```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.border.*;

public class MidiHeroGUI
{
	public MidiHeroGUI()
	{
		Game game = new Game(800, 500, "sample");
		game.start();
   }

   public static void main(String[] args)
   {
		MidiHeroGUI gui = new MidiHeroGUI();
	}
}
```

```
import java.awt.image.BufferStrategy;
import java.awt.image.BufferedImage;
import java.awt.Graphics;
import java.io.File;
import javax.imageio.ImageIO;
import java.io.IOException;
import javax.swing.JOptionPane;
import java.awt.Color;
import java.awt.Font;

public class Game implements Runnable
{
	private Display display;
	public int width, height, highscore;
	public String title, song;
	private boolean isPlaying;
	private Thread thread;
	private ArrowLogic logic;
	private Arrow arrow;
	private Marker marker;
	private PlayMusic music;
	private int score;
	private BufferStrategy bs;
	private Graphics g;

	public Game(int width, int height, String song)
	{
		isPlaying = false;
```

```
        this.width = width;
        this.height = height;
        arrow = new Arrow();
        marker = new Marker();
        music = new PlayMusic();
        score = highscore = 0;
        this.song = "Songs/" + song + ".WAV";
}

private void init()
{
        display = new Display(width, height);
        display.getFrame().addKeyListener(arrow);
        display.getFrame().addKeyListener(marker);
}

public Display getDisplay()
{
        return display;
}

public void tick()
{
        arrow.tick();
        render();
        gameLost();
}
```

```
public void render()
{
	//canvas
	bs = display.getCanvas().getBufferStrategy();
	if(bs == null)
	{
		display.getCanvas().createBufferStrategy(3);
		return;
	}
	g = bs.getDrawGraphics();

	g.setColor(Color.WHITE);
	g.setFont(new Font("Serif", Font.BOLD, 20));

	marker.render(g);
	arrow.render(g);

	//displays image(buffered image)
	bs.show();
	g.dispose();
}

public void run()
{
	init();

	//timer with frames
	final double fps = 60;
```

```
        double timePerTick = 1000000000 / fps;
        double delta = 0;
        long now;
        long lastTime = System.nanoTime();

        while(isPlaying)
        {
                now = System.nanoTime();
                delta +=(now - lastTime) / timePerTick;
                lastTime = now;
                if(delta >= 1)
                {
                        tick();
                        render();
                        delta--;
                }
        }
        stop();
}

/*START THREAD*/
public synchronized void start()
{
        if(isPlaying)
                return;
        isPlaying = true;
        thread = new Thread(this);
        thread.start();
```

```
            music.startSong(song);
        }

        /*STOP THREAD SAFELY*/
        public synchronized void stop()
        {
            if(!isPlaying)
                return;
            isPlaying = false;
            try{
                thread.join();
            }catch (InterruptedException e) {
                e.printStackTrace();
            }

        }

        public void gameLost()
        {
            if(arrow.checkWon() == false)
            {
                isPlaying = false;
                music.stopSong();
                display.youLose();
            }
        }
}
```

```
import javax.swing.JFrame;
import java.awt.Canvas;
import java.awt.Dimension;
import java.awt.image.BufferedImage;
import javax.swing.JOptionPane;

public class Display
{
	public JFrame frame;
	private Canvas canvas;
	private String title;
	private int width, height;

	public Display(int width, int height)
	{
		title = "Midi Hero";
		this.width = width;
		this.height = height;

		createDisplay();
	}

	private void createDisplay()
	{
		//Creates frame
		frame = new JFrame(title);
		frame.setSize(width, height);
		frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
```

```
        frame.setResizable(false);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);

        //Creates canvas
        canvas = new Canvas();
        canvas.setPreferredSize(new Dimension(width, height));
        canvas.setMaximumSize(new Dimension(width, height));
        canvas.setMinimumSize(new Dimension(width, height));
        canvas.setFocusable(false);

        //adds canvas to frame
        frame.add(canvas);
        frame.pack();
}

public Canvas getCanvas()
{
        return canvas;
}

public JFrame getFrame()
{
        return frame;
}

public void youLose()
{
```

```
                JOptionPane.showMessageDialog(frame, "You Have lost!\nPress OK to restart!", "You
Lose!", JOptionPane.ERROR_MESSAGE);
                frame.setVisible(true);
                restart();
        }

        public void restart()
        {
                frame.dispose();
                Game game = new Game(800, 500, "Songs/sample.WAV");
                game.start();
        }
}
```

```
import java.awt.event.KeyEvent;
import javax.swing.ImageIcon;
import java.awt.image.BufferedImage;
import java.lang.*;
import java.awt.Graphics;
import javax.imageio.ImageIO;
import java.io.*;
import java.awt.Rectangle;
import java.util.Scanner;
import java.awt.*;
import javax.swing.*;
import java.io.FileReader;

public class ArrowLogic extends ImageChooser
{
	private int spot = 0;
	public int global = -1;
	private int length;
	private int min = 0;
	public String filename;
	public int[][] keys;
	public int[] keysY, side, keyTime;

   public ArrowLogic()
   {
		keys = new int[getLength()][2];
		keysY = new int[getLength()];
		side = new int[getLength()];
```

```
                keyTime = new int[getLength()];
                readKeys();
    }

    public int getLength()
    {
                filename = "Song-keys/sample.txt";
                try
                {
                        LineNumberReader size = new LineNumberReader(new FileReader(new
File(filename)));
                        Scanner file = new Scanner(new File(filename));
                        size.skip(Long.MAX_VALUE);
                        length = (size.getLineNumber()/2)+1;
                        size.close(); //prevent resource leak
                }
                catch(Exception e){
                        e.printStackTrace();
                }
                return length;
        }

    public void readKeys()
    {
                filename = "Song-keys/sample.txt";
                try
                {
```

```
                    LineNumberReader size = new LineNumberReader(new FileReader(new
File(filename)));
                    Scanner file = new Scanner(new File(filename));
                    size.skip(Long.MAX_VALUE);
                    length = (size.getLineNumber()/2)+1;

                    for(int i = 0; i < length-2; i++)
                    {
                        keys[i][0] = file.nextInt();
                        keys[i][1] = file.nextInt();
                        side[i] = keys[i][0];
                        keyTime[i] = keys[i][1];
                        keysY[i] = 600;
                        System.out.println(keys[i][0]);
                        System.out.println(keys[i][1]);
                    }
                    size.close(); //prevent resource leak
                }
                catch(Exception e){
                    e.printStackTrace();
                }
        }

        public void placeKeys()
        {
            if(setTime(keyTime[spot]) == true)
                spot++;
        }
```

```
public boolean setTime(int tick)
{
	if(tick == global)
		return true;
	return false;
}

public void setSpot()
{
	for(int i = min; i < spot; i++)
	{
		if(keysY[i] <= -3) //decreases size of forloop
		{
			keysY[i] = -100;
			min++;
		}
		else
			keysY[i] += -3;
		//System.out.println("**"+keysY[i]+"**");
	}
}

public boolean checkWon()
{
	if(lives < 0)
		return false;
	return true;
}
```

```
public int getKeysY(int i)
{
	int num = keysY[i];
	return num;
}

public int getSide(int i)
{
	int direction = side[i];
	return direction;
}

public int getSpot()
{
	return spot;
}

public int getMin()
{
	return min;
}

//checks location and uses timer
public void tick()
{
	global++;
	System.out.println("Global: " + global);
```

```
                placeKeys();
                setSpot();
        }
}
```

```
import java.awt.event.KeyEvent;
import javax.swing.ImageIcon;
import java.awt.image.BufferedImage;
import java.lang.*;
import java.awt.Graphics;
import javax.imageio.ImageIO;
import java.io.*;
import java.awt.Rectangle;
import java.util.Scanner;
import java.awt.*;
import javax.swing.*;
import java.io.FileReader;
import java.awt.event.KeyListener;

public class Arrow extends ArrowLogic implements KeyListener
{
	private int spot, i, x;
	private BufferedImage imgLeft, imgRight, imgDown, imgUp, imgBar;
	private boolean upPressed, downPressed, leftPressed, rightPressed, spacePressed;
	public boolean dev = true;
	private SongCreator sc;

   public Arrow(String song)
   {
		try {
			imgLeft = ImageIO.read(new File("pictures/arrows/leftArrow.PNG"));
			imgRight = ImageIO.read(new File("pictures/arrows/rightArrow.PNG"));
			imgUp = ImageIO.read(new File("pictures/arrows/upArrow.PNG"));
```

```
                imgDown = ImageIO.read(new File("pictures/arrows/downArrow.PNG"));
                imgBar = ImageIO.read(new File("pictures/arrows/barArrows.PNG"));
            }catch (IOException e) {

                e.printStackTrace();
                System.exit(1);
            }

            x = 200;
}

    public void tick()
    {
        super.tick();

        i = getMin();
        if(keysY[i] < 0 && keysY[i] > -10)
        {
            keysY[i] = -100;
            missNote();
            setMultiplyer();
        }
    }

    public void render(Graphics g)
    {
        super.render(g);
        spot = getSpot();
```

```
                int min = getMin();
                for(int i = min; i < spot; i++)
                {
                        System.out.println("\n*****Side: " + side[i]);
                        System.out.println("***KeysY: " + keysY[i]);

                        if(side[i] == 0)
                                g.drawImage(imgLeft,x,keysY[i],65,65,null);
                        else if(side[i] == 1)
                                g.drawImage(imgUp,x+70,keysY[i],65,65,null);
                        else if(side[i] == 2)
                                g.drawImage(imgDown,x+140,keysY[i],65,65,null);
                        else if(side[i] == 3)
                                g.drawImage(imgRight,x+210,keysY[i],65,65,null);
                        else if(side[i] == 4)
                                g.drawImage(imgBar,x,keysY[i],null);
                }
        }

public void keyPressed(KeyEvent e)
{
                i = getMin();

                if(e.getKeyCode() == KeyEvent.VK_A)
                        leftPressed = true;
                if(e.getKeyCode() == KeyEvent.VK_W)
                        upPressed = true;
                if(e.getKeyCode() == KeyEvent.VK_D)
```

```
            rightPressed = true;
    if(e.getKeyCode() == KeyEvent.VK_S)
            downPressed = true;
    if(e.getKeyCode() == KeyEvent.VK_SPACE)
            spacePressed = true;



    if(side[i] == 0 && leftPressed == true)
    {
            if((keysY[i] > 0) && (keysY[i] < 60))
            {
                    if(spacePressed == true)
                    {
                            setStreak();
                            setScore();
                            keysY[i] = -100;
                            spacePressed = false;

                            if(dev == true)
                                    sc = new SongCreator(side[i],global,filename+"new");
                    }
            }
    }
    if(side[i] == 1 && upPressed == true)
    {
            if((keysY[i] > 0) && (keysY[i] < 60))
            {
                    if(spacePressed == true)
```

```
                {
                        setStreak();
                        setScore();
                        keysY[i] = -100;
                        spacePressed = false;
                }
        }
}
if(side[i] == 2 && downPressed == true)
{
        if((keysY[i] > 0) && (keysY[i] < 60))
        {
                if(spacePressed == true)
                {
                        setStreak();
                        setScore();
                        keysY[i] = -100;
                        spacePressed = false;
                }
        }
}
if(side[i] == 3 && rightPressed == true)
{
        if((keysY[i] > 0) && (keysY[i] < 60))
        {
                if(spacePressed == true)
                {
                        setStreak();
```

```
                                        setScore();
                                        keysY[i] = -100;
                                        spacePressed = false;
                                }
                        }
                }
                if(side[i] == 4)
                {
                        if((keysY[i] > 0) && (keysY[i] < 60))
                        {
                                if(spacePressed == true)
                                {
                                        setStreak();
                                        setScore();
                                        keysY[i] = -100;
                                        spacePressed = false;
                                }
                        }
                }

                System.out.println(getKeysY(i));
}

public void keyReleased(KeyEvent e)
{
                if(e.getKeyCode() == KeyEvent.VK_W)
                        upPressed = false;
```

```
        if(e.getKeyCode() == KeyEvent.VK_S)
                                downPressed = false;

        if(e.getKeyCode() == KeyEvent.VK_A)
                                leftPressed = false;

        if(e.getKeyCode() == KeyEvent.VK_D)
                                rightPressed = false;

        if(e.getKeyCode() == KeyEvent.VK_SPACE)
            spacePressed = false;
    }
    public void keyTyped(KeyEvent e)
            {

            }
}
```

```
import javax.swing.ImageIcon;
import java.awt.image.BufferedImage;
import java.io.File;
import java.lang.*;
import java.awt.*;
import java.awt.Graphics;
import javax.imageio.ImageIO;
import java.io.IOException;
import javax.swing.*;
import java.awt.Rectangle;

public class ImageChooser extends MakeScore
{
	private int y, x;
	private BufferedImage imgBoard0, imgBoard1, imgBoard2, imgBoard3, imgBoard4,
				imgBoard5, imgBoard6, imgBoard7, imgBoard8, imgHealth, imgBar;

  public ImageChooser()
  {
		try {
			imgBoard0 = ImageIO.read(new File("pictures/scoreboard/board00.PNG"));
			imgBoard1 = ImageIO.read(new File("pictures/scoreboard/board01.PNG"));
			imgBoard2 = ImageIO.read(new File("pictures/scoreboard/board02.PNG"));
			imgBoard3 = ImageIO.read(new File("pictures/scoreboard/board03.PNG"));
			imgBoard4 = ImageIO.read(new File("pictures/scoreboard/board04.PNG"));
			imgBoard5 = ImageIO.read(new File("pictures/scoreboard/board05.PNG"));
			imgBoard6 = ImageIO.read(new File("pictures/scoreboard/board06.PNG"));
			imgBoard7 = ImageIO.read(new File("pictures/scoreboard/board07.PNG"));
```

```
                imgBoard8 = ImageIO.read(new File("pictures/scoreboard/board08.PNG"));
                imgHealth = ImageIO.read(new File("pictures/scoreboard/health.PNG"));
                imgBar = ImageIO.read(new File("pictures/scoreboard/line.PNG"));

            }catch (IOException e) {

                e.printStackTrace();
                System.exit(1);
            }
}

    public void render(Graphics g)
    {
        g.drawImage(imgHealth, 525, 400, 275, 100, null);
        if(lives > 1 && lives < 275)
            g.drawImage(imgBar, 525 + (int)lives, 410, 6, 75, null);
        else if(lives < 1)
            g.drawImage(imgBar, 525, 410, 6, 75, null);
        else if(lives >= 275)
            g.drawImage(imgBar, 525 + 275, 410, 6, 75, null);


        if(streak == 0 || streak == 9 || streak == 18 || streak == 27)
            g.drawImage(imgBoard0, 523, 3, 275, 275, null);
        else if(streak == 1 || streak == 10 || streak == 19 || streak == 28)
            g.drawImage(imgBoard1, 523, 3, 275, 275, null);
        else if(streak == 2 || streak == 11 || streak == 20 || streak == 29)
            g.drawImage(imgBoard2, 523, 3, 275, 275, null);
```

```
else if(streak == 3 || streak == 12 || streak == 21 || streak == 30)
        g.drawImage(imgBoard3, 523, 3, 275, 275, null);
else if(streak == 4 || streak == 13 || streak == 22 || streak == 31)
        g.drawImage(imgBoard4, 523, 3, 275, 275, null);
else if(streak == 5 || streak == 14 || streak == 23 || streak == 32)
        g.drawImage(imgBoard5, 523, 3, 275, 275, null);
else if(streak == 6 || streak == 15 || streak == 24 || streak == 33)
        g.drawImage(imgBoard6, 523, 3, 275, 275, null);
else if(streak == 7 || streak == 16 || streak == 25 || streak == 34)
        g.drawImage(imgBoard7, 523, 3, 275, 275, null);
else if(streak == 8 || streak == 17 || streak == 26 || streak >= 35)
        g.drawImage(imgBoard8, 523, 3, 275, 275, null);

g.drawString("Score: " + score, 10, 50);
g.drawString("Streak: " + streak, 10, 80);
g.drawString("Multiplyer: " + multiplyer, 10, 110);
g.drawString("Lives: " + lives, 10, 130);

g.setFont(new Font("Impact", Font.PLAIN, 100));
g.drawString("" + multiplyer, 650,198);
g.setFont(new Font("Impact", Font.PLAIN, 35));
g.setColor(Color.BLACK);
g.drawString("Streak: " + streak, 550,270);
g.setFont(new Font("Impact", Font.PLAIN, 45));
g.setColor(Color.WHITE);

if(score < 10)
        g.drawString(""+score, 770,50);
```

```
                else if(score < 100)
                        g.drawString(""+score, 750,50);
                else if(score < 1000)
                        g.drawString(""+score, 730,50);
                else if(score < 10000)
                        g.drawString(""+score, 710,50);
                else if(score < 100000)
                        g.drawString(""+score, 690,50);
                else if(score < 1000000)
                        g.drawString(""+score, 670,50);
                else if(score < 10000000)
                        g.drawString(""+score, 650,50);
        }
}
```

```
import java.io.*;
import java.util.Scanner;
import java.awt.*;
import javax.swing.*;

public class MakeScore
{
	public int score, streak, multiplyer;
	public double lives;

   public MakeScore()
   {
		score = streak = 0;
		multiplyer = 1;
		lives = 137.5;
   }

   public void setStreak()
   {
		streak++;
		setMultiplyer();
	}

	public void setScore()
	{
		score += multiplyer;
		if(lives >= 272.25)
			lives = 275;
```

```
        else
                lives += 2.75;
}

public void missNote()
{
        streak = 0;
        if(lives < 11 && lives > 0)
                lives += -lives;
        else if(lives == 0.0)
                lives += -1;
        else
                lives += -11;
}

public void setMultiplyer()
{
        if(streak < 8)
                multiplyer = 1;
        else if(streak < 17)
                multiplyer = 2;
        else if(streak < 26)
                multiplyer = 3;
        else if(streak > 26)
                multiplyer = 4;
}

public int getScore()
```

```
	{
		return score;
	}

	public int getStreak()
	{
		return streak;
	}

	public int getMultiplyer()
	{
		return multiplyer;
	}
}
```

```
import javax.swing.ImageIcon;
import java.awt.image.BufferedImage;
import java.io.File;
import java.lang.*;
import java.awt.*;
import java.awt.Graphics;
import javax.imageio.ImageIO;
import java.io.IOException;
import javax.swing.*;
import java.awt.Rectangle;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

public class Marker implements KeyListener
{
	private int score, speed, multiplyer;
	private int y, x, upX, downX, leftX, rightX, center;
	private int spot, min;
	private boolean upPressed, downPressed, leftPressed, rightPressed;
	private BufferedImage imgUpHit, imgDownHit, imgLeftHit, imgRightHit, imgBackground,
				imgUpSelect, imgRightSelect, imgDownSelect, imgLeftSelect;

   public Marker()
   {
		try {
			imgBackground = ImageIO.read(new File("pictures/Background.JPG"));
			imgUpHit = ImageIO.read(new File("pictures/arrows/upHit.PNG"));
			imgDownHit = ImageIO.read(new File("pictures/arrows/downHit.PNG"));
```

```
                imgLeftHit = ImageIO.read(new File("pictures/arrows/leftHit.PNG"));
                imgRightHit = ImageIO.read(new File("pictures/arrows/rightHit.PNG"));
                imgUpSelect = ImageIO.read(new File("pictures/arrows/upSelect.PNG"));
                imgRightSelect = ImageIO.read(new File("pictures/arrows/rightSelect.PNG"));
                imgDownSelect = ImageIO.read(new File("pictures/arrows/downSelect.PNG"));
                imgLeftSelect = ImageIO.read(new File("pictures/arrows/leftSelect.PNG"));
            }catch (IOException e) {

                e.printStackTrace();
                System.exit(1);
            }



            y = 30;
            center = 60;
            leftX = 200;
            upX = leftX+70;
            downX = upX+70;
   rightX = downX+70;

}

    public void render(Graphics g)
    {
        g.drawImage(imgBackground,0,0,800,600,null); //background

        if(leftPressed == true)
            g.drawImage(imgLeftSelect,leftX,y,65,65,null); //hitboxes
```

```
                else
                        g.drawImage(imgLeftHit,leftX,y,65,65,null);

                if(upPressed == true)
                        g.drawImage(imgUpSelect,upX,y,65,65,null);
                else
                        g.drawImage(imgUpHit,upX,y,65,65,null);

                if(downPressed == true)
                        g.drawImage(imgDownSelect,downX,y,65,65,null);
                else
                        g.drawImage(imgDownHit,downX,y,65,65,null);

                if(rightPressed == true)
                        g.drawImage(imgRightSelect,rightX,y,65,65,null);
                else
                        g.drawImage(imgRightHit,rightX,y,65,65,null);
        }
public void keyPressed(KeyEvent e)
{
                if(e.getKeyCode() == KeyEvent.VK_A)
                        leftPressed = true;
                if(e.getKeyCode() == KeyEvent.VK_W)
                        upPressed = true;
                if(e.getKeyCode() == KeyEvent.VK_D)
                        rightPressed = true;
                if(e.getKeyCode() == KeyEvent.VK_S)
                        downPressed = true;
```

```
    }

    public void keyReleased(KeyEvent e)
    {
                    if(e.getKeyCode() == KeyEvent.VK_A)
                            leftPressed = false;
                    if(e.getKeyCode() == KeyEvent.VK_W)
                            upPressed = false;
                    if(e.getKeyCode() == KeyEvent.VK_D)
                            rightPressed = false;
                    if(e.getKeyCode() == KeyEvent.VK_S)
                            downPressed = false;
    }
    public void keyTyped(KeyEvent e)
            {


            }
}
```

```
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;

public class PlayMusic
{

   private final int BUFFER_SIZE = 128000;
   private File soundFile;
   private AudioInputStream audioStream;
   private AudioFormat audioFormat;
   private SourceDataLine sourceLine;

   public void startSong(String filename)
   {
      String strFilename = filename;

      try {
         soundFile = new File(strFilename);
      } catch (Exception e) {
         e.printStackTrace();
         System.exit(1);
      }
```

```
try {
   audioStream = AudioSystem.getAudioInputStream(soundFile);
} catch (Exception e){
   e.printStackTrace();
   System.exit(1);
}

audioFormat = audioStream.getFormat();

DataLine.Info info = new DataLine.Info(SourceDataLine.class, audioFormat);

try {
   sourceLine = (SourceDataLine) AudioSystem.getLine(info);
   sourceLine.open(audioFormat);
} catch (LineUnavailableException e) {
   e.printStackTrace();
   System.exit(1);
} catch (Exception e) {
   e.printStackTrace();
   System.exit(1);
}

sourceLine.start();

int nBytesRead = 0;
byte[] abData = new byte[BUFFER_SIZE];
while (nBytesRead != -1) {
```

```
            try {
                nBytesRead = audioStream.read(abData, 0, abData.length);
            } catch (IOException e) {
                e.printStackTrace();
            }
            if (nBytesRead >= 0) {
                @SuppressWarnings("unused")
                int nBytesWritten = sourceLine.write(abData, 0, nBytesRead);
            }
        }


    }


    public void stopSong()
    {
        sourceLine.stop();
              }
}
```