**SlingshotApp.java**

```java
/* Name: Richard Eisenberg
 * File: SlingshotApp.java
 * Desc: The slingshot app with number controls
 */
import acm.graphics.*;
import acm.program.*;
import java.awt.event.*;
import acm.util.*;

public class SlingshotApp extends GraphicsProgram
{
  private Slingshot shot; // the rising and falling shot

  // controls for x and y velocities
  private NumberControl xControl;
  private NumberControl yControl;

  // label for firing
  private GLabel fire;

  @Override
  public void run()
  {
    // create shot:
    shot = new Slingshot();
    add(shot);

    xControl = new NumberControl();
    xControl.setLocation(15, 10);
    add(xControl);

    yControl = new NumberControl();
    yControl.setLocation(15, 20);
    add(yControl);

    xControl.setRange(0, 10);
    yControl.setRange(0, 10);

    xControl.setNumber(1.5);
    yControl.setNumber(5);

    GLabel x = new GLabel("x: ", 5, 10);
    GLabel y = new GLabel("y: ", 5, 20);

    add(x);
    add(y);

    fire = new GLabel("Fire!", 5, 35);
    add(fire);

    // enable mouse and timer
    addMouseListeners();

    SwingTimer t = new SwingTimer(25, this);
    t.start();
  }

  @Override
  public void mousePressed(MouseEvent e)
  {
    xControl.click(e.getX(), e.getY());
    yControl.click(e.getX(), e.getY());

    // fire when ready
    if(fire.contains(e.getX(), e.getY()))
    {
      shot.fire(xControl.getNumber(), yControl.getNumber());
    }
  }

  @Override
  public void actionPerformed(ActionEvent e)
```

**SlingshotApp.java**

```java
  {
    shot.update(); // the shot keeps track of whether it’s moving or not
  }
}
```

**Slingshot.java**

```java
/* Name: Richard Eisenberg
 * File: Slingshot.java
 * Desc: slingshot compound object
 */
import acm.graphics.*;

public class Slingshot extends GCompound
{
  // x- and y-components of velocity
  private double xVel;
  private double yVel;

  // are we moving?
  private boolean moving;

  public Slingshot()
  {
    GOval shot = new GOval(0, 0, 5, 5);
    shot.setFilled(true);
    add(shot);

    moving = false; // start out not moving

    // position us appropriately:
    setLocation(5, 190);
  }

  // parameters are initial x- and y-velocities
  public void fire(double x, double y)
  {
    // only fire if we're not moving
    if(moving == false)
    {
      // set the initial velocities as per parameters
      xVel = x;
      yVel = -y; // y has to be negative so that the shot goes up

      // reset to bottom left
      setLocation(5, 190); // use just plain setLocation because we want to move the traci
ng paper

      moving = true;
    }
  }

  // move the ball if appropriate
  public void update()
  {
    if(moving == true)
    {
      move(xVel, yVel);

      // the force of gravity increases yVel
      yVel = yVel + 0.1;

      // when we're past the bottom, stop.
      if(getY() + getHeight() > 200)
      {
        moving = false;
      }
    }
  }
}
```

**NumberControl.java**

```java
/* Name: Richard Eisenberg
 * File: NumberControl.java
 * Desc: simple number control made up of GLabels
 */
import acm.graphics.*;

public class NumberControl extends GCompound
{
  // the buttons and display
  private GLabel left;
  private GLabel number;
  private GLabel right;

  private double num; // the number in the number control

  // ends of allowable range
  private double minimum;
  private double maximum;

  public NumberControl()
  {
    num = 0;

    // default range is 0 to 10:
    minimum = 0;
    maximum = 10;

    left = new GLabel("<", 0, 0);
    number = new GLabel("0", 10, 0);
    right = new GLabel(">", 30, 0);

    add(left);
    add(number);
    add(right);
  }

  // sets the number in the control
  public void setNumber(double newNum)
  {
    num = newNum;

    checkRange(); // make sure we're in the appropriate range
    updateDisplay(); // show the user
  }

  // the coordinates are applet coordinates and will need to be translated
  public void click(double x, double y)
  {
    if(left.contains(x - getX(), y - getY()))
    {
      num = num - 0.5;
    }

    if(right.contains(x - getX(), y - getY()))
    {
      num = num + 0.5;
    }

    checkRange();
    updateDisplay();
  }

  // get the current number
  public double getNumber()
  {
    return num;
  }

  // allow the applet to set the allowable range
  public void setRange(double min, double max)
  {
    // store in fields
```

**NumberControl.java**

```
    minimum = min;
    maximum = max;

    checkRange(); // maybe the current number is out of range and we need to update
    updateDisplay();
  }

  // ensure we're in the range
  public void checkRange()
  {
    if(num > maximum)
    {
      num = maximum;
    }

    if(num < minimum)
    {
      num = minimum;
    }
  }

  // update the label
  public void updateDisplay()
  {
    number.setLabel("" + num);
  }
}
```

**Colors2.java**

```java
/* Name: Richard Eisenberg
   File: Colors2.java
   Desc: Changes colors based on keystrokes
*/

import acm.program.*;
import acm.graphics.*;
import java.awt.event.*;
import java.awt.*;

/* This applet starts out blue.  It turns blue when you press the left arrow
and turns red when you press the right arrow */
public class Colors2 extends GraphicsProgram
{
  private GRect background; // the background rectangle

  @Override
  public void run()
  {
    background = new GRect(0, 0, 200, 200);
    background.setFilled(true);
    add(background);

    addKeyListeners(); // we’re responding to keys, so addKeyListeners()
  }

  /* handles keystrokes */
  @Override
  public void keyPressed(KeyEvent e)
  {
    if(e.getKeyCode() == KeyEvent.VK_LEFT)
    {
      // turn blue if we have pressed left
      background.setFillColor(Color.BLUE);
    }
    else if(e.getKeyCode() == KeyEvent.VK_RIGHT)
    {
      // turn red if we have pressed right
      background.setFillColor(Color.RED);
    }
  }

  /* There also exists a method named keyReleased, which gets called
   * whenever the user releases a key.  It has the exact same structure
   * as keyPressed. */
}
```

**MovingSquare.java**

```java
/* Name: Richard Eisenberg
 * File: MovingSquare.java
 * Desc: Example of how the keys can move a square around the app
 */

import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class MovingSquare extends GraphicsProgram
{
  private GRect square; // our square

  @Override
  public void run()
  {
    square = new GRect(90, 90, 20, 20);
    square.setFilled(true);
    square.setColor(Color.BLUE);
    add(square);

    addKeyListeners(); // respond to keystrokes
  }

  @Override
  public void keyPressed(KeyEvent e)
  {
    // just check each direction one at a time
    if(e.getKeyCode() == KeyEvent.VK_LEFT)
    {
      square.move(-5, 0);
    }
    else if(e.getKeyCode() == KeyEvent.VK_UP)
    {
      square.move(0, -5);
    }
    else if(e.getKeyCode() == KeyEvent.VK_RIGHT)
    {
      square.move(5, 0);
    }
    else if(e.getKeyCode() == KeyEvent.VK_DOWN)
    {
      square.move(0, 5);
    }
  }
}
```