```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.ImageIcon;

public class MoveSprite implements KeyListener, MouseListener
{

    public static void main(String args[])
    {
        MoveSprite test = new MoveSprite();
        test.run();
    }

    private ScreenManager screen;
    private Image bgImage;
    private Sprite sprite;
    private boolean stop;

    private boolean paused=false;
    private double xscale = 1.0;
    private double yscale = 1.0;

    public void loadImages()
    {
        // Load images
        bgImage = loadImage("images/background.jpg");
        Image player1 = loadImage("images/smplayer1.png");
        Image player2 = loadImage("images/smplayer2.png");
        Image player3 = loadImage("images/smplayer3.png");

        // Load Animation
        Animation anim = new Animation();
        anim.addFrame(player1, 250);
        anim.addFrame(player2, 150);
        anim.addFrame(player1, 150);
        anim.addFrame(player2, 150);
        anim.addFrame(player3, 200);
        anim.addFrame(player2, 150);

        // Create Sprite
        sprite = new Sprite(anim);
        sprite.setVelocityX(0.0f);
        sprite.setVelocityY(0.0f);
    }

    public void run() {
        screen = new ScreenManager();

        try {
            DisplayMode displayMode = new DisplayMode(1024,768,16,60);
            screen.setFullScreen(displayMode);

            Window win = screen.getFullScreenWindow();

            // Bug fix for Java 1.7 on OSX
            // failing to register window for events
            win.setVisible(false);
            win.setVisible(true);

            win.addKeyListener(this);
            win.addMouseListener(this);

            loadImages();
            animationLoop();
        }
        finally {
            screen.restoreScreen();
        }
    }
```

```
public void animationLoop() {

    long currTime = System.currentTimeMillis();

    stop = false;

    while (!stop) {
        long elapsedTime = System.currentTimeMillis() - currTime;
        currTime += elapsedTime;

        // update the sprites
        if (!paused)  update(elapsedTime);

        // draw and update the screen
        Graphics2D g = screen.getGraphics();
        draw(g);
        g.dispose();
        screen.update();

        // take a nap
        try
        {
            Thread.sleep(20);
        }
        catch (InterruptedException ex) { }
    }
}


public void update(long elapsedTime) {
    // check sprite bounds
    if (sprite.getX() < 0) {
        sprite.setVelocityX(Math.abs(sprite.getVelocityX()));
    }
    else if (sprite.getX() + sprite.getWidth() >= screen.getWidth())
    {
        sprite.setVelocityX(-Math.abs(sprite.getVelocityX()));
    }
    if (sprite.getY() < 0) {
        sprite.setVelocityY(Math.abs(sprite.getVelocityY()));
    }
    else if (sprite.getY() + sprite.getHeight() >= screen.getHeight())
    {
        sprite.setVelocityY(-Math.abs(sprite.getVelocityY()));
    }

    xscale = Math.abs(xscale);
    if (sprite.getVelocityX() < 0) xscale = -xscale;

    // update sprite
    sprite.update(elapsedTime);
}


public void draw(Graphics g)
{
    // Draw the background
    g.drawImage(bgImage, 0, 0, null);

    // Declare and define the transform
    AffineTransform transform = new AffineTransform();
    transform.translate(Math.round(sprite.getX()),Math.round(sprite.getY()));
    transform.scale(xscale,yscale);

    // Get a reference to a Graphics 2D object and apply the transform
    Graphics2D g2d = (Graphics2D)g;
    g2d.drawImage(sprite.getImage(),transform,null);
}

private Image loadImage(String fileName)
{
    return new ImageIcon(fileName).getImage();
```

```
    }


    public void keyPressed(KeyEvent e) {
        int keyCode = e.getKeyCode();

        // exit the program
        if (keyCode == KeyEvent.VK_ESCAPE) stop = true;

        if (keyCode == KeyEvent.VK_LEFT)
            sprite.setVelocityX(sprite.getVelocityX()-0.1f);

        if (keyCode == KeyEvent.VK_RIGHT)
            sprite.setVelocityX(sprite.getVelocityX()+0.1f);

        if (keyCode == KeyEvent.VK_UP)
            sprite.setVelocityY(sprite.getVelocityY()-0.1f);

        if (keyCode == KeyEvent.VK_DOWN)
            sprite.setVelocityY(sprite.getVelocityY()+0.1f);

        if (keyCode == KeyEvent.VK_P) paused = !paused;

        if (keyCode == KeyEvent.VK_S) yscale = yscale * 2.0f;
        if (keyCode == KeyEvent.VK_W) yscale = yscale / 2.0f;

        if (keyCode == KeyEvent.VK_D) xscale = xscale * 2.0f;
        if (keyCode == KeyEvent.VK_A) xscale = xscale / 2.0f;


        e.consume();
    }


    public void keyReleased(KeyEvent e) { e.consume(); }

    public void keyTyped(KeyEvent e) { e.consume(); }

    // The following methods are from the MouseListener interface
    // We are required to implement them since we said we implemented
    // the MouseListener interface. Empty methods will do for most of them
    public void mousePressed(MouseEvent e) { }

    public void mouseReleased(MouseEvent e) { }

    public void mouseClicked(MouseEvent e)
    {
        sprite.setX(e.getX());
        sprite.setY(e.getY());
    }

    public void mouseEntered(MouseEvent e) { }

    public void mouseExited(MouseEvent e) { }



}
```