**Name –** Ashutosh Agarwal

**Roll No** – 1310110085

**CODE**

package j2me\_assignment;

import javax.microedition.midlet.\*; //header file for functioning of Midlet import javax.microedition.lcdui.\*;

import javax.microedition.lcdui.game.\*; //header file for functioning of GameCanvas

// A Bouncing Square in MIDP

public class Midlet extends MIDlet

{

// Allocate a GameCanvas, set it to the current display, and start the game thread public void startApp()

{

GameMain game = new GameMain(); // GameMain extends GameCanvas for the game UI

Display.getDisplay(this).setCurrent(game);

new Thread(game).start(); // GameMain implements Runnable to run the game thread

}

public void pauseApp()

{

}

public void destroyApp(boolean unconditional)

{

}

// The GameMain class is designed as an inner class, which extends GameCanvas for drawing

// the game graphics, and implements Runnable to run the game logic in its own thread.

class GameMain extends GameCanvas implements Runnable

{

private int ctrx, ctry; // (x,y) of the center of the square private double velocityX = 2, velocityY = 2; // speed in x and y direction private static final int SIDE = 15; // side length of our square

private static final int DELAY = 30; // game update interval in millisec onds int canvasWidth = getWidth();

int canvasHeight = getHeight();

// Bounds of the center of the square int minlimitX = SIDE;

int minlimitY = SIDE ;

int maxlimitX = canvasWidth - SIDE;

int maxlimitY = canvasHeight - SIDE;

// Constructor public GameMain()

{

super(false); //used to prevent the suppressing of keyevents. If super(true) was called, only main game keys (2,4,6,8) would work.

// Initializes the Square

ctrx = canvasWidth / 2; // place the square in the center of the canvas ctry = canvasHeight / 2;

}

// Running behavior of the game thread public void run()

{

// Retrieve the off-screen graphics buffer for graphics drawing

Graphics g = getGraphics();

// Game loop while (true)

{

int keyState = getKeyStates(); // used because "UP" and "DOWN" arrow are not identified in GameAction using KeyPressed.

if((keyState & UP\_PRESSED) != 0)

{

velocityX += 0.4\*velocityX ; // increasing velocity in X and Y direction proportionally velocityY += 0.4\*velocityY ;

if((velocityX == 0) && (velocityY == 0)) // to increase the velocity incase, it becomes

0.

{

velocityX = 2 ;

velocityY = 3 ;

}

if(velocityY >= 432) // reinitialize velocity in case the velocity becomes vigorous.

432 was calculated, the max speed needed in Y direction.

{

velocityX = 1 ;

velocityY = 1 ;

}

}

if((keyState & DOWN\_PRESSED) != 0)

{

velocityX -= 0.4\*velocityX ; // decreasing velocity in X and Y direction proportionally velocityY -= 0.4\*velocityY ;

}

if((keyState & FIRE\_PRESSED) != 0)

{

velocityX = 0 ; // decreasing velocity in X and Y direction proportionally

velocityY = 0 ;

}

// Update the square's position on key activity. ctrx += velocityX;

ctry += velocityY;

// Check if the square hits the screen boundaries. 'Reflect' the square if so.

if (ctrx > maxlimitX) //if square hits the rig ht boundary, reflect it.

{

ctrx = maxlimitX;

velocityX = -velocityX;

}

else if (ctrx < minlimitX) //if square hits the left boundary, reflect it.

{

ctrx = minlimitX;

velocityX = -velocityX;

}

if (ctry > maxlimitY) //if square hits the bottom boundary, reflect it.

{

ctry = maxlimitY;

velocityY = -velocityY;

}

else if (ctry < minlimitY) //if square hits the top boundary, reflect it.

{

ctry = minlimitY;

velocityY = -velocityY;

}

// Set Screen colour to white as instruted in the Demo g.setColor(255,255,255);

g.fillRect(0, 0, canvasWidth, canvasHeight);

// Set Square colour to blue as instruted in the Demo

g.setColor(0,0,255);

g.fillRect(ctrx - SIDE, ctry - SIDE, 2 \* SIDE, 2 \* SIDE);

// Predefined Function used to flush the drawing and update the visible component with the rendered frame.

flushGraphics();

// Provide delay in currently executing thread to manage the gamespeed that must produce a fixed frame-rate.

try

{

Thread.sleep(DELAY);

}

catch (InterruptedException e)

{

}

}

}

protected void keyPressed(int keyCode) //keyPressed used for easy identification of key activities.

{

switch(keyCode)

{

case 49 : if(Math.abs(velocityX)>0) //move in direction of NUM\_1

{

velocityX = -(Math.abs(velocityX)) ;

velocityY = -(Math.abs(velocityX)) ;

}

else if(Math.abs(velocityY)>0)

{

velocityX = -(Math.abs(velocityY)) ;

velocityY = -(Math.abs(velocityY)) ;

}

else if((Math.abs(velocityX)==0)&&(Math.abs(velocityY)==0))

{

velocityX = -1 ;

velocityY = -1 ;

}

break ;

case 50 : if(Math.abs(velocityX)>0 ) //move in direction of NUM\_2

{

velocityY = -Math.abs(velocityX);

velocityX = 0 ;

}

else if(velocityY>0) velocityY = -velocityY ; else if(velocityY==0) velocityY = -2 ;

break ;

case 51 : if(Math.abs(velocityX)>0) //move in direction of NUM\_3

{

velocityX = +(Math.abs(velocityX)) ;

velocityY = -(Math.abs(velocityX)) ;

}

else if(Math.abs(velocityY)>0)

{

velocityX = +(Math.abs(velocityY)) ;

velocityY = -(Math.abs(velocityY)) ;

}

else if((Math.abs(velocityX)==0)&&(Math.abs(velocityY)==0))

{

velocityX = +1 ;

velocityY = -1 ;

}

break ;

case 52 : if(Math.abs(velocityY)>0 ) //move in direction of NUM\_4

{

velocityX = -Math.abs(velocityY);

velocityY = 0 ;

}

else if(velocityX>0) velocityX = -velocityX ; else if(velocityX==0) velocityX = -2 ;

break ;

case 53 : velocityX = 0 ; //Stop the execution velocityY = 0 ;

break ;

case 54 : if(Math.abs(velocityY)>0 ) //move in direction of NUM\_6

{

velocityX = +Math.abs(velocityY);

velocityY = 0 ;

}

else if(Math.abs(velocityX)>0) velocityX = +Math.abs(velocityX) ; else if(Math.abs(velocityX)==0) velocityX = +2 ;

break ;

case 55 : if(Math.abs(velocityX)>0) //move in direc tion of NUM\_7

{

velocityX = -(Math.abs(velocityX)) ;

velocityY = +(Math.abs(velocityX)) ;

}

else if(Math.abs(velocityY)>0)

{

velocityX = -(Math.abs(velocityY)) ;

velocityY = +(Math.abs(velocityY)) ;

}

else if((Math.abs(velocityX)==0)&&(Math.abs(velocityY)==0))

{

velocityX = -1 ;

velocityY = +1 ;

}

break ;

case 56 : if(Math.abs(velocityX)>0 ) //move in direction of NUM\_8

{

velocityY = +Math.abs(velocityX);

velocityX = 0 ;

}

else if(Math.abs(velocityY)>0) velocityY = +Math.abs(velocityY) ; else if(Math.abs(velocityY)==0) velocityY = +2 ;

break ;

case 57 : if(Math.abs(velocityX)>0) //move in direction of NUM\_9

{

velocityX = +(Math.abs(velocityX)) ;

velocityY = +(Math.abs(velocityX)) ;

}

else if(Math.abs(velocityY)>0)

{

velocityX = +(Math.abs(velocityY)) ;

velocityY = +(Math.abs(velocityY)) ;

}

else if((Math.abs(velocityX)==0)&&(Math.abs(velocityY)==0))

{

velocityX = +1 ;

velocityY = +1 ;

}

break ;

default : // Do Nothing

}

repaint() ; //repaint method called on identification of the pressed key, followed by the action.

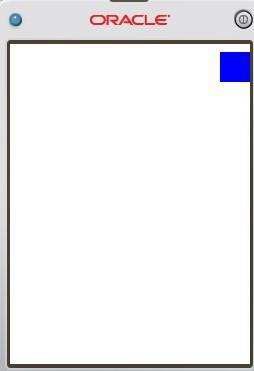
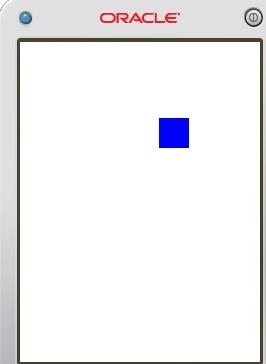
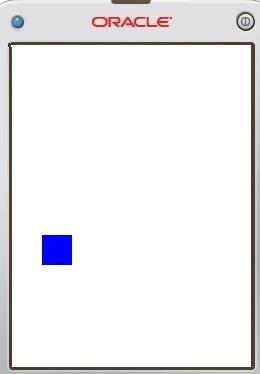
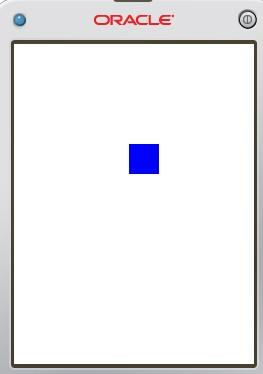
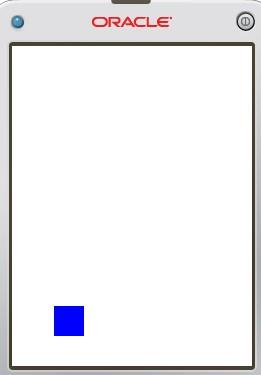
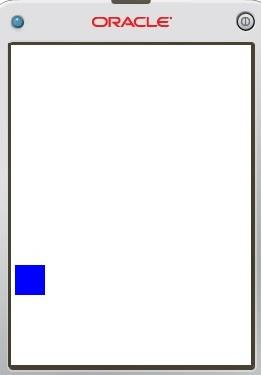
}

}

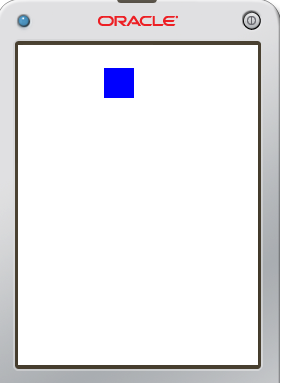
}

**SCREENSHOTS**

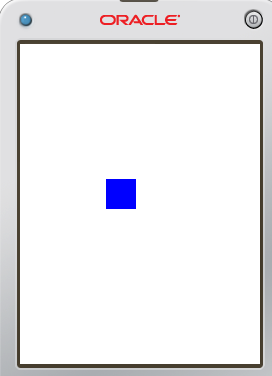
Reflection of the square onto the left ball has been shown in the Screenshots below. The square approached the left wall from the bottom, hit the wall and got reflected into the other direction.



On Pressing Key 1



On Pressing Key 8



On Pressing key 6

