```java

/*
 * Audio.java
 */

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import java.io.File;

public class Audio {
   String[] song = {"Music/1.wav", "Music/2.wav", "Music/3.wav",
   "Music/4.wav",  "Music/5.wav", "Music/background.wav",};
   Clip clip;
   AudioInputStream audioInputStream;

   public Audio() {
   }

   public Audio(int s) {
      try {
         audioInputStream = AudioSystem.getAudioInputStream(new File(song
         [s]).getAbsoluteFile());
         clip = AudioSystem.getClip();
         clip.open(audioInputStream);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   public void playAudio() {
      clip.start();
      clip.loop(Clip.LOOP_CONTINUOUSLY);
   }

   public void stopAudio() {
      clip.stop();
   }
}

```

```java
/*
 * Button.java
 */

/*
 * WHITE = 0x000000, rgba: (255,255,255,0/1), (0,0,0,0), or
 * BLACK = 0xFFFFFF, rgba: (0,0,0,1)
 * RED = = 0xFF0000, rgba: (255,0,0,0.00-1.00)
 * GREEN = 0x00FF00
 * BLUE = 0x0000FF
 * LIME_GREEN =
 * HOT_PINK = 0xFF00FF
 * PURPLE = 0x800080
 */

import java.awt.*;

public class Button {
   int x = 0;
   int y = 0;
   boolean keyPressed = false;

   public void drawCircle(Graphics g, int move) {
      if (move != 1) {
         g.setColor(Color.BLACK); // // g.setColor(new Color(0xFFFFFF));
         g.fillOval(504, 234, 41, 41);
         g.setColor((Color.GREEN)); // g.setColor((Color.green));
         g.fillOval(500, 230, 40, 40);
      }
      if (move == 1) {
         g.setColor(Color.DARK_GRAY); // g.setColor((Color.red));
         g.fillOval(504, 234, 40, 40);
      }
      if (move != 2) {
         g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));
         g.fillOval(504, 334, 41, 41);
         g.setColor(Color.RED); // g.setColor((Color.green));
         g.fillOval(500, 330, 40, 40);
      }
      if (move == 2) {
```

```
79            g.setColor(Color.DARK_GRAY); // g.setColor((Color.red));
80            g.fillOval(504, 334, 40, 40);
81         }
82         if (move != 3) {
83            g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF))
84            g.fillOval(504, 434, 41, 41);
85            g.setColor(Color.YELLOW); // g.setColor((Color.green));
86            g.fillOval(500, 430, 40, 40);
87         }
88         if (move == 3) {
89            g.setColor(Color.DARK_GRAY); // g.setColor((Color.red));
90            g.fillOval(504, 434, 40, 40);
91         }
92         if (move != 4) {
93            g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));
94            g.fillOval(504, 534, 41, 41);
95            g.setColor(Color.BLUE); // g.setColor((Color.green));
96            g.fillOval(500, 530, 40, 40);
97         }
98         if (move == 4) {
99            g.setColor(Color.DARK_GRAY); // g.setColor((Color.red));
100           g.fillOval(504, 534, 40, 40);
101        }
102        if (move != 5) {
103           g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));
104           g.fillOval(504, 634, 41, 41);
105           g.setColor(Color.ORANGE); // g.setColor((Color.green));
106           g.fillOval(500, 630, 40, 40);
107        }
108        if (move == 5) {
109           g.setColor(Color.DARK_GRAY); // g.setColor((Color.red));
110           g.fillOval(504, 634, 40, 40);
111        }
112     }
113
114     public void drawTriangle(Graphics g, int move) {
115        //g.setColor(Color.WHITE); // g.setColor(new Color(0xFFFFFFFF,
           true));
116        //g.fillPolygon(new int[]{513,513,533}, new int[]{242,262,252},3);
117        g.setColor(Color.blue);
```

```
118         if (move == 1) {
119             g.setColor(Color.WHITE);
120             g.fillPolygon(new int[] {514, 514, 534}, new int[] {245, 265,
                255}, 3);
121         }
122         if (move != 1) {
123             g.setColor(Color.BLACK);
124             g.fillPolygon(new int[] {510, 510, 530}, new int[] {241, 261,
                251}, 3);
125         }
126         if (move == 2) {
127             g.setColor(Color.WHITE);
128             g.fillPolygon(new int[] {514, 514, 534}, new int[] {345, 365,
                355}, 3);
129         }
130         if (move != 2) {
131             g.setColor(Color.BLACK);
132             g.fillPolygon(new int[] {510, 510, 530}, new int[] {341, 361,
                351}, 3);
133         }
134         if (move == 3) {
135             g.setColor(Color.WHITE);
136             g.fillPolygon(new int[] {514, 514, 534}, new int[] {445, 465,
                455}, 3);
137         }
138         if (move != 3) {
139             g.setColor(Color.BLACK);
140             g.fillPolygon(new int[] {510, 510, 530}, new int[] {441, 461,
                451}, 3);
141         }
142         if (move == 4) {
143             g.setColor(Color.WHITE);
144             g.fillPolygon(new int[] {514, 514, 534}, new int[] {545, 565,
                555}, 3);
145         }
146         if (move != 4) {
147             g.setColor(Color.BLACK);
148             g.fillPolygon(new int[] {510, 510, 530}, new int[] {541, 561,
                551}, 3);
149         }
```

```
150         if (move == 5) {
151             g.setColor(Color.WHITE);
152             g.fillPolygon(new int[] {514, 514, 534}, new int[] {645, 665,
                655}, 3);
153         }
154         if (move != 5) {
155             g.setColor(Color.BLACK);
156             g.fillPolygon(new int[] {510, 510, 530}, new int[] {641, 661,
                651}, 3);
157         }
158     }
159
160     // Version 4
161     public void gameButton(Graphics g, boolean[] key) {
162         Graphics2D g2 = (Graphics2D) g;
163         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints
            .VALUE_ANTIALIAS_ON);
164         String[] letters = {"1", "2", "3", "4"};
165         String[] buttonColors = {"GREEN", "Color.RED", "Color.YELLOW",
            "Color.BLUE"};
166         Color[] colors = {Color.GREEN, Color.YELLOW, Color.RED, Color.BLUE
            };
167         Color[] darkColors = {Color.GREEN, Color.YELLOW, Color.RED, Color.
            BLUE};
168         int[] x = {45, 195, 345, 495};
169         int y = 625;
170         for (int i = 0; i < 4; i++) {
171             g2.setColor(key[i] ? colors[i].darker() : colors[i]);
172             g2.fillOval(x[i], y, 60, 60);
173             g2.setColor(Color.BLACK);
174             g2.setFont(new Font("SansSerif", Font.BOLD, 32));
175             g2.drawString(letters[i], x[i] + 20, y + 42);
176         }
177     }
178
179     /*
180     // Version 3
181     public void gameButton(Graphics g, boolean[] key) {
182         Graphics2D g2 = (Graphics2D) g;
183         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
```

```
        RenderingHints.VALUE_ANTIALIAS_ON);
184         String[] letters = {"A","S","J","K"};
185         int[] x = {45,195,345,495};
186         int y = 625;
187         for(int i=0;i<4;i++) {
188            g2.setColor(key[i] ? new Color(40,40,40) : new
            Color(220,220,220));
189            g2.fillOval(x[i], y, 60, 60);
190            g2.setColor(Color.BLACK);
191            g2.setFont(new Font("SansSerif",Font.BOLD,32));
192            g2.drawString(letters[i], x[i]+20, y+42);
193         }
194      }
195   */
196
197   /*
198      // Version 2
199      public void gameButton (Graphics g, boolean[] key) {
200          String[] letters = {"A","S","J","K"};
201          int[] x = {45,195,345,495};
202          int y = 625;
203
204       }
205
206         g.setFont(new Font("monospaced",Font.BOLD,50));
207           if(!key[0]) {
208               g.setColor(new Color(0x000000)); // 0xFFFFFFF = RED
209               g.fillOval(45, 625, 60, 60);
210               g.setColor((Color.green));
211               g.fillOval(40, 620, 60, 60);
212               g.setColor(Color.GREEN);
213               g.drawString("1",57,665);
214           }
215           if(key[0]) {
216               g.setColor((Color.red));
217               g.fillOval(45, 625, 60, 60);
218               g.setColor(Color.BLUE);
219               g.drawString("1",62,670);
220           }
221           if(!key[1]) {
```

```
222             g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED
223             g.fillOval(195, 625, 60, 60);
224             g.setColor((Color.green));
225             g.fillOval(190, 620, 60, 60);
226             g.setColor(Color.BLUE);
227             g.drawString("2",207,665);
228         }
229         if(key[1]) {
230             g.setColor((Color.red));
231             g.fillOval(195, 625, 60, 60);
232             g.setColor(Color.BLUE);
233             g.drawString("2",212,670);
234         }
235         if(!key[2]) {
236             g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED
237             g.fillOval(345, 625, 60, 60);
238             g.setColor((Color.green));
239             g.fillOval(340, 620, 60, 60);
240             g.setColor(Color.BLUE);
241             g.drawString("3",357,665);
242         }
243         if(key[2]) {
244             g.setColor((Color.red));
245             g.fillOval(345, 625, 60, 60);
246             g.setColor(Color.BLUE);
247             g.drawString("3",362,670);
248         }
249         if(!key[3]) {
250             g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED
251             g.fillOval(495, 625, 60, 60);
252             g.setColor((Color.green));
253             g.fillOval(490, 620, 60, 60);
254             g.setColor(Color.BLUE);
255             g.drawString("4",507,665);
256         }
257         if(key[3]) {
258             g.setColor((Color.red));
259             g.fillOval(495, 625, 60, 60);
260             g.setColor(Color.BLUE);
261             g.drawString("4",512,670);
```

```
262             }
263         }
264      */
265     }
266
267     /*
268      * GameText.java
269      */
270
271     import java.awt.*;
272
273     public class GameText {
274        private String TITLE = "Manuvo";
275
276        public void gameName(Graphics g) {
277           g.setFont(new Font(Font.DIALOG_INPUT, Font.BOLD, 90));
278           g.setColor(Color.GRAY); // g.setColor(new Color(0xFFFFFF00, true));
279           g.drawString(TITLE, 264, 100);
280           g.setColor(Color.BLUE); // g.setColor(new Color(156, 112, 248,
              255));
281           g.drawString(TITLE, 260, 100);
282           g.setColor(new Color(0xFFFFFF00, true)); // 0xFFFFFFFF = Black
283     /*
284             g.drawString("Tiles",264,190);
285             g.setColor(new Color(156, 112, 248, 255)); // Color(156, 112,
              248, 255) = Purple
286             g.drawString("Tiles",260,190);
287      */
288        }
289
290        public void songText(Graphics g) {
291           g.setColor(new Color(114, 222, 210, 255)); // Color(156, 112, 248,
              255) = Ugly Cyan_Green
292           g.setFont(new Font("monospaced Bold", Font.ITALIC, 30));
293
294           g.drawString("Someone you Loved", 180, 230);
295           g.drawString("Memories", 180, 330);
296           g.drawString("Play Date", 180, 430);
297           g.drawString("Dance Monkey", 180, 530);
298           g.drawString("Counting Stars", 180, 630);
```

```
299     }
300
301     // Guitar = G, R, Y, Bl, O
302     public void difficultyText(Graphics g) {
303         g.setFont(new Font("serif", Font.BOLD, 20));
304
305         // Green
306         //g.setColor(new Color(0x0FFD0FF, true)); // 0xFFD0FF00 = Pink
307         g.setColor(new Color(0xCC00FF00, true)); // 0xFFD0FF00 = Purple
308         g.drawString("Very Easy", 400, 280);
309
310         g.setColor(Color.RED);
311         g.drawString("Easy", 400, 380);
312
313         g.setColor(new Color(0xFFFFFF00, true)); // 0x099202 = Green
314         g.drawString("Medium", 400, 480);
315
316         g.setColor(Color.BLUE);
317         g.drawString("Hard", 400, 580);
318
319         g.setColor(Color.ORANGE);
320         g.drawString("Very Hard", 400, 680);
321     }
322
323     public void gameOver(Graphics g, int score) {
324         g.setFont(new Font("serif", Font.BOLD, 50));
325         g.setColor(new Color(0xFFFFFFFF, true)); // g.setColor(new
        Color(0xFFE600));
326         g.drawString("Score: " + score, 180, 255);
327         g.setColor(new Color(0xFFFFFFFF)); // g.setColor(new
        Color(0x800000));
328         g.drawString("Game Over", 150, 325);
329         g.setColor(new Color(0xFFFFFFFF)); // g.setColor(new
        Color(0x26FF00));
330         g.drawString("Press Enter", 160, 400);
331     }
332
333     // Version 4
334     public void score(Graphics g, int score, String compliment, int
        complimentSize) {
```

```
335         Graphics2D g2 = (Graphics2D) g;
336         g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
            RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
337         g2.setColor(Color.BLACK);
338         g2.setFont(new Font("SansSerif", Font.BOLD, 48));
339         g2.drawString(String.valueOf(score), 270, 70);
340         g2.setFont(new Font("SansSerif", Font.PLAIN, complimentSize));
341         g2.setColor(new Color(140, 140, 140)); // soft gray
342         g2.drawString(compliment, 230, 140);
343     }
344
345     /*
346     // Version 3
347     public void score(Graphics g,int score, String compliment, int
        complimentSize) {
348         Graphics2D g2 = (Graphics2D) g;
349         g2.setFont(new Font("SansSerif",Font.BOLD,50));
350         g2.setColor(Color.white);
351         g2.drawString(String.valueOf(score),270,70);
352         g2.setFont(new Font("SansSerif",Font.BOLD,complimentSize));
353         g2.setColor(new Color(255, 255, 255, 200));
354         g2.drawString(compliment,230,140);
355     }
356     */
357
358 /*
359     // Version 2
360     public void score(Graphics g,int score, String compliment, int
        complimentSize) {
361         g.setColor(new Color(0xFFFFFF)); // 0xFFFFFF = White
362         g.setFont(new Font("serif",Font.BOLD,50));
363         g.drawString(String.valueOf(score),300,70);
364         if(compliment.equals("Perfect"))
365             g.setColor(new Color(0x630061)); // 0x630061 = Purple
366         else if(compliment.equals("Great"))
367             g.setColor(new Color(0x001E99)); // 0x001E99 = Blue
368         else
369             g.setColor(new Color(0x008787)); // 0x008787 = Cyan
370         g.setFont(new Font("serif",Font.BOLD,complimentSize));
371         if(complimentSize == 50)
```

```
372             g.drawString(compliment,280,120);
373         else
374             g.drawString(compliment,230,140);
375     }
376  */
377 }
378
379 /*
380  * Main.java
381  */
382
383 import javax.swing.*;
384
385 public class Main {
386    public static void main(String[] args) {
387       JFrame frame = new JFrame();
388       MainScreen mainScreen = new MainScreen();
389       //GamePlay gamePlay = new GamePlay();
390       frame.setBounds(400,10,600,750); // frame.setBounds(x,y,width,
          height);
391       frame.setTitle("Manuvo");
392       frame.setResizable(true);
393       frame.setVisible(true);
394       frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
395       frame.add(mainScreen);
396    }
397 }
398
399 /*
400  * MainScreen.java
401  */
402
403 import javax.imageio.ImageIO;
404 import javax.swing.*;
405 import java.awt.*;
406 import java.awt.event.*;
407 import java.awt.image.BufferedImage;
408 import java.io.File;
409 import java.io.IOException;
410 import java.util.Arrays;
```

```
411  import java.util.Random;
412
413  public class MainScreen extends JPanel implements KeyListener,
     ActionListener {
414      private Timer time;
415      private int delay = 15;
416      private boolean play = false;
417      private boolean over = false;
418      Random rand = new Random();
419      private int[] backBallX = new int[40];
420      private int[] backBallY = new int[40];
421      private boolean check = true;
422      private final int MISTAKE = 50;
423      //private final int bottomBond = 440;
424      private final int bottomBond = 450; // Hit Tile Out-Of-Bounds now
425      private int x = 0;
426      private int score = 0;
427      private int foulY = 0;
428      private int foulPlace;
429      private int totalTiles = 0;
430      private int move = 1;
431      private int selectRectY = 200;
432      private int speed = 5;
433      private boolean go = false;
434      private boolean key[] = new boolean[5];
435      private boolean tilesCheck[] = new boolean[5];
436      private boolean checkTilesProduce[] = new boolean[5];
437      private int tilesY[] = new int[5];
438      private String compliment = "";
439      private int complimentSize = 50; // Compliment {"Perfect","Great"}
440      private boolean foul = false;
441      private boolean startSong = false;
442      private boolean audioStatus = false;
443      private boolean backgroundAudioStatus = false;
444      Audio audio;
445      Audio backgroundAudio;
446      //Audio2 audio2;
447      //Audio3 audio3;
448      //Audio4 audio4;
449      //Audio5 audio5;
```

```
450     //BackgroundAudio backgroundAudio;
451     String[] images = {
452           "Images/someoneYouLoved.jpeg",
453           "Images/memories.jpg",
454           "Images/playDate.jpeg",
455           "Images/danceMonkey.jpeg",
456           "Images/countingStars.jpeg",
457           "Images/HandGrip2.jpg"
458     };
459     BufferedImage[] songImage = new BufferedImage[5];
460     BufferedImage iconImage = null;
461
462     /* BufferedImage song1 = null;
463     BufferedImage song2 = null;
464     BufferedImage song3 = null;
465     BufferedImage song4 = null;
466     BufferedImage song5 = null;
467     BufferedImage iconImage = null; */
468
469     public MainScreen() {
470        initialValues();
471        addKeyListener(this);
472        setFocusable(true);
473        setFocusTraversalKeysEnabled(false);
474        time = new Timer(delay, this);
475        time.start();
476     }
477
478     public void initialValues() {
479        compliment = "";
480        totalTiles = 0;
481        score = 0;
482        delay = 15;
483        play = false;
484        over = false;
485        check = true;
486        x = 0;
487        move = 1;
488        selectRectY = 200; // height of Song and Tile Hit rectangles
489        go = false;
```

```
490         foulY = 0;
491         audioStatus = false;
492         startSong = false;
493         backgroundAudioStatus = false;
494         foul = false;
495         //songImage = null;
496         /* song1 = null;
497         song2 = null;
498         song3 = null;
499         song4 = null;
500         song5 = null; */
501         for (int i = 0; i < 5; i++) {
502            key[i] = false;
503            tilesCheck[i] = false;
504            checkTilesProduce[i] = false;
505            tilesY[i] = -150; // Hit Tile Height = 150px tall
506         }
507         for (int i = 0; i < 40; i++) {
508            backBallX[i] = rand.nextInt(550);
509            backBallY[i] = rand.nextInt(700);
510         }
511      }
512
513      public void paint(Graphics graphics) {
514         //BackGround
515         backGround(graphics);
516         //backBalls(graphics);
517         //name
518         GameText gameText = new GameText();
519         gameText.gameName(graphics);
520         //icon
521         icon(graphics);
522         //song
523         //playDate, danceMonkey, memories, countingStars, someoneYouLoved.
524         drawSongImage(graphics);
525         getSongImage(check);
526         check = false;
527         //songName
528         gameText.songText(graphics);
529         //difficulty
```

```
530         gameText.difficultyText(graphics);
531         //playButtonIn
532         Button button = new Button();
533         button.drawCircle(graphics, move);
534         //buttonTriangleIn
535         button.drawTriangle(graphics, move);
536         Song1 song1 = new Song1();
537         if (!go && !backgroundAudioStatus) {
538            backgroundAudio = new Audio(5);
539            backgroundAudio.playAudio();
540            backgroundAudioStatus = true;
541         }
542         if (go) {
543            backgroundAudio.stopAudio();
544         }
545         if (move == 1) {
546            speed = 2;
547            if (startSong) {
548               audio = new Audio(0);
549               audio.playAudio();
550               audioStatus = true;
551               startSong = false;
552            }
553            if (over && audioStatus) {
554               audio.stopAudio();
555               audioStatus = false;
556            }
557         }
558         if (move == 2) {
559            speed = 4;
560            if (startSong) {
561               audio = new Audio(1);
562               audio.playAudio();
563               audioStatus = true;
564               startSong = false;
565            }
566            if (over && audioStatus) {
567               audio.stopAudio();
568               audioStatus = false;
569            }
```

```
570         }
571         if (move == 3) {
572            speed = 6;
573            if (startSong) {
574               audio = new Audio(2);
575               audio.playAudio();
576               audioStatus = true;
577               startSong = false;
578            }
579            if (over && audioStatus) {
580               audio.stopAudio();
581               audioStatus = false;
582            }
583         }
584         if (move == 4) {
585            speed = 8;
586            if (startSong) {
587               audio = new Audio(3);
588               audio.playAudio();
589               audioStatus = true;
590               startSong = false;
591            }
592            if (over && audioStatus) {
593               audio.stopAudio();
594               audioStatus = false;
595            }
596         }
597         if (move == 5) {
598            speed = 10;
599            if (startSong) {
600               audio = new Audio(4);
601               audio.playAudio();
602               audioStatus = true;
603               startSong = false;
604            }
605            if (over && audioStatus) {
606               audio.stopAudio();
607               audioStatus = false;
608            }
609         }
```

```
610         if (go) {
611             song1.gameInBackGround(graphics);
612             button.gameButton(graphics, key);
613             if (!over) {
614                 Tiles tiles = new Tiles();
615                 tiles.drawTiles(graphics, tilesCheck, tilesY, play);
616                 GameText gameText1 = new GameText();
617                 gameText1.score(graphics, score, compliment, complimentSize);
618             }
619         }
620         if (foul) {
621             Tiles tiles = new Tiles();
622             tiles.drawFoul(graphics, foulPlace, foulY);
623         }
624         if (over) {
625             GameText gameTextOver = new GameText();
626             gameTextOver.gameOver(graphics, score);
627             try {
628                 Thread.sleep(500);
629             } catch (InterruptedException e) {
630                 e.printStackTrace();
631             }
632             foul = false;
633         }
634         graphics.dispose();
635         repaint();
636     }
637 
638     public void backGround(Graphics g) {
639         // Start Screen Background - Black with rising bubbles
640         g.setColor(new Color(0, 0, 0, 255)); // Black Background
641         g.fillRect(0, 0, 600, 750);
642         // Select Song Block Perimeter
643         g.setColor(Color.WHITE); // g.setColor(new Color(255, 255, 255,
            163)); = White
644         g.draw3DRect(0, selectRectY, 580, 100, true); // Select Song
            Outline size/pos
645     }
646 
647     /*
```

```
648      // Start Screen - Background Rising Bubbles
649      private void backBalls(Graphics g) {
650         g.setColor(Color.RED); // g.setColor(new Color(255, 255, 255,
            131)); = Gray
651         for(int i = 0; i < 40; i++) {
652            g.fillOval(backBallX[i], backBallY[i], 10, 10); // Bubble Size
653         }
654      }
655       */
656      // Start screen - Song Image
657      public void getSongImage(boolean check) {
658         if (check) {
659            try {
660               for (int i = 0; i <= 4; i++) {
661                  songImage[i] = ImageIO.read(new File(images[i]));
662               }
663               //song1 = ImageIO.read(new
                  File("Images/someoneYouLoved.jpeg"));
664               //song2 = ImageIO.read(new File("Images/memories.jpg"));
665               //song3 = ImageIO.read(new File("Images/playDate.jpeg"));
666               //song4 = ImageIO.read(new File("Images/danceMonkey.jpeg"));
667               //song5 = ImageIO.read(new
                  File("Images/countingStars.jpeg"));
668
669            } catch (IOException e) {
670               e.printStackTrace();
671            }
672         }
673      }
674
675      // Start Screen - Song Select Image (image,x,y,width,height)
676      public void drawSongImage(Graphics g) {
677         int y = 210;
678         for (int i = 0; i <= 4; i++) {
679            g.drawImage(songImage[i], 20, y, 150, 80, this);
680            y += 100;
681         }
682         /* g.drawImage(song1, 20, 210, 150, 80, this);
683         g.drawImage(song2, 20, 310, 150, 80, this);
684         g.drawImage(song3, 20, 410, 150, 80, this);
```

```
685         g.drawImage(song4, 20, 510, 150, 80, this);
686         g.drawImage(song5, 20, 610, 150, 80, this); */
687      }
688
689      // Song Select Page Icon Image
690      public void icon(Graphics g) {
691         if (check) try {
692            //iconImage = ImageIO.read(new File("Images/HandGrip2.jpg"));
693            iconImage = ImageIO.read(new File(images[5]));
694         } catch (IOException e) {
695            e.printStackTrace();
696         }
697         // Image (image,x,y,width,height)
698         g.drawImage(iconImage, 20, 10, 200, 190, this);
699
700      }
701
702      // create tiles from top of screen
703      public void tilesProduce() {
704         totalTiles++;
705         x = rand.nextInt(4);
706         tilesCheck[x] = true;
707      }
708
709      @Override
710      public void actionPerformed(ActionEvent e) {
711         for (int i = 0; i < 40; i++) { // when bubbles float off top of
            screen
712            backBallY[i] -= 1;
713            if (backBallY[i] == 0) {
714               backBallY[i] = 700; // create new bubble y bottom of screen
715               backBallX[i] = rand.nextInt(550); // random x position
716            }
717         }
718         for (int i = 0; i <= 3; i++) {
719            if (tilesCheck[i]) {
720               tilesY[i] += speed;
721            }
722            if (tilesY[i] >= 80 && !checkTilesProduce[i]) {
723               tilesProduce();
```

```
724             checkTilesProduce[i] = true;
725          }
726          if (tilesY[i] >= bottomBond) {
727             over = true;
728             play = false;
729             tilesCheck[i] = false;
730          }
731       }
732
733  /*
734          if(tilesCheck[0]) {
735             tilesY[0]+=speed;
736          }
737          if(tilesCheck[1]) {
738             tilesY[1]+=speed;
739          }
740          if(tilesCheck[2]) {
741             tilesY[2]+=speed;
742          }
743          if(tilesCheck[3]) {
744             tilesY[3]+=speed;
745          }
746          if(tilesY[0]>=80 && !checkTilesProduce[0]) {
747             tilesProduce();
748             checkTilesProduce[0] = true;
749          }
750          if(tilesY[1]>=80 && !checkTilesProduce[1]) {
751             tilesProduce();
752             checkTilesProduce[1] = true;
753          }
754          if(tilesY[2]>=80 && !checkTilesProduce[2]) {
755             tilesProduce();
756             checkTilesProduce[2] = true;
757          }
758          if(tilesY[3]>=80 && !checkTilesProduce[3]) {
759             tilesProduce();
760             checkTilesProduce[3] = true;
761          }
762          if(tilesY[0]>=bottomBond) {
763             over = true;
```

```
764             play = false;
765             tilesCheck[0]=false;
766         }
767         if(tilesY[1]>=bottomBond) {
768             over = true;
769             play=false;
770             tilesCheck[1]=false;
771         }
772         if(tilesY[2]>=bottomBond) {
773             over = true;
774             play=false;
775             tilesCheck[2]=false;
776         }
777         if(tilesY[3]>=bottomBond) {
778             over = true;
779             play = false;
780             tilesCheck[3]=false;
781         }
782     */
783       }
784
785       @Override
786       public void keyTyped(KeyEvent e) {
787       }
788
789       @Override
790       public void keyPressed(KeyEvent e) {
791          if (e.getKeyCode() == KeyEvent.VK_DOWN) {
792             move += 1;
793             selectRectY += 100;
794             if (move == 6) move = 1;
795             if (selectRectY == 700) selectRectY = 200;
796          }
797          if (e.getKeyCode() == KeyEvent.VK_UP) {
798             move -= 1;
799             selectRectY -= 100;
800             if (move == 0) move = 5;
801             if (selectRectY == 100) selectRectY = 600;
802          }
803          if (go && e.getKeyCode() == KeyEvent.VK_1) {
```

```
804             play = true;
805             if (!audioStatus) startSong = true;
806             tilesProduce();
807         }
808         if (e.getKeyCode() == KeyEvent.VK_ENTER) {
809             go = true;
810         }
811         if (over && e.getKeyCode() == KeyEvent.VK_ENTER) {
812             initialValues();
813         }
814         if (e.getKeyCode() == KeyEvent.VK_1 && !over) {
815             complimentSize = 80;
816             if (tilesY[0] < tilesY[1] - MISTAKE || tilesY[0] < tilesY[2] -
            MISTAKE || tilesY[0] < tilesY[3] - MISTAKE) {
817                 over = true;
818                 foul = true;
819                 foulPlace = 0;
820                 foulY = sortTilesY(tilesY);
821             }
822             key[0] = true;
823         }
824         if (e.getKeyCode() == KeyEvent.VK_2 && !over) {
825             complimentSize = 80;
826             if (tilesY[1] < tilesY[0] - MISTAKE || tilesY[1] < tilesY[2] -
            MISTAKE || tilesY[1] < tilesY[3] - MISTAKE) {
827                 over = true;
828                 foul = true;
829                 foulPlace = 1;
830                 foulY = sortTilesY(tilesY);
831             }
832             key[1] = true;
833         }
834         if (e.getKeyCode() == KeyEvent.VK_3 && !over) {
835             complimentSize = 80;
836             if (tilesY[2] < tilesY[0] - MISTAKE || tilesY[2] < tilesY[1] -
            MISTAKE || tilesY[2] < tilesY[3] - MISTAKE) {
837                 over = true;
838                 foul = true;
839                 foulPlace = 2;
840                 foulY = sortTilesY(tilesY);
```

```
841             }
842             key[2] = true;
843         }
844         if (e.getKeyCode() == KeyEvent.VK_4 && !over) {
845             complimentSize = 80;
846             if (tilesY[3] < tilesY[0] - MISTAKE || tilesY[3] < tilesY[1] -
                MISTAKE || tilesY[3] < tilesY[2] - MISTAKE) {
847                 over = true;
848                 foul = true;
849                 foulPlace = 3;
850                 foulY = sortTilesY(tilesY);
851             }
852             key[3] = true;
853         }
854     }
855 
856     @Override
857     public void keyReleased(KeyEvent e) {
858         if (e.getKeyCode() == KeyEvent.VK_1) {
859             complimentSize = 50; // Compliment Player Size = 50px
860             key[0] = false;
861         }
862         if (e.getKeyCode() == KeyEvent.VK_2) {
863             complimentSize = 50;
864             key[1] = false;
865         }
866         if (e.getKeyCode() == KeyEvent.VK_3) {
867             complimentSize = 50;
868             key[2] = false;
869         }
870         if (e.getKeyCode() == KeyEvent.VK_4) {
871             complimentSize = 50;
872             key[3] = false;
873         }
874         if (e.getKeyCode() == KeyEvent.VK_1 && tilesCheck[0] && tilesY[0] >
             200) {
875             ScoreCalculate scoreCalculate = new ScoreCalculate();
876             compliment = scoreCalculate.score(tilesY[0]);
877             if (compliment.equals("Perfect")) score += 3;
878             else if (compliment.equals("Great")) score += 2;
```

```
879             else score += 1;
880             key[0] = false;
881             tilesCheck[0] = false;
882             checkTilesProduce[0] = false;
883             tilesY[0] = -150;
884             tilesProduce();
885         }
886         if (e.getKeyCode() == KeyEvent.VK_2 && tilesCheck[1] && tilesY[1] >
         200) {
887             ScoreCalculate scoreCalculate = new ScoreCalculate();
888             compliment = scoreCalculate.score(tilesY[1]);
889             if (compliment.equals("Perfect")) score += 3;
890             else if (compliment.equals("Great")) score += 2;
891             else score += 1;
892             key[1] = false;
893             tilesCheck[1] = false;
894             checkTilesProduce[1] = false;
895             tilesY[1] = -150;
896             tilesProduce();
897         }
898         if (e.getKeyCode() == KeyEvent.VK_3 && tilesCheck[2] && tilesY[2] >
         200) {
899             ScoreCalculate scoreCalculate = new ScoreCalculate();
900             compliment = scoreCalculate.score(tilesY[2]);
901             if (compliment.equals("Perfect")) score += 3;
902             else if (compliment.equals("Great")) score += 2;
903             else score += 1;
904             key[2] = false;
905             tilesCheck[2] = false;
906             checkTilesProduce[2] = false;
907             tilesY[2] = -150;
908             tilesProduce();
909         }
910         if (e.getKeyCode() == KeyEvent.VK_4 && tilesCheck[3] && tilesY[3] >
         200) {
911             ScoreCalculate scoreCalculate = new ScoreCalculate();
912             compliment = scoreCalculate.score(tilesY[3]);
913             if (compliment.equals("Perfect")) score += 3;
914             else if (compliment.equals("Great")) score += 2;
915             else score += 1;
```

```
916             key[3] = false;
917             tilesCheck[3] = false;
918             checkTilesProduce[3] = false;
919             tilesY[3] = -150;
920             tilesProduce();
921          }
922       }
923
924       public int sortTilesY(int[] tilesY) {
925          int[] sort = new int[4];
926          for (int i = 0; i < 4; i++) {
927             sort[i] = tilesY[i];
928          }
929          Arrays.sort(sort);
930          return sort[3];
931       }
932    }
933
934    /*
935     * ScoreCalculate.java
936     */
937
938    public class ScoreCalculate {
939       public String score(int place) {
940          if (place >= 330 && place <= 370) {
941             return "Perfect";
942          } else if (place >= 300 && place < 320 || place > 360 && place <=
          380) {
943             return "Great";
944          } else {
945             return "Cool";
946          }
947       }
948    }
949
950    /*
951     * Song1.java
952     */
953
954    import javax.swing.*;
```

```
955  import java.awt.*;
956  import java.awt.event.ActionEvent;
957  import java.awt.event.ActionListener;
958  import java.awt.event.KeyEvent;
959  import java.awt.event.KeyListener;
960
961  class Song1 extends JPanel implements KeyListener, ActionListener {
962     /*
963     // Version 4
964     public void gameInBackGround(Graphics g) {
965        Graphics2D g2 = (Graphics2D) g;
966        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
          RenderingHints.VALUE_ANTIALIAS_ON);
967        // Pure white background
968        g2.setColor(Color.WHITE);
969        g2.fillRect(0, 0, 600, 600); // Glowing lane separators
970        // Very subtle lane separators
971        g2.setColor(new Color(200, 200, 200));
972        g2.setStroke(new BasicStroke(2f));
973        g2.drawLine(150, 0, 150, 600);
974        g2.drawLine(300, 0, 300, 600);
975        g2.drawLine(450, 0, 450, 600); // Hit zone at bottom
976        // Hit zone (light neutral)
977        g2.setColor(new Color(240, 240, 240));
978        g2.fillRect(0, 450, 600, 200);
979     }
980     */
981
982     /*
983     // version 3
984     public void gameInBackGround(Graphics g) {
985        // Smooth rendering
986        Graphics2D g2 = (Graphics2D) g;
987        // Gradient background
988        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
          RenderingHints.VALUE_ANTIALIAS_ON);
989        GradientPaint bg = new GradientPaint(0, 0, new Color(70,0,120), 0,
          600, new Color(140,0,255));
990        g2.setPaint(bg);
991        g2.fillRect(0, 0, 600, 600); // Glowing lane separators
```

```
992         g2.setStroke(new BasicStroke(6f));
993         g2.setColor(new Color(255,255,255,90));
994         g2.drawLine(150,0,150,600);
995         g2.drawLine(300,0,300,600);
996         g2.drawLine(450,0,450,600); // Hit zone at bottom
997         g2.setPaint(new Color(0,0,0,160));
998         g2.fillRect(0, 450, 600, 200);
999      }
1000     */
1001
1002     // version 2
1003     public void gameInBackGround(Graphics g) {
1004        // Smooth rendering
1005        Graphics2D g2 = (Graphics2D) g;
1006        // Gradient background
1007        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints
           .VALUE_ANTIALIAS_ON);
1008        GradientPaint bg = new GradientPaint(0, 0, new Color(70,0,120), 0,
           600, new Color(140,0,255));
1009        g2.setPaint(bg);
1010        g2.fillRect(0, 0, 600, 600); // Glowing lane separators
1011        g2.setStroke(new BasicStroke(6f));
1012        g2.setColor(new Color(255,255,255,90));
1013        g2.drawLine(150,0,150,600);
1014        g2.drawLine(300,0,300,600);
1015        g2.drawLine(450,0,450,600); // Hit zone at bottom
1016        g2.setPaint(new Color(0,0,0,160));
1017        g2.fillRect(0, 450, 600, 200);
1018     }
1019
1020  /*
1021     // version 1
1022     public void gameInBackGround(Graphics g) {
1023        g.setColor(new Color(0x0000FF)); // 0x842EDC = purple
1024        g.fill3DRect(0, 0, 600, 600, true); // glowing lane separators
1025        g.setColor(new Color(0xFFFFFF));
1026        g.drawLine(150, 0, 150, 600);
1027        g.drawLine(300, 0, 300, 600);
1028        g.drawLine(450, 0, 450, 600); // bottom hit zone
1029        g.setColor(Color.BLACK);
```

```
1030        g.fill3DRect(0, 600, 600, 150, true);
1031        g.setColor(new Color(0x34000000, true)); // 0x34000000 = RED
1032        g.fill3DRect(0, 450, 600, 200, true);
1033     }
1034  */
1035     @Override
1036     public void actionPerformed(ActionEvent e) {
1037     }
1038
1039     @Override
1040     public void keyTyped(KeyEvent e) {
1041     }
1042
1043     @Override
1044     public void keyPressed(KeyEvent e) {
1045     }
1046
1047     @Override
1048     public void keyReleased(KeyEvent e) {
1049     }
1050  }
1051
1052  /*
1053   * Tiles.java
1054   */
1055
1056  import javax.swing.*;
1057  import java.awt.*;
1058  import java.util.Random;
1059
1060  public class Tiles extends JPanel {
1061     private int pos;
1062     private int YHEIGHT = 200;
1063     Random random = new Random();
1064
1065     // Version 4
1066     public void drawTiles(Graphics g, boolean[] tilesCheck, int[] tilesY,
       boolean play) {
1067        Graphics2D g2 = (Graphics2D) g;
1068        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints
```

```
            .VALUE_ANTIALIAS_ON);
1069        int w = 150, h = 200;
1070        if (!play) {
1071           g2.setColor(Color.BLUE.darker());
1072           g2.fillRect(225, 300, w, h);
1073           g2.setColor(Color.WHITE);
1074           g2.setFont(new Font("SansSerif", Font.BOLD, 24));
1075           g2.drawString("Press   1\n", 249, 361);
1076           g2.drawString("to  START", 243, 415);
1077           return;
1078        }
1079        g2.setColor(Color.BLUE.darker());
1080        for (int i = 0; i < 4; i++) {
1081           if (tilesCheck[i]) {
1082              g2.fillRect(i * w, tilesY[i], w, h);
1083           }
1084        }
1085     }
1086
1087  /*
1088  // Version 3
1089  public void drawTiles(Graphics g, boolean[] tilesCheck, int[] tilesY,
    boolean play) {
1090     Graphics2D g2 = (Graphics2D) g;
1091     g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
       RenderingHints.VALUE_ANTIALIAS_ON);
1092     int tileWidth = 150;
1093     int tileHeight = 200;
1094     Color tileColor = new Color(10,10,10,220);
1095     Color tileHighlight = new Color(255,255,255,45);
1096     if (!play) {
1097        g2.setColor(tileColor);
1098        g2.fillRoundRect(225, 300, 150, 200, 30, 30);
1099        g2.setColor(Color.white);
1100        g2.setFont(new Font("SansSerif", Font.BOLD, 40));
1101        g2.drawString("START", 245, 415);
1102        return;
1103     }
1104     for(int i = 0; i < 4; i++) {
1105        if(tilesCheck[i]) {
```

```
1106            int x = i * tileWidth;
1107            int y = tilesY[i];
1108            // tile rectangle g2.setColor(tileColor);
1109            g2.fillRoundRect(x, y, tileWidth, tileHeight, 30, 30);
1110            // highlight glow g2.setColor(tileHighlight);
1111            g2.fillRoundRect(x+10, y+10, tileWidth-20, tileHeight-20, 30,
                30);
1112         }
1113      }
1114   }
1115   */
1116
1117      /*
1118      // Version 2
1119      public void drawTiles (Graphics g, boolean[] tilesCheck, int[]
        tilesY, boolean play) {
1120         g.setColor(Color.BLACK);
1121         if(!play) {
1122            g.fillRect(150,350,150,200);
1123            g.setColor(Color.white);
1124            g.setFont(new Font(Font.DIALOG_INPUT,Font.BOLD,30));
1125            g.drawString("Start",180,460);
1126         }
1127         if(tilesCheck[0]) {
1128            g.fillRect(0,tilesY[0],150,YHEIGHT); // YHEIGHT = 200
1129         }
1130         if(tilesCheck[1]) {
1131            g.fillRect(150,tilesY[1],150,YHEIGHT);
1132         }
1133         if(tilesCheck[2]) {
1134            g.fillRect(300,tilesY[2],150,YHEIGHT);
1135         }
1136         if(tilesCheck[3]) {
1137            g.fillRect(450,tilesY[3],150,YHEIGHT);
1138         }
1139      }
1140
1141    */
1142      public void drawFoul(Graphics g, int foulPlace, int foulY) {
1143         g.setColor(Color.red);
```

```
1144         if (foulPlace == 0) {
1145            g.fillRect(0, foulY, 150, YHEIGHT);
1146         }
1147         if (foulPlace == 1) {
1148            g.fillRect(150, foulY, 150, YHEIGHT);
1149         }
1150         if (foulPlace == 2) {
1151            g.fillRect(300, foulY, 150, YHEIGHT);
1152         }
1153         if (foulPlace == 3) {
1154            g.fillRect(450, foulY, 150, YHEIGHT);
1155         }
1156      }
1157   }
1158
1159
1160
```