



Department of Electronics Engineering
Project Report

Title of the project :	Atari Breakout
SAP no and Name of the students:	1. 60001190037 Parth Dundh 2. 60001190033 Neel Karia 3. 60001190034 Neel Panchal
Tasks/Problem Statement:	1. Making the gameplay for a brick breaking game. 2. Developing the GUI for the game. 3. Adding generic features.
Code :	<ul style="list-style-type: none">• runfile.java <pre>import java.awt.Color; import java.awt.*; import Elements.*; import javax.swing.*; import java.awt.event.*; public class runfile{ JComboBox levelbox; JFrame welcome; ; static boolean reset = true; static boolean on = false; static bg_music bg; static JFrame window; static JButton start; static int lev_no; //boolean is_mute = false; public static void main (String[] args) { bg = new bg_music(); while (true) { if(reset) { runfile run = new runfile(); reset = false; bg.start(); } if(on) { game_window(); window.dispose(); on = false; reset = true; } } } }</pre>



Department of Electronics Engineering

```
}

        System.out.println(); // I DONT EVEN KNOW
    }
}

public runfile()
{
    String lev_arr[] = {"Demo","Level 1","Level 2","Level
3","Level 4","Level 5", "Level 6"};
    JButton exit = new JButton(new
ImageIcon("sprites\\exit2.jpg"));
    JButton start = new JButton(new
ImageIcon("sprites\\start_button.jpg"));
    JLabel lev_label = new JLabel(new
ImageIcon("sprites\\level.jpg"));
    JLabel title = new JLabel(new
ImageIcon("sprites\\welcome.png"));
    JLabel name_sap = new JLabel("Neel Karia - 60001190033
|| Parth Dundh - 60001190037 || Neel Panchal - 60001190034");

    levelbox = new JComboBox(lev_arr);

    //Hide drop down button of Combo box
    levelbox.setUI(new
javax.swing.plaf.metal.MetalComboBoxUI()
    {public void layoutComboBox(Container parent,
MetalComboBoxLayoutManager manager)
        {super.layoutComboBox(parent, manager);
        arrowButton.setBounds(0,0,0,0);
        }});

    welcome = new JFrame();
    welcome.setExtendedState(JFrame.MAXIMIZED_BOTH);
    welcome.setUndecorated(true);
    welcome.getContentPane().setBackground(new
Color(50,50,50));
    welcome.setLayout(null);
    welcome.setVisible(true);

    int w = welcome.getWidth();
    int h = welcome.getHeight();

    name_sap.setBounds(w/4,h-40,1000,20);
    name_sap.setFont(new Font("Verdana", Font.PLAIN, 18));
    name_sap.setForeground(new Color(253,188,4,120));

    title.setBounds(180, 200, 1200, 400);
    lev_label.setBounds(650,720,220,70);
    lev_label.setVisible(true);
    start.setBounds(400,720,220,70);
    bg.mute_button.setBounds(25*w/26,h/80,50,50);
    levelbox.setBounds(650,720,220,70);
    exit.setBounds(900,720,220,70);

    welcome.add(name_sap);
    welcome.add(bg.mute_button);
```



Department of Electronics Engineering

```
welcome.add(title);
welcome.add(lev_label);
welcome.add(start);
welcome.add(levelbox);
welcome.add(exit);

start.addActionListener(new start());
exit.addActionListener(new exit());

}

public static void game_window()
{
    bg.volume();
    window = new JFrame();

    window.setExtendedState(JFrame.MAXIMIZED_BOTH);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    window.setUndecorated(true);
    window.setTitle("ATRIA BREAKOUT");
    window.setVisible(true);

    int w = window.getSize().width;
    int h = window.getSize().height;

    Board game = new Board(w,h,lev_no, bg);

    window.addKeyListener(game.new AL());
    window.add(game);
    game.startgame();
    game.gameloop();

}

public class start implements ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e)
    {
        lev_no = levelbox.getSelectedIndex();
        on = true;

    }

}

public class exit implements ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);

    }

}

}
```



Department of Electronics Engineering

• Board.java

```
import java.awt.*;
import javax.swing.*;
import Elements.*;
import java.awt.event.*;

public class Board extends JPanel {

    static int w;
    static int h;
    boolean pause = true;
    boolean end = false;
    boolean reset = false;
    int lev_no;
    int numBrick;
    bg_music bg;

    Image canvas;
    paddle pad1;
    ball ball1;
    Menus menu = new Menus();
    JPanel panel= new JPanel();
    JLabel score_disp;
    JLabel start_game;
    levels lev1;

    Board(int w, int h, int lev_no, bg_music bg)
    {
        this.w= w;
        this.h = h;
        this.lev_no = lev_no;
        this.bg = bg;

        score_disp = new JLabel();
        start_game = new JLabel(new
ImageIcon("sprites\\start.jpg"));
        startgame();

        lev1 = new levels(6,9, lev_no);
        pad1 = new paddle((w-pad1.w)/2,750);
        ball1 = new ball((w-ball1.r)/2,600);
        canvas = createImage(w,h);
        lev1.generate();
        numBrick = lev1.numBrick();

        this.setBounds(0,0,w,h);
        this.setLayout(null);
        this.setBackground(new Color(50,50,50));

    }

    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        g.drawImage(canvas,0,0,this);
        pad1.draw(g);
```



Department of Electronics Engineering

```
        ball1.draw(g);
        lev1.draw(g);
    }
    public void gameloop()
    {
        while(true)
        {
            if(!pause&&!end)
            {
                ball1.motion();
                collisionCheck();
                score_display(lev1.score);
                repaint();
            }
            else
            {
                pause = menu.resume();
            }

            if (reset)
            {
                break;
            }
            try
            {
                Thread.sleep(2);
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
            reset = menu.re_check();
        }
    }
    public class AL extends KeyAdapter
    {
        @Override
        public void keyPressed(KeyEvent e)
        {
            if(!pause&&!end) // Paddle Motion
            {
                pad1.keyPress(e,w);
                repaint();
            }

            if (e.getKeyCode() == KeyEvent.VK_SHIFT)
            {
                if (!pause&&!end)
                {
                    pause = true;
                    menu.pausegame(w,h,bg);
                }
            }

            if (e.getKeyCode() == KeyEvent.VK_ENTER)
            {
                if (pause && !end)
```

**Department of Electronics Engineering**

```

        {
            start_game.setVisible(false);
            pause = false;
        }
    }
}

public void score_display(byte score)
{
    String s = String.valueOf(score);
    score_disp.setText(s);
    score_disp.setBounds(40,20,100,50);
    score_disp.setFont(new Font("Verdana", Font.BOLD, 40));
    score_disp.setForeground(new Color(255, 150, 150));
    add(score_disp);
}

public void startgame()
{
    start_game.setBounds(0,0,w,h);
    start_game.setFont(new Font("Verdana", Font.BOLD, 80));
    start_game.setBackground(new Color(0,0,0,0));
    start_game.setOpaque(false);
    start_game.setVisible(true);
    add(start_game);
}

public void collisionCheck()
{
    ball1.bounce_h(ball1.x + ball1.r >= w || ball1.x <= 0);

    ball1.bounce_v(ball1.y <= 0);
    ball1.paddle_hit((ball1.y + ball1.r > pad1.y && ball1.y
< pad1.y) && (ball1.x >= pad1.x && ball1.x + ball1.r <=
pad1.x+pad1.w));

    if (ball1.y < 600)
    {
        boolean[] side = lev1.BrickCollision(ball1.x,
ball1.y, ball1.r);
        //score = (byte) (score +
(side[0]||side[1]?1:0));
        ball1.bounce_v(side[0]);
        ball1.bounce_h(side[1]);
    }

    if ( ball1.y >= h || numBrick == lev1.score)
    {
        bg.stop();
        end = true;
        menu.endgame(numBrick == lev1.score);
    }
}
}

```



Department of Electronics Engineering

ELEMENTS PACKAGE

- ball.java

```
package Elements;
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class ball
{
    public int x;
    public int y;

    int step = 1;
    boolean east = false, south = false;
    public static int r = 36;

    public ball(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public void draw(Graphics g)
    {
        g.setColor(new Color(255, 255, 255));
        g.fillOval(this.x, this.y, this.r, this.r);
    }

    public void motion()
    {
        this.x = this.x + (east?step:-step);
        this.y = this.y + (south?step:-step);
    }

    public void bounce_h(boolean trig)
    {
        east = !(east ^ trig);
    }

    public void bounce_v(boolean trig)
    {
        south = !(south ^ trig);
    }

    public void paddle_hit(boolean trig)
    {
        if (trig)
        {
            south = false;
            sound paddle_hit = new sound
("sounds\\paddle_hit.wav");
        }
    }
}
```



Department of Electronics Engineering

- paddle.java

```
package Elements;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.KeyEvent;

public class paddle
{
    public int x,y;
    public static int w = 250;
    int h = 25;
    int step= 35;

    public paddle(int x, int y)
    {
        this.x = x;
        this.y =y;
    }

    public void keyPress(KeyEvent e, int w_screen)
    {
        if (e.getKeyCode() == KeyEvent.VK_LEFT && x > 0 )
        {
            this.x = this.x - step;
        }

        if (e.getKeyCode() == KeyEvent.VK_RIGHT && x + w <
w_screen)
        {
            this.x = this.x + step;
        }

        public void draw(Graphics g)
        {
            g.setColor(new Color(255, 150, 150));
            g.fillRoundRect(this.x, this.y, this.w, this.h,
this.h, this.h);
        }
    }
}
```




Department of Electronics Engineering

• **brick.java**

```
package Elements;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;

public class brick
{
    public int x;
    public int y;
    public int w=140;
    public int h=55;
    boolean state;

    public brick(int x, int y)
    {
        this.x = x;
        this.y = y;
        state = true;
    }

    public void draw(Graphics g, Color color )
    {
        if (state)
        {
            g.setColor(color);
            g.fillRect(this.x, this.y, w, h);
        }
    }
}
```



Department of Electronics Engineering

• levels.java

```
package Elements;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;

public class levels
{
    int row,col;
    brick[][] bricks;
    boolean[][] pattern;
    pattern pat = new pattern();
    public byte score = 0;

    public levels(int row ,int col, int pat_no)
    {
        this.row = row;
        this.col = col;
        this.pattern = pat.pat[pat_no];
        bricks = new brick[row][col];
    }
    public void generate()
    {
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < col; j++)
            {
                if(pattern[i][j])
                {
                    bricks[i][j] = new brick(j * 150 + 120, i * 65 + 85);
                }
            }
        }
    }
    public void draw(Graphics g)
    {
        Color color;
        for (int i = 0; i < row; i++)
        {
            color = new Color(90+10*i,90+10*i,195+10*i);

            for (int j = 0; j < col; j++)
            {
                if(pattern[i][j])
                {
                    bricks[i][j].draw(g,color);
                }
            }
        }
    }
}
```



Department of Electronics Engineering

```
public boolean[] BrickCollision(int ballx, int bally, int ballr)
{
    boolean[] returning = {false, false}; // {vertical col, horizontal col}

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if(pattern[i][j] && bricks[i][j].state)
            {
                if(bricks[i][j].x < ballx + ballr &&
bricks[i][j].x + bricks[i][j].w > ballx) // Vertical Collision
                {
                    if (bally + ballr == bricks[i][j].y ||
bally == bricks[i][j].y + bricks[i][j].h )
                    {
                        sound brick_hit = new
sound("sounds\\brick_hit_v.wav");
                        bricks[i][j].state = false;
                        returning[0] = true;
                        score++;
                    }
                }

                if(bricks[i][j].y < bally + ballr &&
bricks[i][j].y + bricks[i][j].h > bally) // Horizontal Collision
                {
                    if (ballx +ballr == bricks[i][j].x
|| ballx == bricks[i][j].x + bricks[i][j].w )
                    {
                        sound brick_hit = new
sound("sounds\\brick_hit.wav");
                        bricks[i][j].state = false;
                        returning[1] = true;
                        score++;
                    }
                }

                if(ballx +ballr == bricks[i][j].x || ballx
== bricks[i][j].x + bricks[i][j].w)
                { // Corner Collision
                    if(bally + ballr == bricks[i][j].y
|| bally == bricks[i][j].y + bricks[i][j].h)
                    {
                        sound brick_hit = new
sound("sounds\\brick_hit_v.wav");
                        bricks[i][j].state = false;
                        returning[0] = true;
                        returning[1] = true;
                        score++;
                    }
                }
            }
        }
    }
}
```



Department of Electronics Engineering

```
}

return returning;
}

public byte numBrick()
{
    byte count = 0;
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if(pattern[i][j] && bricks[i][j].state)
            {
                count++;
            }
        }
    }
    return count;
}
}
```

**Department of Electronics Engineering**● **Menus.java**

```
package Elements;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Menus extends JFrame {

    boolean pause = true;
    boolean reset = false;
    JLabel pause_img;
    bg_music bg;

    public void endgame(boolean won)
    {
        JLabel l1;
        JLabel l2;

        if (won)
        {
            l1 = new JLabel(new
ImageIcon("sprites\\finished.jpeg"));
            sound win_sound = new sound("sounds\\win.wav");
        }

        else
        {
            l1 = new JLabel(new ImageIcon("sprites\\me.jpg"));
            l2 = new JLabel(new ImageIcon("sprites\\lose.jpg"));
            l2.setBounds(130,20,500,100);
            add(l2);
            sound lose_sound = new sound("sounds\\lose.wav");
        }
        JButton reset = new JButton(new
ImageIcon("sprites\\menu.jpg"));
        JButton exit = new JButton(new
ImageIcon("sprites\\exit2.jpg"));

        reset.setBounds(280,575,220,70);
        exit.setBounds(280,655,220,70);
        l1.setBounds(90,120,600,440);

        exit.addActionListener(new BL());
        reset.addActionListener(new RL());

        setOpacity(1f);
        setUndecorated(true);
        getContentPane().setBackground(Color.BLACK);
        setLayout(null);
        setVisible(true);
        setBounds(430,100,800,750);

        add(reset);
        add(exit);
        add(l1);
    }
}
```



Department of Electronics Engineering

```
        setAlwaysOnTop(true);
    }

    public void pausegame(int w, int h, bg_music bg)
    {
        pause_img = new JLabel(new
ImageIcon("sprites\\paused.jpg"));
        pause = true;
        setLocation(0,0);
        setSize(w,h);
        setUndecorated(true);
        getContentPane().setBackground(Color.BLACK);
        setOpacity(0.5f);
        setLayout(null);
        setVisible(true);
        pause_img.setBounds(0,0,w,h);
        bg.mute_button.setBounds(25*w/26,h/80,50,50);
        bg.mute_button.addKeyListener(new AL());
        add(pause_img);
        add(bg.mute_button);

        this.bg = bg;

        this.addKeyListener(new AL());
        setAlwaysOnTop(true);
    }

    public boolean resume()
    {
        return pause;
    }

    public boolean re_check()
    {
        return reset;
    }

    public class AL extends KeyAdapter
    {
        @Override
        public void keyPressed(KeyEvent e)
        {
            if (e.getKeyCode() == KeyEvent.VK_SHIFT)
            {
                pause_img.setVisible(false);
                dispose();
                pause = false;
            }

            if (e.getKeyCode() == KeyEvent.VK_ESCAPE)
            {
                System.exit(0);
            }
        }
    }
}
```



Department of Electronics Engineering

```
public class BL implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
}
public class RL implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        dispose();
        reset = true;
    }
}
}
```



Department of Electronics Engineering

- **sound.java**

```
package Elements;

import java.io.File;
import java.io.IOException;
import javax.sound.sampled.*;

public class sound {

    public sound(String loc)
    {

        File file = new File(loc);

        try
        {
            AudioInputStream audio =
AudioSystem.getAudioInputStream(file);
            Clip clip = AudioSystem.getClip();
            clip.open(audio);
            clip.start();

            // TODO Auto-generated catch blocks
        }
        catch (UnsupportedAudioFileException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        catch (LineUnavailableException e)
        {
            e.printStackTrace();
        }
    }

}
```




Department of Electronics Engineering

• bg_music.java

```
package Elements;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.*;
import javax.swing.ImageIcon;
import javax.swing.JButton;

public class bg_music {

    Clip clip;
    AudioInputStream audio;
    boolean mute = false;
    public JButton mute_button = new JButton(new
ImageIcon("sprites\\unmuted.jpg"));

    public bg_music()
    {
        mute_button.addActionListener(new mute());
        File file = new File("sounds//bg.wav");

        try
        {

            audio = AudioSystem.getAudioInputStream(file);
            clip = AudioSystem.getClip();
            clip.loop(Clip.LOOP_CONTINUOUSLY);
            clip.open(audio);
            // TODO Auto-generated catch block
        }
        catch (UnsupportedAudioFileException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        catch (LineUnavailableException e)
        {
            e.printStackTrace();
        }
    }

    public void start()
    {

        if (!mute)
        {
            clip.start();
        }
    }

    public void stop()
    {

```



Department of Electronics Engineering

```
clip.stop();
}

public void volume()
{
    FloatControl gainControl = (FloatControl)
clip.getControl(FloatControl.Type.MASTER_GAIN);
    gainControl.setValue(-6.0f);
}

public void mute()
{
    if(mute)
    {
        clip.start();
        mute_button.setIcon(new
ImageIcon("sprites\\unmuted.jpg"));
    }
    else
    {
        clip.stop();
        mute_button.setIcon(new
ImageIcon("sprites\\muted.jpg"));
    }
    mute = !mute;
}

public class mute implements ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e)
    {
        mute();
    }
}
}
```

Department of Electronics Engineering

- `patterns.java`

```
package Elements;
```

```
public class pattern {
```

```
boolean pat [][][] = {
```

```
{ //Demo
```

```
{false, false , false , false, false, false, false, false, false},
{false, false , false , false, false, false, false, false, false},
{false, false , false , false, false, false, false, false, false},
{false, false , false , false, false, false, false, false, false},
{true, false , false , true, false, false, false, false, false},
{false, false , false , false, false, false, false, false, false}
},
```

```
{ //Level1
```

```
{true, true , true , true, true, true , true, true, true},
{false, true , true , false, false, true, true, false, true},
{true, true, true, false, false, true, true , true, true},
{true, true , true, true, true, true, true, true, true},
{false, false, true , true, true, true, false ,false, true},
{false, false, true , true, true, true, false ,false, true}
},
```

```
{ //Level2
```

```
{true , true , true, true, true , true, true, true , true},
{true , false, false, false, false, false, false, false, true},
{true , true, true, true, true, true , true , true , true},
{true , false, false, true, true, true, false, false, true},
{true , false, false, false, true, false, false, false, true},
{true , false, false, false, true, false, false, false, true}
},
```

```
{ //Level3
```

```
{false, false, false, true, true, true, false, false, false},
{true, false, true, true, false, true, true, false, true},
{true, true, true, false, false, false, true, true, true},
{true, true, false, false, false, false, false, true, true},
{true, false, true, false, false, false, true, false, true},
{false, false, false, true, true, true, false, false, false}
},
```

```
{ //Level4
```

```
{true, true, true, true, true, true, true, true, true},
{true, true, true, true, false, true, true, true, true},
{true, true, true, false, true, false, true, true, true},
{true, true, false, true, true, true, false, true, true},
{true, false, false, true, true, true, false, true, true},
{true, false, false, false, true, false, false, true, true}
},
```

```
{ //Level15
```

```
{false, false, false, false, false, false, false, false, false},
{false, true, true, false, false, false, true, true, false},
{false, false, true, true, true, true, true, false, false},
{false, false, false, true, true, true, false, false, false},
{false, false, false, false, true, false, false, false, false},
```

**Department of Electronics Engineering**

	<pre>{false, true , true, true, true, true, true, true, false} }, { //Level6 {false, false , false, false, false, false, false, false, false}, {false, false , false , true , true, true, false, false, false}, {false, false , true , false, true, false, true, false, false}, {false, false , true , true , true, true, true, false, false}, {false, false , true , false, true, false, true, false, false}, {false, false , false, true , true, true, false, false, false}, } }; }</pre>
Explanation of the code :	<ul style="list-style-type: none"> File Structure of the Project <pre>~/Breakout _ sounds (contains SFX & music) _ sprites (contains images for GUI display) _ src _ Elements (package) _ ball.java _ paddle.java _ brick.java _ levels.java _ Menus.java _ sound.java _ bg_music.java _ patterns.java _ Board.java _ runfile.java</pre> <p>Link to git - https://github.com/french-fri/Breakout</p> <ul style="list-style-type: none"> runfile <p>The runfile class is the starting point of the game. The class consists of the main method, a constructor that initializes the main menu GUI window, "start_game" method that starts up the gameplay by calling the Board class, boolean "reset" used to return to the main menu after game ends, boolean "on" that indicates if the gameplay is active, bg_music object "bg" that initializes and controls background music and int "lev_no" to store the level index selected by the user.</p> <p>The main function makes a while(true) loop to keep the game running until termination by user exit. Two if statements with "reset" and "on" booleans manage the generation of the main menu window and the gameplay window. Turning the "reset"</p>

Department of Electronics Engineering

boolean true makes a new **runfile** object **"run"** starting the **constructor** and initializing the main manu **JFrame** object named **"welcome"** while turning the **"on"** boolean true disposes the **"welcome"** **JFrame** and calls the **"game_window"** method.

The GUI window consists of three **JButtons** (**start**, **exit** and **mute**) a **Jcombobox** **"levelbox"** and **Jlabel** **"title"** displaying the title at the center of the window. We create an array of strings **"lev_arr"** containing names of the levels which is displayed in the **"levelbox"** and used for level selection. The selected index of the **"levelbox"** is stored in int **"lev_no"**. Classes **"start"** and **"exit"** are an extension of **ActionListener** class from **"java.awt.event"** that detect when the respective button is pressed. We **override** the **"actionPerformed"** method of **Actionlistener** to call the required method on button press. The **"exit"** class calls **"system.exit(0)"** to terminate the program whereas the **"start"** button passes the level number to the game through int **"lev_no"** and sets boolean **"on"** to true, hence starting gameplay.

The **"game_window"** method makes a new **Board** object **"game"** and a **JFrame** called **"window"** to display the board. It adds a new key listener **"AL"** to the window and initializes the gameplay from the **Board** class.

The mute button calls the **"mute"** method from the object **"bg"**. The icons for the buttons and title screen are stored in the **"sprites"** folder of the project.

● **Board**

The **Board** class is an extension of the **JPanel** class, it generates and displays the gameplay. The constructor calls and stores the screen resolutions, int **"lev_no"** and background_music object **"bg"** from the runfile. The constructor generates a **ball ball1**, **paddle pad1**, **levels** object **lev1** and sets its own bounds and background colour.

The **"paintcomponent"** is an inbuilt method of **"java.awt.Graphics"** that displays all the elements on a canvas on the extended **jPanel** **"Board"**.

The **"gameloop"** method sets up the game loop in which the gameplay is executed. The **"gameloop"** calls the motion method from **ball1**, **"collisionCheck"** method to check for collisions, **"score_display"** method to keep the **"score_disp"** updated and the inbuilt **"repaint"** method from **"java.awt.Graphics"** to refresh the screen. The **"gameloop"** also calls the **"re_check"** method from the **"Menus"** class to constantly check for a request to reset the game.

class **"AL"** is an extension of **"keyAdaptar"** class and it detects the keyboard keys pressed by the user. We **override** the **"keyPressed"** method from the parent class to call required

**Department of Electronics Engineering**

methods when the player uses the keyboard. The `pad1.keyPressed` method is called for paddle motion on `arrow` key press while `shift` key is detected for calling the `"pausegame"` method from `"Menus"` class, and `enter` is detected to begin the game. `"score_display"` method is used to display and update the score at the top left corner of the screen using a `Jlabel` `"score_disp"`.

The `"start_game"` method is used to display the "press enter to start" message at the start of the game using a `Jlabel` `"start_game"`.

The `"collisionCheck"` method checks for ball collisions and calls the appropriate functions on detection. When the ball reaches the ends of the screen, `"bounce_v"` & `"bounce_h"` methods are triggered to move the ball in the appropriate direction. If the ball is at the upper half of the screen, `"lev1.BrickColision"` method is called to check for collisions with bricks. the function returns an array of two booleans named `"side"` where `side[0]` is for **vertical collision** detection and `side[1]` is for **horizontal** collision detection.

```
side[] = {vertical collision, horizontal collision}
```

if horizontal & vertical both are **true**, the collision is detected as **corner detection** and if **none** are **true**, it is detected as **no collision**.

The `"collisionCheck"` also checks collisions with the bottom of the screen for a loss or if score (`int lev1.score`) becomes equal to the initial number of bricks (`int numBrick`) for a win to call the `"menu.endgame"` method with the appropriate message.

Elements Package

- **ball**

This class contains the methods for generating the ball and deciding its motion on our Board. The `"draw()"` method makes use of `"java.awt.Graphics"` library for "drawing" the ball. An integer `"step"` is declared to decide how much the ball will move in each step. The `"motion"` method increments the movement of the ball by `"step"` units on the basis of booleans `"east"` which when **true** means the ball is moving left and `"south"` which when **true** means the ball is moving downwards. The `"bounce_v"` and `"bounce_h"` methods are used to check whether the collision of the ball on the bricks is vertical or horizontal. The `"trig"` boolean decides the direction of movement of the ball on the basis of these collisions. If in `"bounce_h"` trig is **true** the ball continues to move **east** if

Department of Electronics Engineering

false it changes direction to **west**. If in **"bounce_v"**, trig is true the direction of the ball changes to **!south**.

- **paddle**

This class contains the methods for generating the paddle and deciding its motion on our Board. The **"draw()"** method makes use of **"java.awt.Graphics"** library for **"drawing"** the paddle. The **"keyPress"** method calls **"KeyEvent"** from the **"keyAdapter"** in class **"Board"** for the movement of the paddle. When the **"Left"** key is pressed the paddle moves to the left till the paddle is not touching the leftmost border of the screen. Similarly when the **"right"** key is pressed the same concept is used for the rightmost border .

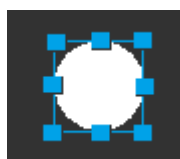
- **brick**

This class contains the height and width of the bricks to be generated on Board. The **"draw()"** method makes use of **"java.awt.Graphics"** library for **"drawing"** the brick according to the color passed to the method.

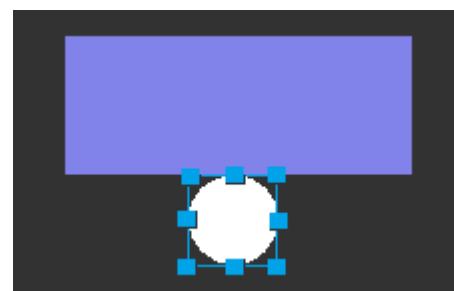
- **levels**

This class is used to generate a brick pattern for given **"lev_no"** according to the **"pat"** array. The **"draw()"** method makes use of **"java.awt.Graphics"** library for **"drawing"** the pattern of bricks. Additionally the class contains the main collision system of the game ball and also keeps the count of the number of bricks generated which is used for score. The **"generate()"** method creates a 2D array of bricks on the basis of rows and columns. This 2D array is created as an object of **"brick"** class.

The **"BrickCollision"** method is the collision detection system that handles collision detection of the ball with the bricks. The method iterates through every single brick one by one and compares the borders of the bricks with the ball. If any collision points overlap with the surface of a brick, it detects it as a collision, sets the state of the respective brick to false and sends the detection back to the board in the form of the 2d array mentioned in the board class explanation.



**Collision points
around the ball**



**collision points
overlap with brick**

Department of Electronics Engineering

● Menus

The Menus class contains methods for displaying pause screen and game over screen. When the game ends, a boolean “won” is passed to the “endgame()” method that indicates if a player lost or won. According to the state of the boolean the corresponding JLabel is created in the “endgame()” method’s JFrame and is displayed when the game ends. Each of the two “game over” menus have the option to either return to the main menu or exit the game. For restarting the game we declare a “reset” boolean whose initial state is false. The Classes “BL” and “RL” are an extension of ActionListener class from “java.awt.event” that detect when the “exit” and “menu” button is pressed. The “exit” button calls “system.exit(0)” to terminate the program whereas the “menu” button changes the state of “reset” to true.

For “pausegame()” a “pause” boolean is declared whose initial state is true. A class “AL” which is an extension of KeyAdapter detects when a particular key is pressed. When shift key is pressed, the “pausegame” window is disposed and state of “pause” boolean changes to false. When in the pause menu, if the player presses “Esc” key, the KeyAdapter method calls “system.exit(0)” to terminate the program.

● sound

This class is used to obtain the audio file from the mentioned directory in the system saved in String “loc”. We generate a Clip “clip”, and AudioInputStream called “audio” and a File “file” to call the WAV file from the “sounds” folder of the project. We use the “clip.start()” method to play the audio.

● bg_music

This class is responsible for calling and handling the background music of the game. We generate a Clip “clip”, an AudioInputStream called “audio” and a File “file” to call the WAV file from the “sounds” folder of the project. A boolean “mute” and JButton “mute_button” are also generated to handle the mute - unmute feature of the game. The “start” method checks if mute is false and then plays the music using the “clip.start” method. The “stop” method is used to stop the background music when the game ends.

The “volume” method reduces the volume of the music when gameplay starts.

The “mute” function toggles the boolean “mute” and changes the icon of the mute button appropriately while starting or stopping the music.

the ActionListener listens for when the “mute_button” is pressed and calls the “mute” method.



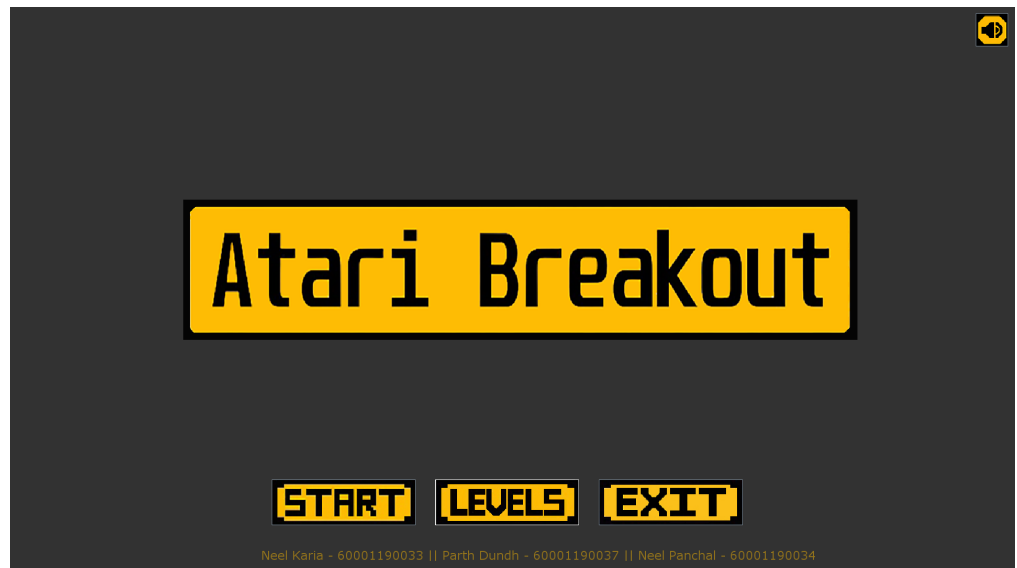
Department of Electronics Engineering

- **pattern**

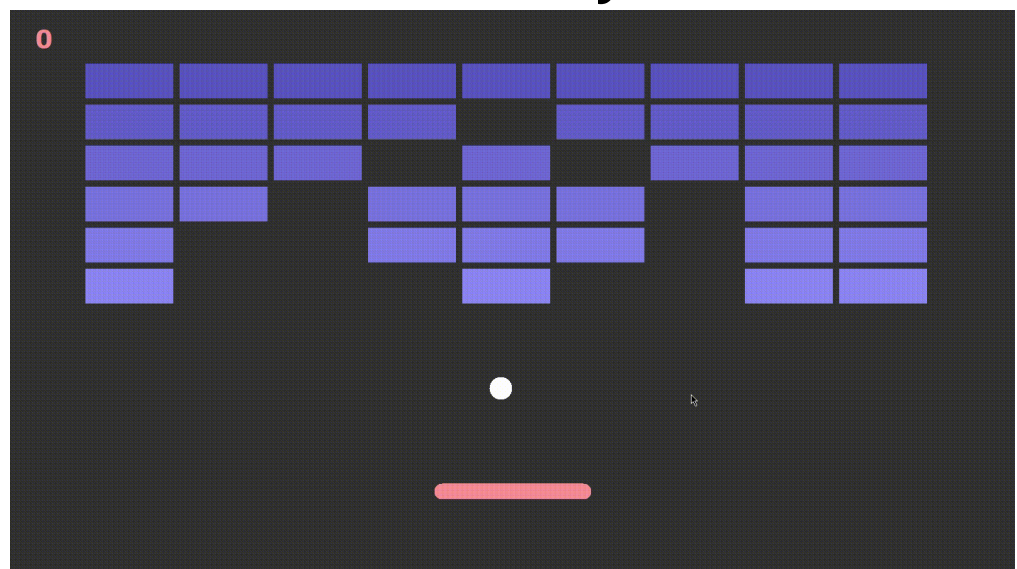
The “**pattern**” class contains a boolean array “**pat**” that contains the designs of all the levels. Every “**true**” element in the “**pat**” array is generated as a brick. This generates a pattern of bricks.

Output :

Main Menu



GamePlay





Department of Electronics Engineering

Game Pause

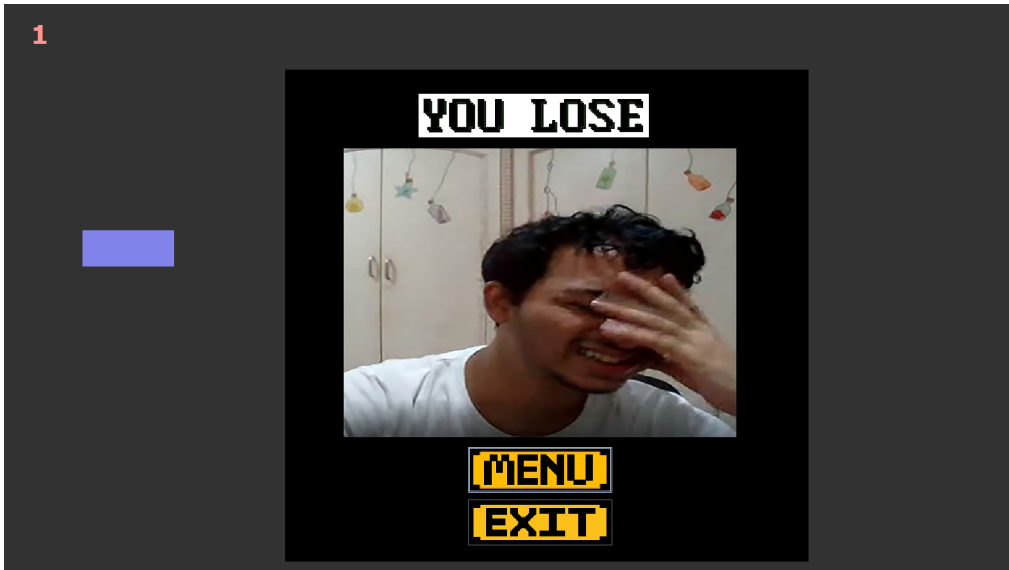


Win Screen





Department of Electronics Engineering

	<p style="text-align: center;">Lose Screen</p> 
<p>Future scope modifications :</p>	<ul style="list-style-type: none">• Concept of “Lives”• Bug fixes• Performance optimization• Saving High scores & having a global leaderboard• Choices in difficulty, speed, size etc.• Allowing users to design levels