    turnToFace(turtle.xPos,turtle.yPos);
  }

   /**
   * Method to turn towards the given x and y
   * @param x the x to turn towards
   * @param y the y to turn towards
   */
  public void turnToFace(int x, int y)
  {
    double dx = x - this.xPos;
    double dy = y - this.yPos;
    double arcTan = 0.0;
    double angle = 0.0;

    // avoid a divide by 0
    if (dx == 0)
    {
      // if below the current turtle
      if (dy > 0)
        heading = 180;

      // if above the current turtle
      else if (dy < 0)
        heading = 0;
    }
    // dx isn't 0 so can divide by it
    else
    {
      arcTan = Math.toDegrees(Math.atan(dy / dx));
      if (dx < 0)
        heading = arcTan - 90;
      else
        heading = arcTan + 90;
    }

    // notify the display that we need to repaint
    updateDisplay();
  }

  /**
```

```
   * Method to get the picture for this simple turtle
   * @return the picture for this turtle (may be null)
   */
  public Picture getPicture() { return this.picture; }

  /**
   * Method to set the picture for this simple turtle
   * @param pict the picture to use
   */
  public void setPicture(Picture pict) { this.picture = pict; }

  /**
   * Method to get the model display for this simple turtle
   * @return the model display if there is one else null
   */
  public ModelDisplay getModelDisplay() { return this.modelDisplay; }

  /**
   * Method to set the model display for this simple turtle
   * @param theModelDisplay the model display to use
   */
  public void setModelDisplay(ModelDisplay theModelDisplay)
  { this.modelDisplay = theModelDisplay; }

  /**
   * Method to get value of show info
   * @return true if should show info, else false
   */
  public boolean getShowInfo() { return this.showInfo; }

  /**
   * Method to show the turtle information string
   * @param value the value to set showInfo to
   */
  public void setShowInfo(boolean value) { this.showInfo = value; }

  /**
   * Method to get the shell color
   * @return the shell color
   */
  public Color getShellColor()
  {
    Color color = null;
    if (this.shellColor == null && this.bodyColor != null)
      color = bodyColor.darker();
    else color = this.shellColor;
    return color;
  }

  /**
   * Method to set the shell color
```

```
   * @param color the color to use
   */
  public void setShellColor(Color color) {  this.shellColor = color; }

  /**
   * Method to get the body color
   * @return the body color
   */
  public Color getBodyColor() { return this.bodyColor; }

  /**
   * Method to set the body color which
   * will also set the pen color
   * @param color the color to use
   */
  public void setBodyColor(Color color)
  {
    this.bodyColor = color;
    setPenColor(this.bodyColor);
  }

  /**
   * Method to set the color of the turtle.
   * This will set the body color
   * @param color the color to use
   */
  public void setColor(Color color) { this.setBodyColor(color); }

  /**
   * Method to get the information color
   * @return the color of the information string
   */
  public Color getInfoColor() { return this.infoColor; }

  /**
   * Method to set the information color
   * @param color the new color to use
   */
  public void setInfoColor(Color color) { this.infoColor = color; }

  /**
   * Method to return the width of this object
   * @return the width in pixels
   */
  public int getWidth() { return this.width; }

  /**
   * Method to return the height of this object
   * @return the height in pixels
   */
  public int getHeight() { return this.height; }
```

```
  /**
   * Method to set the width of this object
   * @param theWidth in width in pixels
   */
  public void setWidth(int theWidth) { this.width = theWidth; }

  /**
   * Method to set the height of this object
   * @param theHeight the height in pixels
   */
  public void setHeight(int theHeight) { this.height = theHeight; }

  /**
   * Method to get the current x position
   * @return the x position (in pixels)
   */
  public int getXPos() { return this.xPos; }

  /**
   * Method to get the current y position
   * @return the y position (in pixels)
   */
  public int getYPos() { return this.yPos; }

  /**
   * Method to get the pen
   * @return the pen
   */
  public Pen getPen() { return this.pen; }

  /**
   * Method to set the pen
   * @param thePen the new pen to use
   */
  public void setPen(Pen thePen) { this.pen = thePen; }

  /**
   * Method to check if the pen is down
   * @return true if down else false
   */
  public boolean isPenDown() { return this.pen.isPenDown(); }

  /**
   * Method to set the pen down boolean variable
   * @param value the value to set it to
   */
  public void setPenDown(boolean value) { this.pen.setPenDown(value); }

  /**
   * Method to lift the pen up
```

```
   */
  public void penUp() { this.pen.setPenDown(false);}

  /**
   * Method to set the pen down
   */
  public void penDown() { this.pen.setPenDown(true);}

  /**
   * Method to get the pen color
   * @return the pen color
   */
  public Color getPenColor() { return this.pen.getColor(); }

  /**
   * Method to set the pen color
   * @param color the color for the pen ink
   */
  public void setPenColor(Color color) { this.pen.setColor(color); }

  /**
   * Method to set the pen width
   * @param width the width to use in pixels
   */
  public void setPenWidth(int width) { this.pen.setWidth(width); }

  /**
   * Method to get the pen width
   * @return the width of the pen in pixels
   */
  public int getPenWidth() { return this.pen.getWidth(); }

  /**
   * Method to clear the path (history of
   * where the turtle has been)
   */
  public void clearPath()
  {
    this.pen.clearPath();
  }

  /**
   * Method to get the current heading
   * @return the heading in degrees
   */
  public double getHeading() { return this.heading; }

  /**
   * Method to set the heading
   * @param heading the new heading to use
   */
```

```
  public void setHeading(double heading)
  {
    this.heading = heading;
  }

  /**
   * Method to get the name of the turtle
   * @return the name of this turtle
   */
  public String getName() { return this.name; }

  /**
   * Method to set the name of the turtle
   * @param theName the new name to use
   */
  public void setName(String theName)
  {
    this.name = theName;
  }

  /**
   * Method to get the value of the visible flag
   * @return true if visible else false
   */
  public boolean isVisible() { return this.visible;}

  /**
   * Method to hide the turtle (stop showing it)
   * This doesn't affect the pen status
   */
  public void hide() { this.setVisible(false); }

  /**
   * Method to show the turtle (doesn't affect
   * the pen status
   */
  public void show() { this.setVisible(true); }

  /**
   * Method to set the visible flag
   * @param value the value to set it to
   */
  public void setVisible(boolean value)
  {
    // if the turtle wasn't visible and now is
    if (visible == false && value == true)
    {
      // update the display
      this.updateDisplay();
    }
```

```
    // set the visibile flag to the passed value
    this.visible = value;
  }

  /**
   * Method to update the display of this turtle and
   * also check that the turtle is in the bounds
   */
  public synchronized void updateDisplay()
  {
    // check that x and y are at least 0
    if (xPos < 0)
      xPos = 0;
    if (yPos < 0)
      yPos = 0;

    // if picture
    if (picture != null)
    {
      if (xPos >= picture.getWidth())
        xPos = picture.getWidth() - 1;
      if (yPos >= picture.getHeight())
        yPos = picture.getHeight() - 1;
      Graphics g = picture.getGraphics();
      paintComponent(g);
    }
    else if (modelDisplay != null)
    {
      if (xPos >= modelDisplay.getWidth())
        xPos = modelDisplay.getWidth() - 1;
      if (yPos >= modelDisplay.getHeight())
        yPos = modelDisplay.getHeight() - 1;
      modelDisplay.modelChanged();
    }
  }

  /**
   * Method to move the turtle foward 100 pixels
   */
  public void forward() { forward(100); }

  /**
   * Method to move the turtle forward the given number of pixels
   * @param pixels the number of pixels to walk forward in the heading direction
   */
  public void forward(int pixels)
  {
    int oldX = xPos;
    int oldY = yPos;

    // change the current position
```

```
    xPos = oldX + (int) (pixels * Math.sin(Math.toRadians(heading)));
    yPos = oldY + (int) (pixels * -Math.cos(Math.toRadians(heading)));

    // add a move from the old position to the new position to the pen
    pen.addMove(oldX,oldY,xPos,yPos);

    // update the display to show the new line
    updateDisplay();
  }

  /**
   * Method to go backward by 100 pixels
   */
  public void backward()
  {
    backward(100);
  }

  /**
   * Method to go backward a given number of pixels
   * @param pixels the number of pixels to walk backward
   */
  public void backward(int pixels)
  {
    forward(-pixels);
  }

  /**
   * Method to move to turtle to the given x and y location
   * @param x the x value to move to
   * @param y the y value to move to
   */
  public void moveTo(int x, int y)
  {
    this.pen.addMove(xPos,yPos,x,y);
    this.xPos = x;
    this.yPos = y;
    this.updateDisplay();
  }

  /**
   * Method to turn left
   */
  public void turnLeft()
  {
   this.turn(-90);
  }

  /**
   * Method to turn right
   */
```

```
  public void turnRight()
  {
    this.turn(90);
  }

  /**
   * Method to turn the turtle the passed degrees
   * use negative to turn left and pos to turn right
   * @param degrees the amount to turn in degrees
   */
  public void turn(int degrees)
  {
    this.heading = (heading + degrees) % 360;
    this.updateDisplay();
  }

  /**
   * Method to draw a passed picture at the current turtle
   * location and rotation in a picture or model display
   * @param dropPicture the picture to drop
   */
  public synchronized void drop(Picture dropPicture)
  {
    Graphics2D g2 = null;

    // only do this if drawing on a picture
    if (picture != null)
      g2 = (Graphics2D) picture.getGraphics();
    else if (modelDisplay != null)
      g2 = (Graphics2D) modelDisplay.getGraphics();

    // if g2 isn't null
    if (g2 != null)
    {

      // save the current tranform
      AffineTransform oldTransform = g2.getTransform();

      // rotate to turtle heading and translate to xPos and yPos
      g2.rotate(Math.toRadians(heading),xPos,yPos);

      // draw the passed picture
      g2.drawImage(dropPicture.getImage(),xPos,yPos,null);

      // reset the tranformation matrix
      g2.setTransform(oldTransform);

      //  draw the pen
      pen.paintComponent(g2);
    }
  }
```

```
/**
 * Method to paint the turtle
 * @param g the graphics context to paint on
 */
public synchronized void paintComponent(Graphics g)
{
  // cast to 2d object
  Graphics2D g2 = (Graphics2D) g;

  // if the turtle is visible
  if (visible)
  {
    // save the current tranform
    AffineTransform oldTransform = g2.getTransform();

    // rotate the turtle and translate to xPos and yPos
    g2.rotate(Math.toRadians(heading),xPos,yPos);

    // determine the half width and height of the shell
    int halfWidth = (int) (width/2); // of shell
    int halfHeight = (int) (height/2); // of shell
    int quarterWidth = (int) (width/4); // of shell
    int thirdHeight = (int) (height/3); // of shell
    int thirdWidth = (int) (width/3); // of shell

    // draw the body parts (head)
    g2.setColor(bodyColor);
    g2.fillOval(xPos - quarterWidth,
                yPos - halfHeight - (int) (height/3),
                halfWidth, thirdHeight);
    g2.fillOval(xPos - (2 * thirdWidth),
                yPos - thirdHeight,
                thirdWidth,thirdHeight);
    g2.fillOval(xPos - (int) (1.6 * thirdWidth),
                yPos + thirdHeight,
                thirdWidth,thirdHeight);
    g2.fillOval(xPos + (int) (1.3 * thirdWidth),
                yPos - thirdHeight,
                thirdWidth,thirdHeight);
    g2.fillOval(xPos + (int) (0.9 * thirdWidth),
                yPos + thirdHeight,
                thirdWidth,thirdHeight);


    // draw the shell
    g2.setColor(getShellColor());
    g2.fillOval(xPos - halfWidth,
                yPos - halfHeight, width, height);

    // draw the info string if the flag is true
```

```
      if (showInfo)
        drawInfoString(g2);

      // reset the tranformation matrix
      g2.setTransform(oldTransform);
    }

    //  draw the pen
    pen.paintComponent(g);
  }

  /**
   * Method to draw the information string
   * @param g the graphics context
   */
  public synchronized void drawInfoString(Graphics g)
  {
    g.setColor(infoColor);
    g.drawString(this.toString(),xPos + (int) (width/2),yPos);
  }
  //This toString method was modified.
  /**
   * Method to return a string with informaiton
   * about this turtle
   * @return a string with information about this object
   */
  //THIS IS A MODIFICATION
  //MODIFIED toString METHOD
  public String toString()
  {
//    return this.name + " turtle at " + this.xPos + ", " +
//      this.yPos + " heading " + this.heading + ".";
    return "My name is " + turtleName + " the turtle.";
  }

} // end of class
```

-end-