```java
package beginninggraphics;

public class App {


	private containers.Column _column;

	private Drawing _drawing;


	public App() {
		javax.swing.JFrame frame = new javax.swing.JFrame();
		_column = new containers.Column();
		this.init();
		frame.getContentPane().add(_column);
		frame.pack();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
	}

	private void init() {
		_drawing = new Drawing(_column);
		_column.add(new containers.Panel());
		this.setUpButtons();
	}

	private void setUpButtons() {
		containers.Row row = new containers.Row();

		javax.swing.JButton upButton = new javax.swing.JButton("Up");
		upButton.addActionListener(new UpListener(_drawing));
		row.add(upButton);

		javax.swing.JButton downButton = new javax.swing.JButton("Down");
		downButton.addActionListener(new DownListener(_drawing));
		row.add(downButton);

		javax.swing.JButton leftButton = new javax.swing.JButton("Left");
		leftButton.addActionListener(new LeftListener(_drawing));
		row.add(leftButton);

		javax.swing.JButton rightButton = new javax.swing.JButton("Right");
		rightButton.addActionListener(new RightListener(_drawing));
		row.add(rightButton);

		_column.add(row);

		containers.Row row2 = new containers.Row();
		javax.swing.JButton rotateLeftButton = new javax.swing.JButton("Rota…Left");
		rotateLeftButton.addActionListener(new RotateLeftListener(_drawing));
		row2.add(rotateLeftButton);

		_column.add(row2);
	}



	public static void main(String[] args) {
		new App();
	}

}
```

```
private void setUpButtons() {
            containers.Row row = new containers.Row();

            javax.swing.JButton upButton = new javax.swing.JButton("Up");
            upButton.addActionListener(new UpListener(_drawing));
            row.add(upButton);

            javax.swing.JButton downButton = new javax.swing.JButton("Down");
            downButton.addActionListener(new DownListener(_drawing));
            row.add(downButton);

            javax.swing.JButton leftButton = new javax.swing.JButton("Left");
            leftButton.addActionListener(new LeftListener(_drawing));
            row.add(leftButton);

            javax.swing.JButton rightButton = new javax.swing.JButton("Right");
            rightButton.addActionListener(new RightListener(_drawing));
            row.add(rightButton);

            _column.add(row);

            containers.Row row2 = new containers.Row();
            javax.swing.JButton rotateLeftButton = new javax.swing.JButton("Rota…Left");
            rotateLeftButton.addActionListener(new RotateLeftListener(_drawing));
            row2.add(rotateLeftButton);

            _column.add(row2);















}
```

```java
package beginninggraphics;

public class Drawing {

	private graphics.DrawingCanvas _canvas;
	private graphics.Rectangle _rectangle;

	public Drawing(containers.Column column) {
		_canvas = new graphics.DrawingCanvas();
		_canvas.setColor(new graphics.colors.Yellow());
		_canvas.setDimension(new java.awt.Dimension(500,500));
		_rectangle = new graphics.Rectangle();
		_rectangle.setDimension(new java.awt.Dimension(80,80));
		_rectangle.setCenterLocation(new java.awt.Point(250,250));
		_rectangle.setColor(new graphics.colors.Black());

		_canvas.add(_rectangle);
		column.add(_canvas);
	}

	public void up() {
		Integer yCoordinate = _rectangle.getCenterLocation().y - 20;
		Integer xCoordinate = _rectangle.getCenterLocation().x;
		_rectangle.setCenterLocation(new java.awt.Point(xCoordinate, yCoordinate));
	}

	public void down() {
		Integer yCoordinate = _rectangle.getCenterLocation().y + 20;
		Integer xCoordinate = _rectangle.getCenterLocation().x;
		_rectangle.setCenterLocation(new java.awt.Point(xCoordinate, yCoordinate));

	}

	public void left() {
		Integer xCoordinate = _rectangle.getCenterLocation().x - 20;
		Integer yCoordinate = _rectangle.getCenterLocation().y;
		_rectangle.setCenterLocation(new java.awt.Point(xCoordinate, yCoordinate));

	}

	public void right() {
		Integer xCoordinate = _rectangle.getCenterLocation().x + 20;
		Integer yCoordinate = _rectangle.getCenterLocation().y;
		_rectangle.setCenterLocation(new java.awt.Point(xCoordinate, yCoordinate));
	}


	public void rotateLeft() {
		Integer rotation = _rectangle.getRotation();
		_rectangle.setRotation(rotation - 10);
	}


}
```

```
package beginninggraphics;

import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

public class SelectSquareToMoveListener implements MouseListener {




      public SelectSquareToMoveListener() {




      }

      @Override
      public void mouseClicked(MouseEvent arg0) {





      }

      @Override
      public void mouseEntered(MouseEvent arg0) {
            // TODO Auto-generated method stub

      }

      @Override
      public void mouseExited(MouseEvent arg0) {
            // TODO Auto-generated method stub

      }

      @Override
      public void mousePressed(MouseEvent arg0) {
            // TODO Auto-generated method stub

      }

      @Override
      public void mouseReleased(MouseEvent arg0) {
            // TODO Auto-generated method stub

      }

}
```

What else needed to be changed to be able to select the shape?

```java
public class ShapeHolder {

	private graphics.IColorableGraphic _currentShape;

	public ShapeHolder(graphics.IColorableGraphic shape) {
		_currentShape = shape;
	}

	public void setShape(graphics.IColorableGraphic shape) {
		_currentShape = shape;
	}

	public graphics.IColorableGraphic getShape() {
		return _currentShape;
	}

}


public class SelectionListener implements MouseListener {

	private ShapeHolder _holder;
	private graphics.IColorableGraphic _graphic;

	public SelectionListener(ShapeHolder holder, graphics.IColorableGraphic graphic) {
		_holder = holder;
		_graphic = graphic;
	}

	/**
	 * @see java.awt.event.MouseListener#mouseClicked(java.awt.event.MouseEvent)
	 * @param arg0
	 */
	@Override
	public void mouseClicked(MouseEvent arg0) {
		_holder.setShape(_graphic);
	}
}


public class TopLevel implements IBoardConstants {

	private MovingEllipse _ellipse;
	private MovingRectangle _rectangle;
	private ShapeHolder _holder;

	public TopLevel(Column column) {
		//Make drawing canvas
		graphics.DrawingCanvas canvas = new graphics.DrawingCanvas();
		canvas.setColor(new graphics.colors.Cyan());
		canvas.setDimension(BOARD_SIZE);
		column.add(canvas);

		_ellipse = new MovingEllipse();

		_holder = new ShapeHolder(_ellipse);

		_ellipse.setColor(new graphics.colors.Black());
		_ellipse.setDimension(new java.awt.Dimension(50, 156));
		_ellipse.setLocation(new Position(1, 2));
		_ellipse.addMouseListener(new SelectionListener(_holder, _ellipse));
		canvas.add(_ellipse);
```

```java
        _rectangle = new MovingRectangle();
        _rectangle.setColor(new graphics.colors.Black());
        _rectangle.setDimension(new java.awt.Dimension(SQUARE_SIZE * 2, SQUARE_SIZE * 5));
        _rectangle.setLocation(new Position(20, 40));
        _rectangle.addMouseListener(new SelectionListener(_holder, _rectangle));
        canvas.add(_rectangle);


        //Create buttons
        this.createButtons(column);
    }

    private void createButtons(Column column) {
        javax.swing.JButton button = new javax.swing.JButton("North");
        button.addActionListener(new MoveAction(_holder, Movement.NORTH));
        column.add(button);

        javax.swing.JButton button2 = new javax.swing.JButton("South");
        button2.addActionListener(new MoveAction(_holder, Movement.SOUTH));
        column.add(button2);

        javax.swing.JButton button3 = new javax.swing.JButton("East");
        button3.addActionListener(new MoveAction(_holder, Movement.EAST));
        column.add(button3);

        javax.swing.JButton button4 = new javax.swing.JButton("West");
        button4.addActionListener(new MoveAction(_holder, Movement.WEST));
        column.add(button4);
    }

}




public interface IBoardConstants {

    public static final Integer NUM_ROWS = 50;
    public static final Integer NUM_COLS = 50;
    public static final Integer SQUARE_SIZE = 10;
    public static final java.awt.Dimension BOARD_SIZE = new
                          java.awt.Dimension(NUM_ROWS*SQUARE_SIZE, NUM_COLS*SQUARE_SIZE);


}
```

```java
public class MoveAction implements ActionListener {

	private ShapeHolder _holder;
	private Movement _direction;

	public MoveAction(ShapeHolder holder, Movement direction) {
		_holder = holder;
		_direction = direction;
	}

	/**
	 * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
	 * @param arg0
	 */
	@Override
	public void actionPerformed(ActionEvent arg0) {
		graphics.IGraphic shape = _holder.getShape();
		Position position = new Position(shape.getLocation());
		Integer row = position.getRow() + _direction.deltaRows();
		Integer col = position.getCol() + _direction.deltaColumns();
		position = new Position(row, col);
		shape.setLocation(position.toPoint());
	}

}
```

```java
public enum Movement implements IBoardConstants {
	/**
	 * Movement to the north (up)
	 */
	NORTH (0, -1, 1, null),
	/**
	 * Movement to the south (down)
	 */
	SOUTH  (0, 1, NUM_COLS - 2, null),
	/**
	 * Movement to the east (right)
	 */
	EAST (1, 0, null, NUM_ROWS - 2),
	/**
	 * Movement to the west (left)
	 */
	WEST (-1, 0, null, 1);

	private final Integer _xMovement, _yMovement;
	private final Integer _rowCheck, _columnCheck;

	/**
	 * Creates a new instance of Movement
	 * @param xMov the increment of the movement in the x direction
	 * @param yMov the increment of the movement in the y direction
	 * @param rowCheck1 the number that the row would need to be to tell us we were at the edge
	 *             (null if not needed for that direction)
	 * @param colCheck the number that the column would need to be to tell us we were at the edge
	 *             (null if not needed for that direction)
	 */
	Movement(Integer xMov, Integer yMov, Integer rowCheck1, Integer colCheck) {
		this._xMovement = xMov;
		this._yMovement = yMov;
		this._rowCheck = rowCheck1;
		this._columnCheck = colCheck;
	}
	/**
	 * Returns the change in rows for this movement
	 */
	public Integer deltaRows()   { return _yMovement; }
	/**
	 * Returns the change in columns for this movement
	 */
	public Integer deltaColumns()   { return _xMovement; }
	/**
	 * Returns the change in rows if moving two rows away
	 */
	public Integer deltaTwoRows() { return _yMovement * 2; }
	/**
	 * Returns the change in columns if moving two columns away
	 */
	public Integer deltaTwoColumns() { return _xMovement * 2; }

	/**
	 * Tells whether we are at the edge based on the position passed in.  For example, if
	 * we are NORTH, we are interested in the north (or top) edge, EAST is right edge, etc.
	 * @param position the position to be tested for edge conditions
	 * @return true if at the edge, false otherwise
	 */
	public boolean atEdge(Position position) {
		//We either care about rows or columns when considering
		//boundary conditions.  If we have a null for a rowCheck, then
		//we must be concerned with columns.  This assumes that the
		//enumerands were created in this way.
		if(_rowCheck != null) {
			return position.getRow() == _rowCheck;
		}
		return position.getCol() == _columnCheck;
	}
}
```