```
package gameutil;

import java.util.*;
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;

public abstract class GameComponent extends JPanel
{
    public static int WIDTH, HEIGHT;
    protected BufferedImage background = null;
    public int delay = 25;

    /**
     * Constructs a GameComponent with a width of w, and a height of h.
     *
     * @param  w  width
     * @param  h  height
     * @see         JPanel
     */
    public GameComponent(int w, int h)
    {
        super();
        WIDTH = w;
        HEIGHT = h;
        setSize(WIDTH, HEIGHT);
        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        setBackground(Color.WHITE);
        setVisible(true);
        Thread t = new Thread()
        {
            public void run()
            {
                while(true)
                {
                    long time = System.currentTimeMillis();
                    if(background == null)
                    {
                        background = new BufferedImage(WIDTH, HEIGHT,
                        BufferedImage.TYPE_INT_RGB);
                        background.getGraphics().setColor(Color.WHITE);
                        background.getGraphics().fillRect(0,0,WIDTH,HEIGHT);
                    }

                    requestFocus();

                    //update game state
                    standardUpdates();
                    update();

                    //draw stuff
                    standardDraw(getCanvas());
                    draw(background.getGraphics());
                    refreshImage();

                    time = System.currentTimeMillis()-time;
                    try
                    {
                        if(delay-(int)time > 0)
                            sleep(delay-(int)time);
                    }
                    catch(Exception ex)
                    {
                    }
                }
            }
        };
```

```
        try{Thread.sleep(500);}catch(Exception ex){}
        t.start();
    }

    //get a blank image to draw onto
    private Graphics getCanvas()
    {
        if(background == null)
        {
            background = new BufferedImage(WIDTH, HEIGHT, BufferedImage.
            TYPE_INT_RGB);
        }
        background.getGraphics().setColor(Color.WHITE);
        background.getGraphics().fillRect(0,0,WIDTH,HEIGHT);
        return background.getGraphics();
    }

    //take the canvas that you have drawn on and draw it onto the component
    private void refreshImage()
    {
        if(background != null)
        {
            if(getGraphics() != null)
            {
                getGraphics().drawImage(background,0,0,null);
            }
        }
    }

    /**
     * Creates a {@link JFrame} that contains this GameComponent.
     *
     * @return  the {@link JFrame} created
     */
    public JFrame makeTestWindow()
    {
        JFrame frame = new JFrame();
        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(this);
        frame.pack();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        return frame;
    }

    /**
     * Creates a fullscreen {@link JFrame} that contains this GameComponent.
     * <br>*Note that the width and height of the component must be 640x480
     *
     * @return  the {@link JFrame} created
     */
    public JFrame makeFullScreenWindow()
    {
        JFrame frame = new JFrame();
        GraphicsDevice device = GraphicsEnvironment.
        getLocalGraphicsEnvironment().getDefaultScreenDevice();
        GraphicsConfiguration gc = device.getDefaultConfiguration();
        DisplayMode oldDisplayMode = device.getDisplayMode();
        DisplayMode newDisplayMode = new DisplayMode(640, 480,
        (oldDisplayMode.getBitDepth()), (oldDisplayMode.getRefreshRate()));
        frame.getContentPane().setLayout(null);
        frame.getContentPane().add(this, 0, 0);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(WIDTH, HEIGHT);
        frame.setResizable(false);
        frame.setUndecorated(true);
        frame.setVisible(true);
```

```
        if(device.isFullScreenSupported())
        {
            device.setFullScreenWindow(frame);
            device.setDisplayMode(newDisplayMode);
        }
        else
        {
            System.out.println("ARGS! NO FULLSCRENE!");
        }
        return frame;
    }

    /**
     * Preforms the standard updates of the component.(Preformed befor {@link
     #update()})
     */
    public void standardUpdates()
    {
    }

    /**
     * The method that draws the component.
     *
     * @param   g   the {@link Graphics} on which the component will be drawn
     */
    public abstract void draw(Graphics g);

    /**
     * Draws the sandard parts of the component. (Preformed befor {@link
     #draw(Graphics)})
     *
     * @param   g   the {@link Graphics} on which the component will be drawn
     */
    public void standardDraw(Graphics g)
    {
    }


    /**
     * The method that updates the component.
     */
    public abstract void update();
}
```