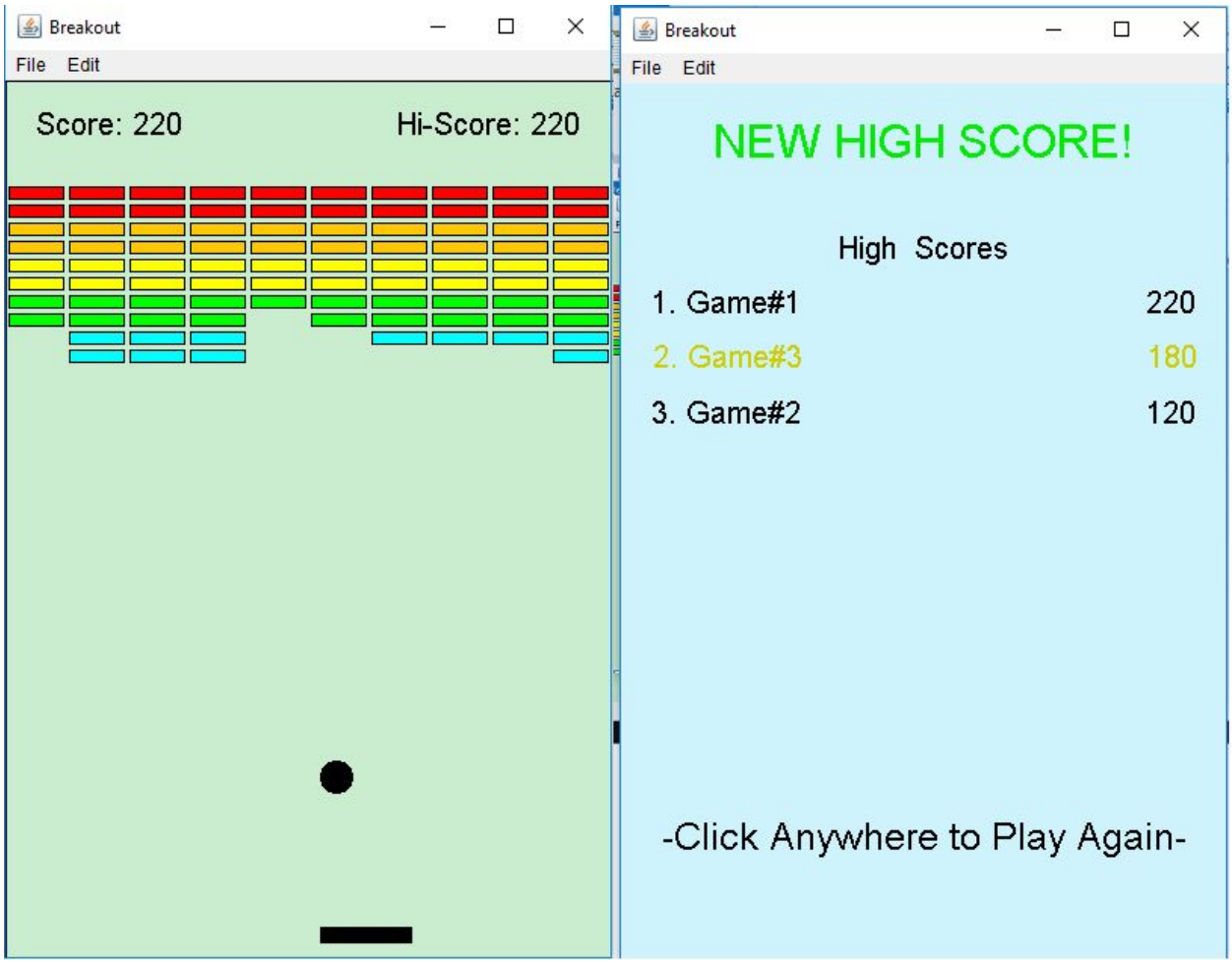
LBYCP12

Data Structures and Algorithm Analysis Laboratory



Laboratory Activity 2

Breakout Game

By

Chan, Patrick Matthew J, LBYCP12-EQ1

INTRODUCTION

In this activity, the students will try to recreate the classic arcade game of Breakout using Java, and to integrate the usage and implementation of a custom List ADT into the program.

OBJECTIVES

- To learn how to tackle a complicated programming assignment by by dividing it into small manageable parts
- To create a program that relies on constants, rather than simply Hardcoded values, so that various parameters can readily be changed before runtime.
- To introduce event-driven programming, by using listeners in order to successfully use the mouse to control the paddle in-game
- To learn how to utilize a List ADT in storing game data
- To integrate game design with ADT implementation

MATERIALS

1. Java SE Development Kit (JDK) 8
2. NetBeans integrated development environment (IDE)
3. `acm.jar` by the ACM Java Task Force
4. Bounce.au sound file

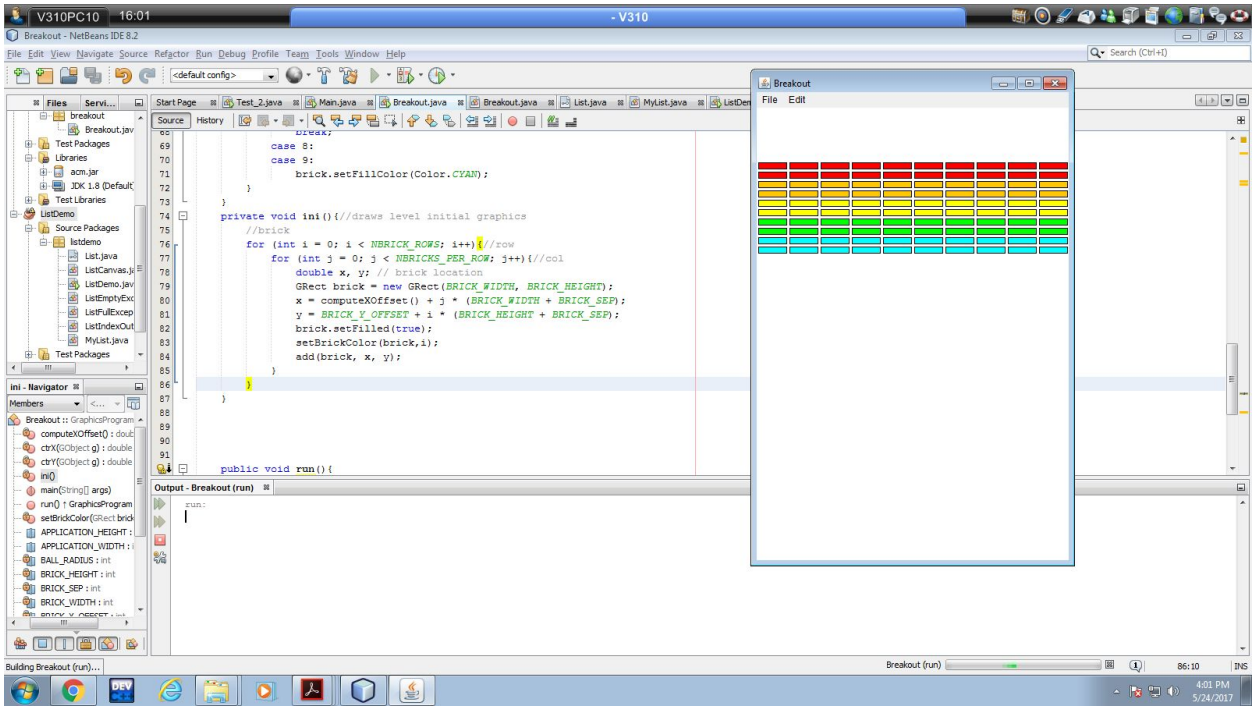
PROCEDURE

1. Use the provided starter file for the given game parameter constants
2. Use the given constants in the code so that changing the values would effectively change the needed parameters in the program
3. Setting up the Bricks
    a. Use for loops to generate rows of bricks, and determine the spacing between them
    b. Color these bricks according to row
4. Create a Paddle
    a. Create a GRect to serve as the paddle

    b. Add mouse listeners to the program to be able to track the mouse
    c. Program the paddle so that it follows the x position of the mouse on the canvas
5. Creating a Ball and Making it Bounce off walls
    a. Create a GOval for the Ball
    b. Use a random number generator to set the initial velocity of the ball
    c. Create a loop that moves the ball to these velocities for every iteration
    d. Set the program to always check the four corners of the bounding rectangle of the ball to check for collisions
        i. Use getElementAt method to check these corners one by one, until a collision is found (null means that no collider is found at the particular corner)
        ii. Return this collider as type GObject
    e. In accordance with the law of reflection, set the ball's velocity vx=-vx for hits on the left or right side of the ball, while vy=-vy, for the top or bottom side of the ball
    f. If the collider is a brick, remove that brick from the canvas
6. Score Keeping
    a. Create a new GLabel to display the score
    b. Compute the score by checking the color of the brick, then adding the appropriate score
7. High Scores
    a. Add conditions that stop the game on whether all brick are cleared, or the player runs out of turns/lives
    b. Using a List ADT, create a list to keep track of the scores
    c. Whenever a game is over, let the program check on where the new score should be inserted into the list.
    d. Make sure the the program only keeps track of a limited number of Hi-Scores, and eliminates the excess "low-scorers" from the list
8. Add some finishing touches, some improvements, and test run the program.

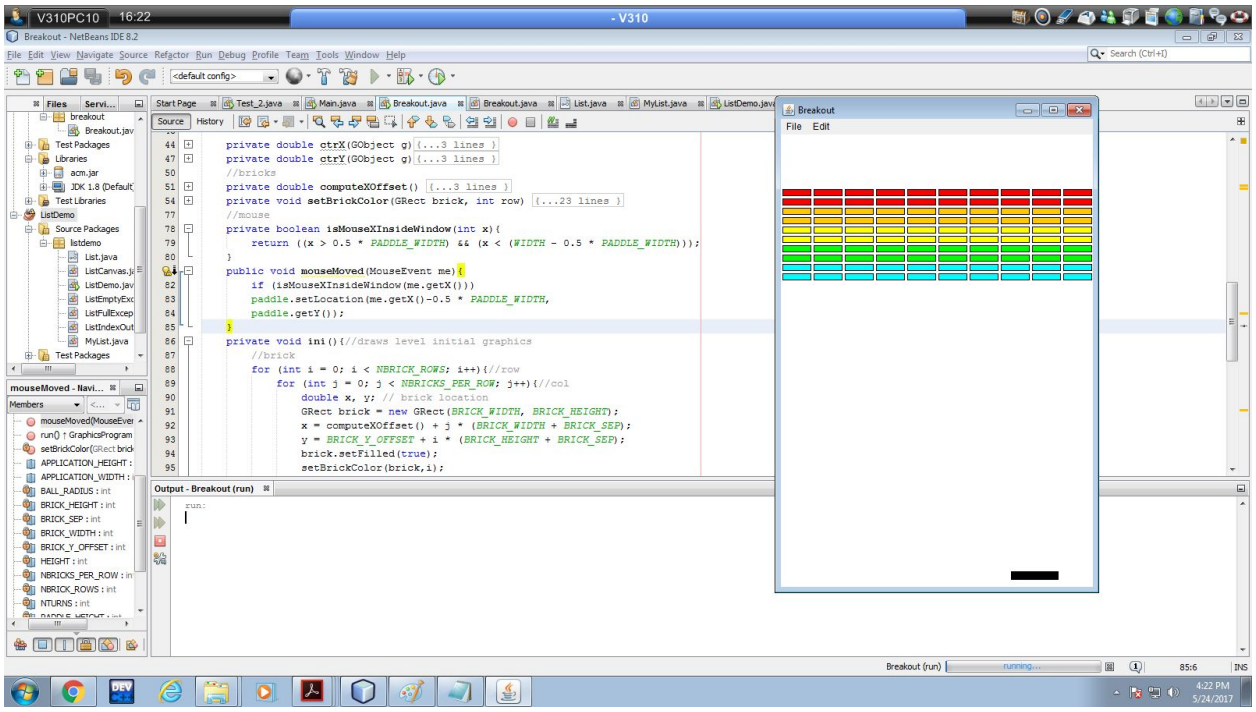
RESULTS AND DISCUSSION

a) Creating the Brick Wall



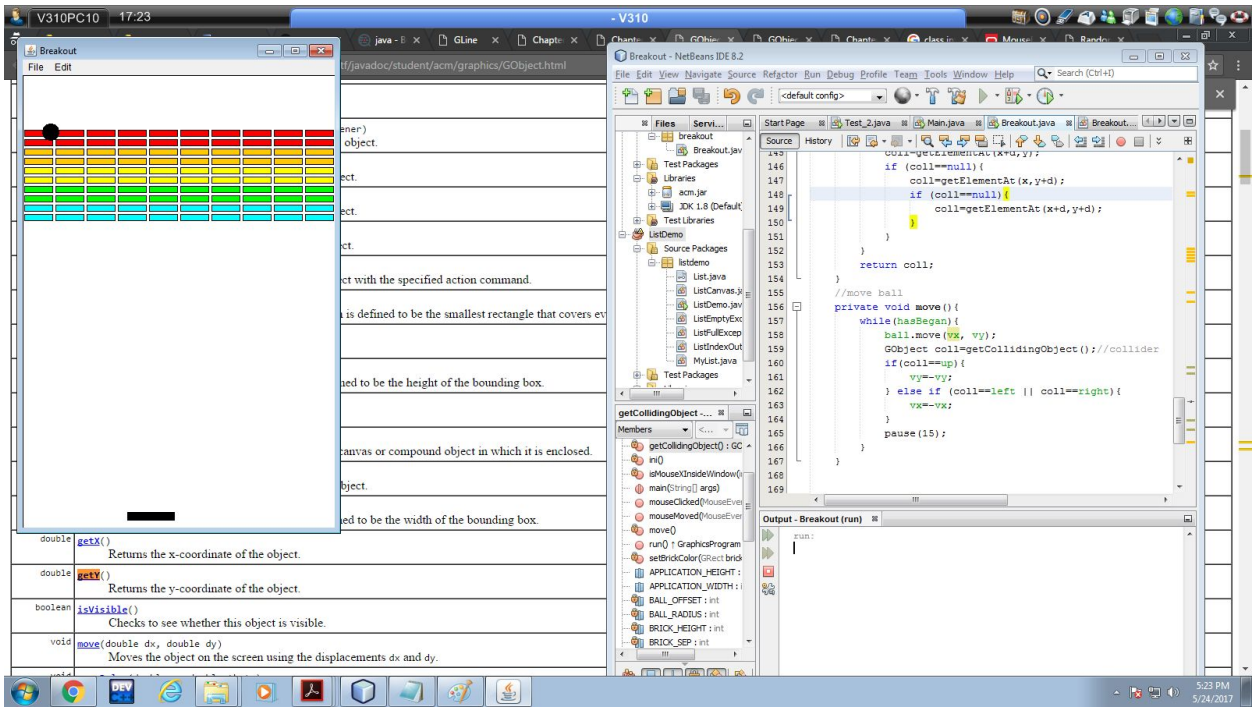
The program colors the bricks according to its row number, after it is drawn into the screen.

b) Creating the paddle



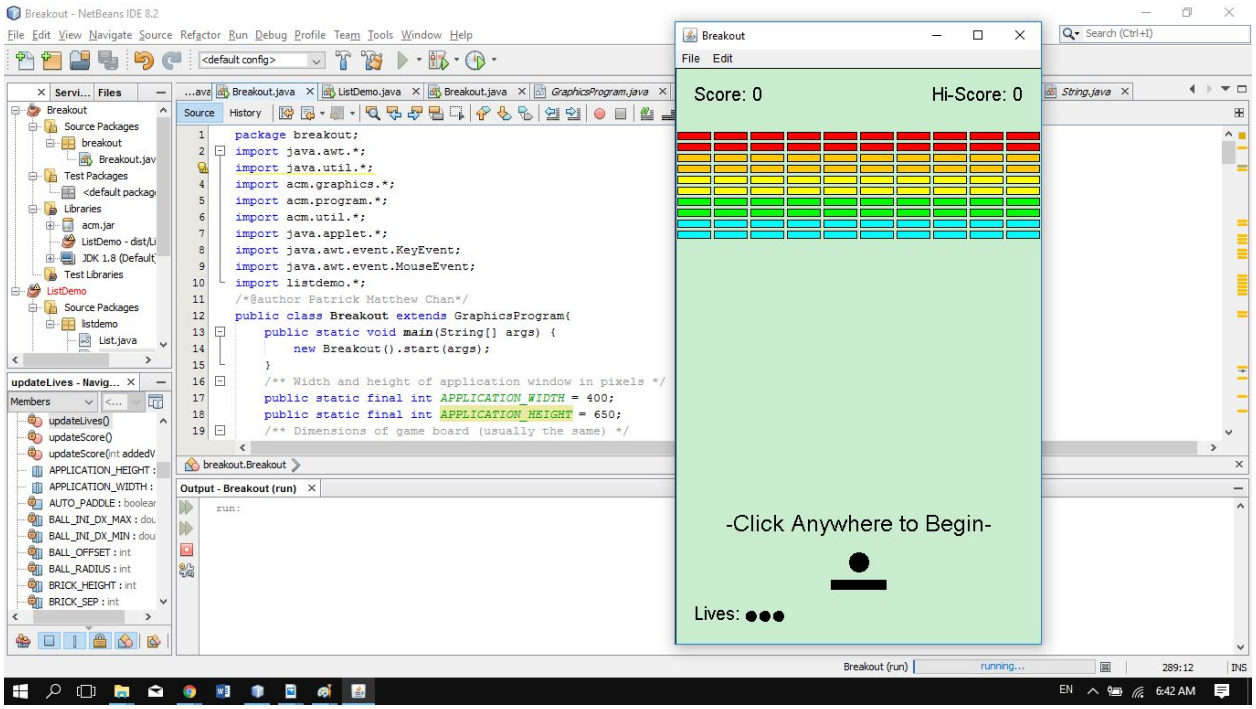
Using addMouseListeners() method, I was able to control the paddle with my overriding mouseMotion() method.

c) Adding the ball, and making it bounce off walls

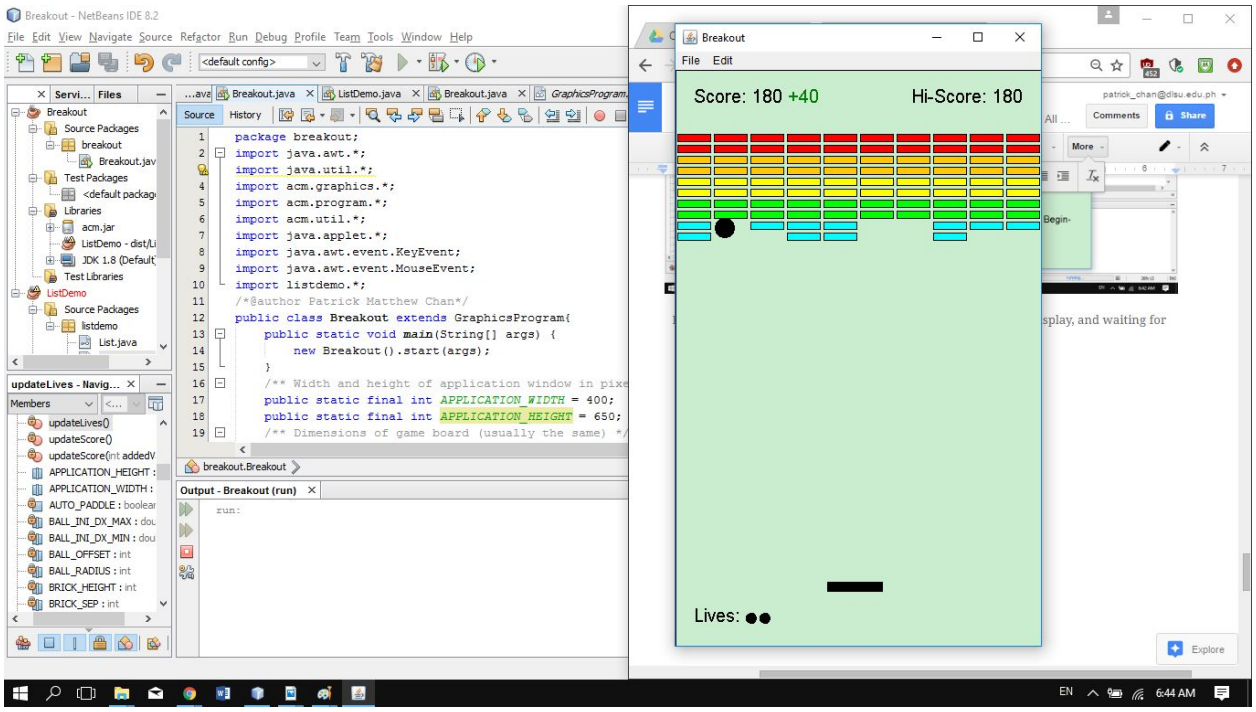


Using getElementAt() method, the program automatically “reflects” the ball whenever it hits a wall.

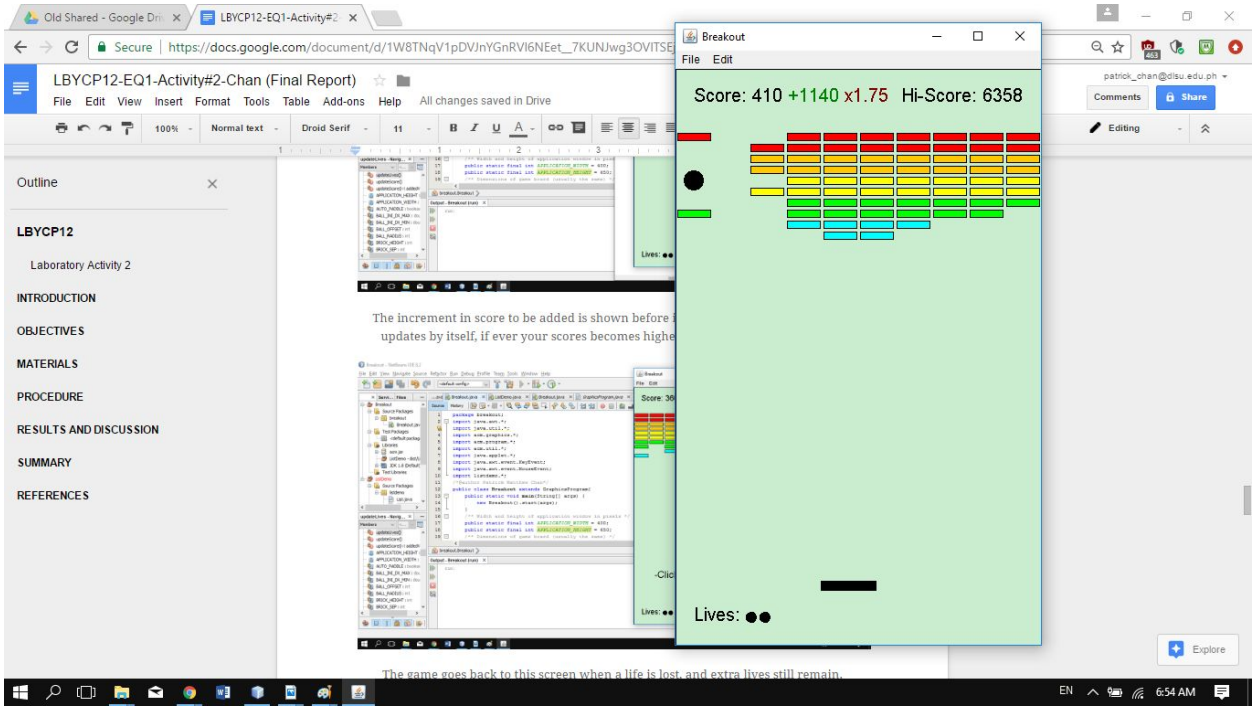
d) The Final Program



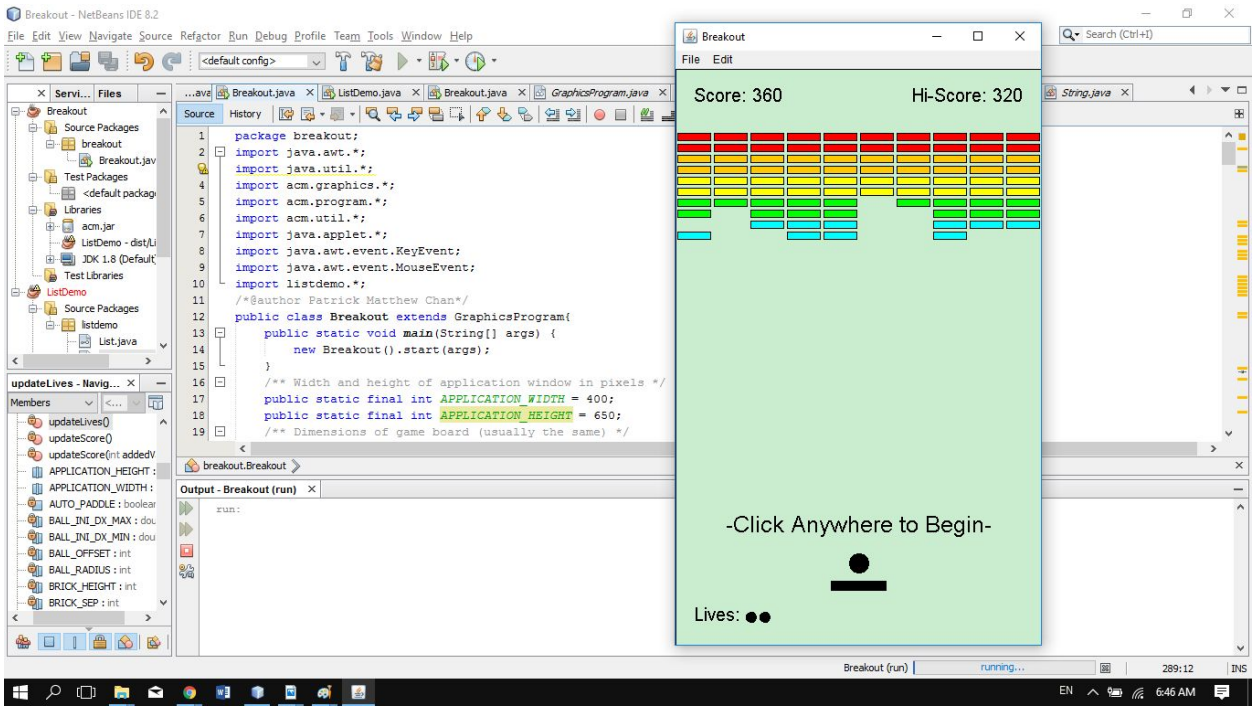
Final Program Features a Lives-system, a score and high score display, and waiting for the user before starting the game.



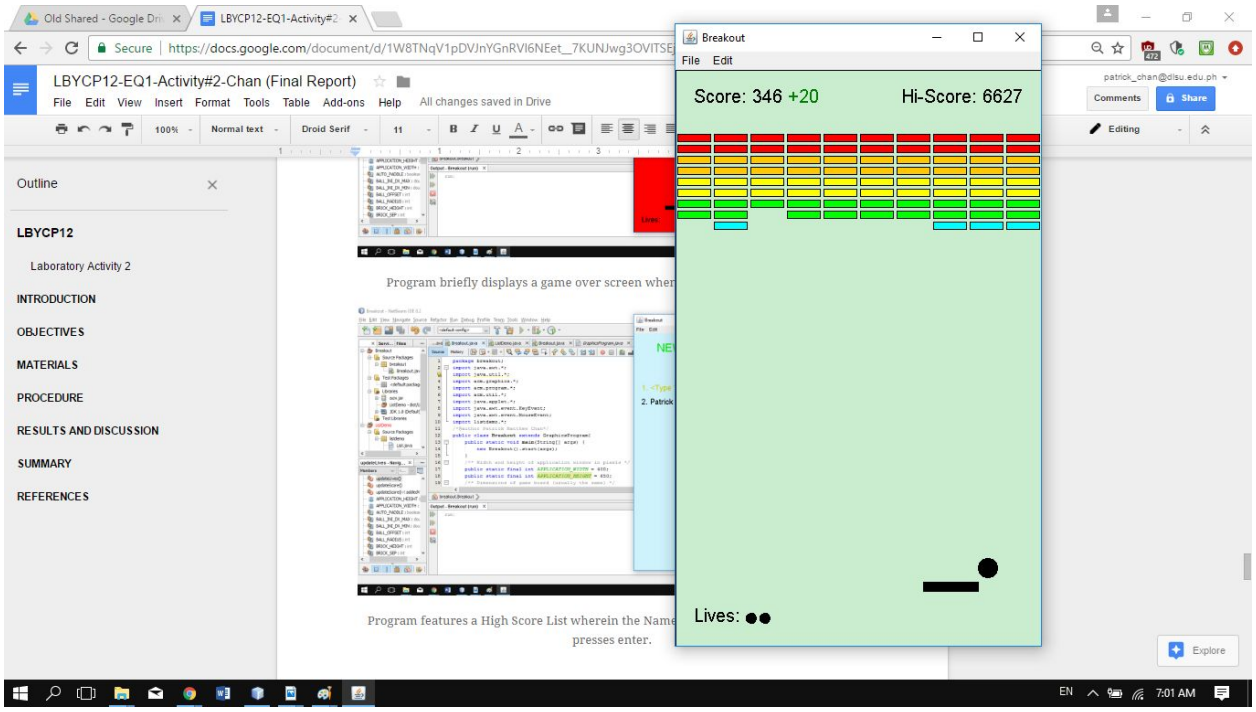
The increment in score to be added is shown before it is added. Also, the High Score updates by itself, if ever your scores becomes higher than the current high score. Moreover, a sound plays whenever a brick is “destroyed”.



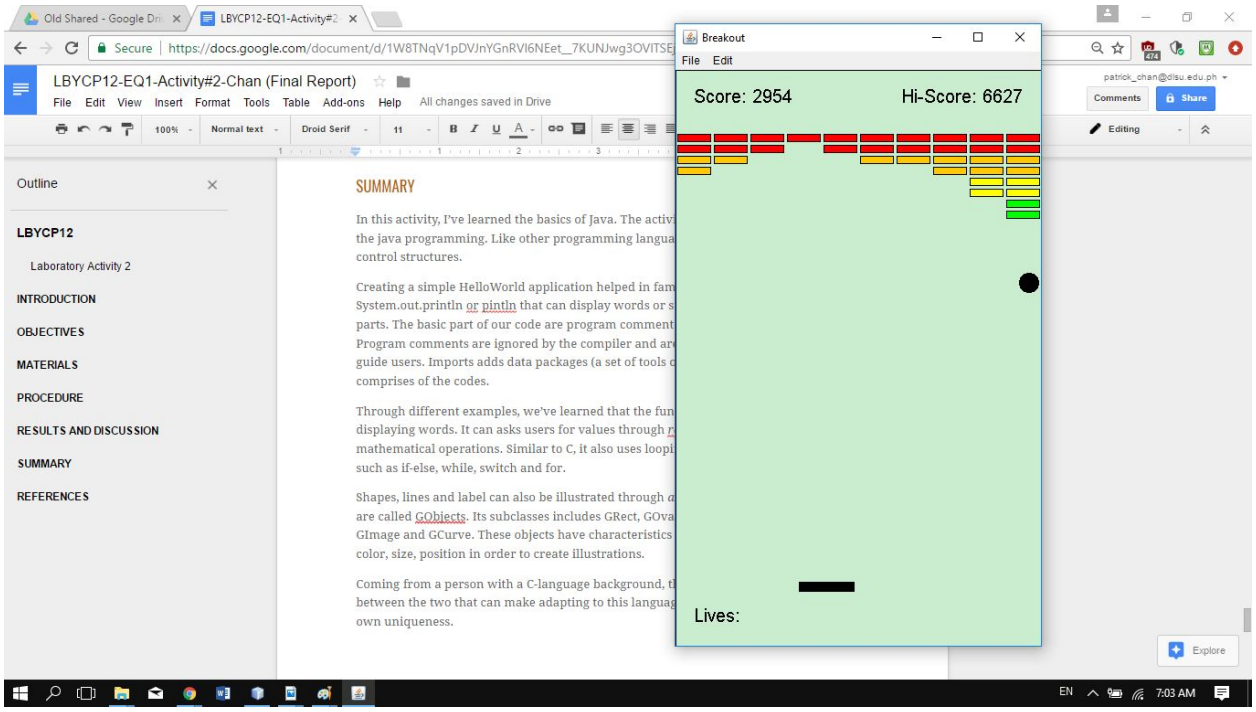
The program also features a multiplier, the increases depending on how much bricks the player is able to hit, before landing on the paddle. This multiplier is applied to the total scores to be added (the increment) while it is active, which increases depending on which rows of bricks are hit.



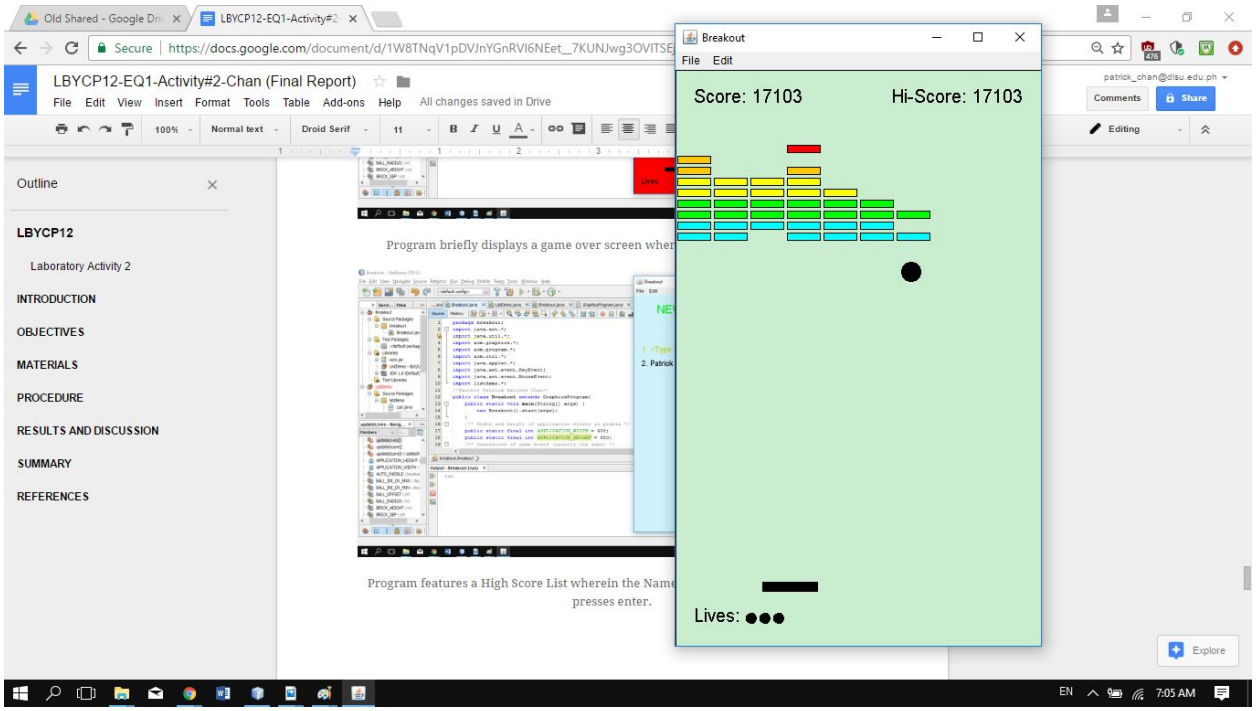
The game goes back to this screen when a life is lost, and extra lives still remain.



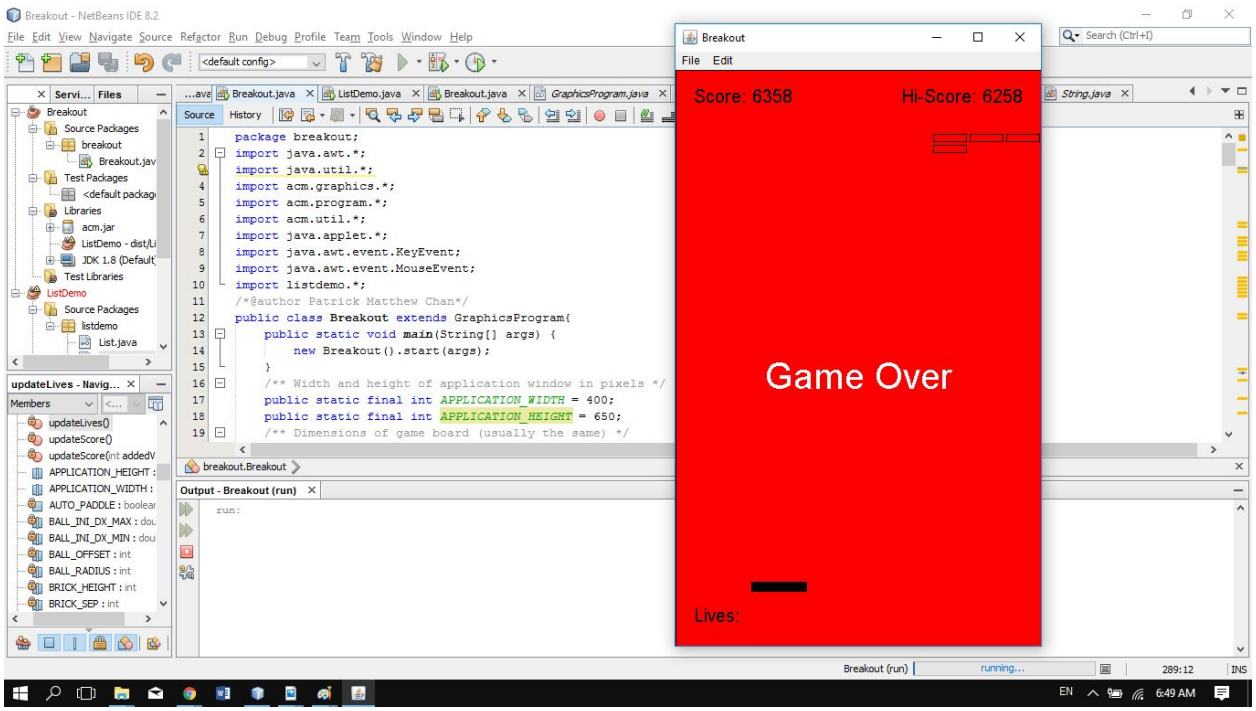
Hitting the Ball by the corner of the paddle reflects it diagonally instead.



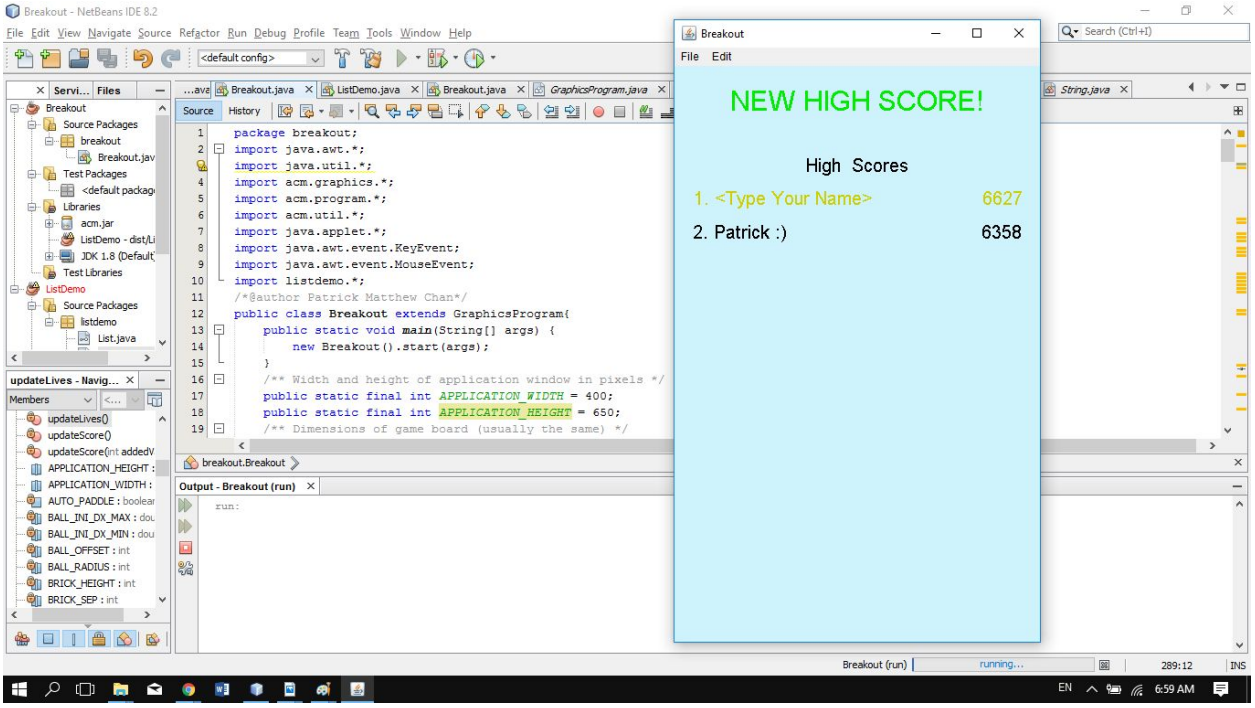
Game goes faster as fewer bricks remain.



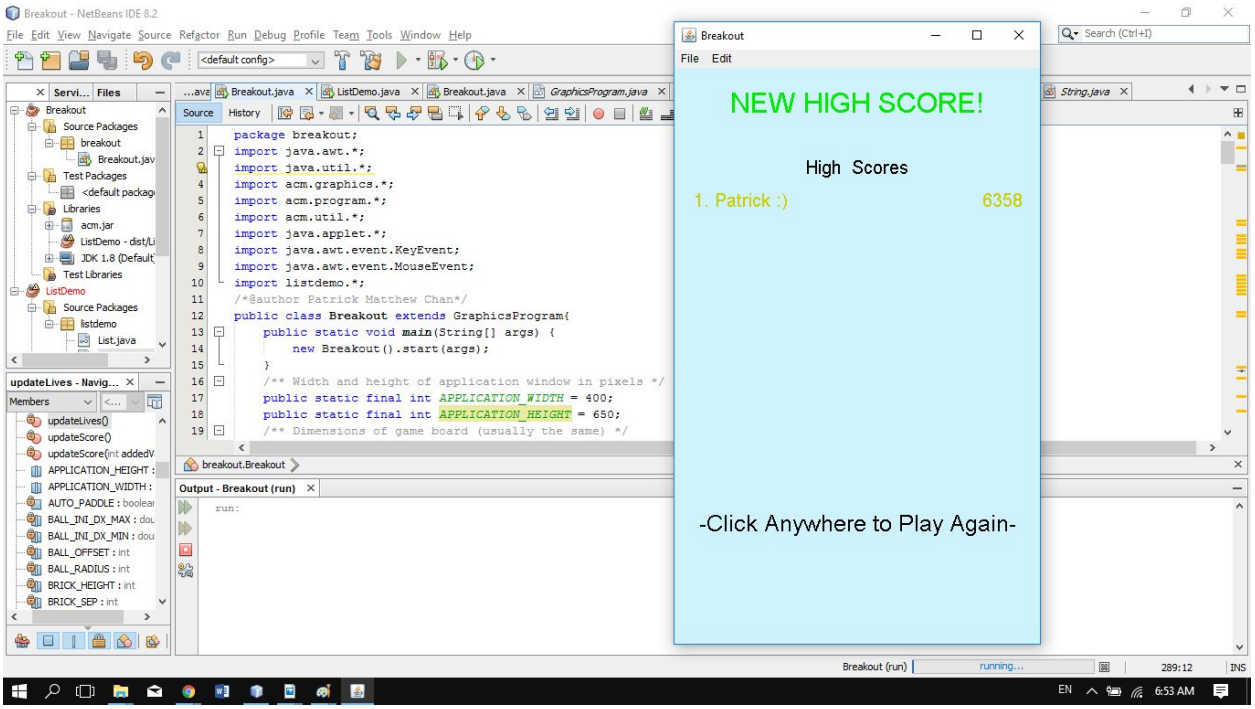
The game also awards bonus lives for every 10,000 points.



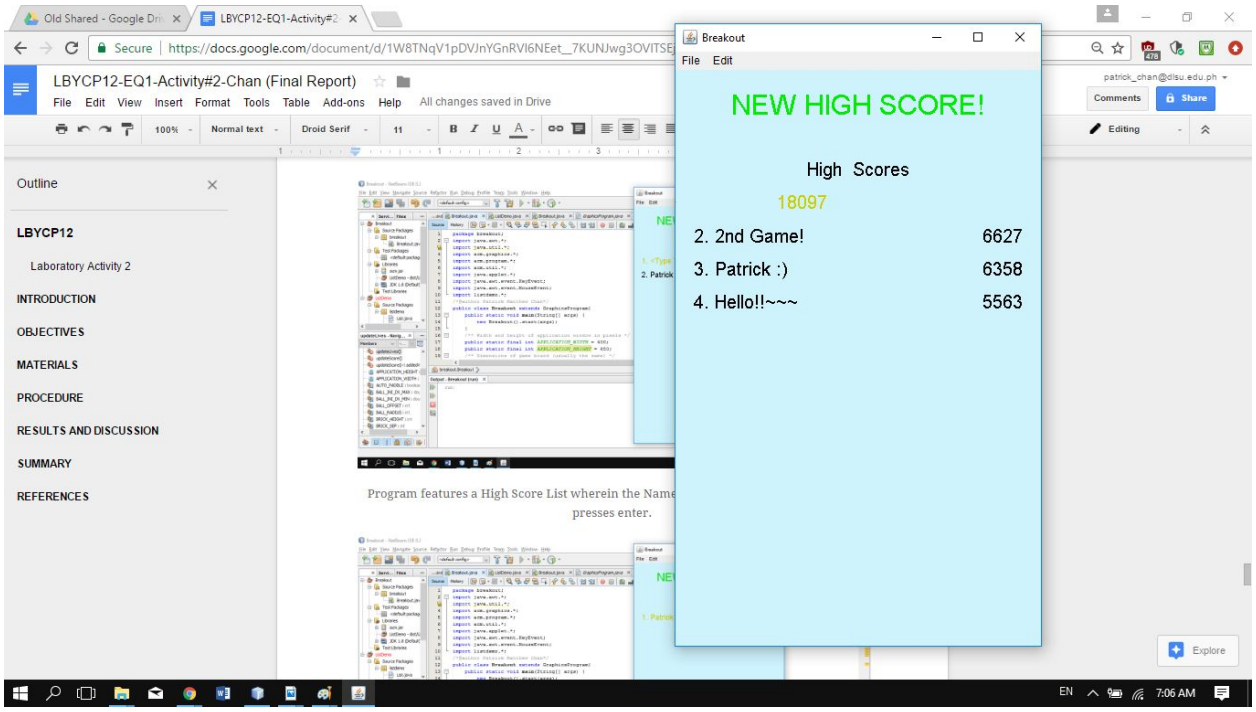
Program briefly displays a game over screen when the player runs out of lives.



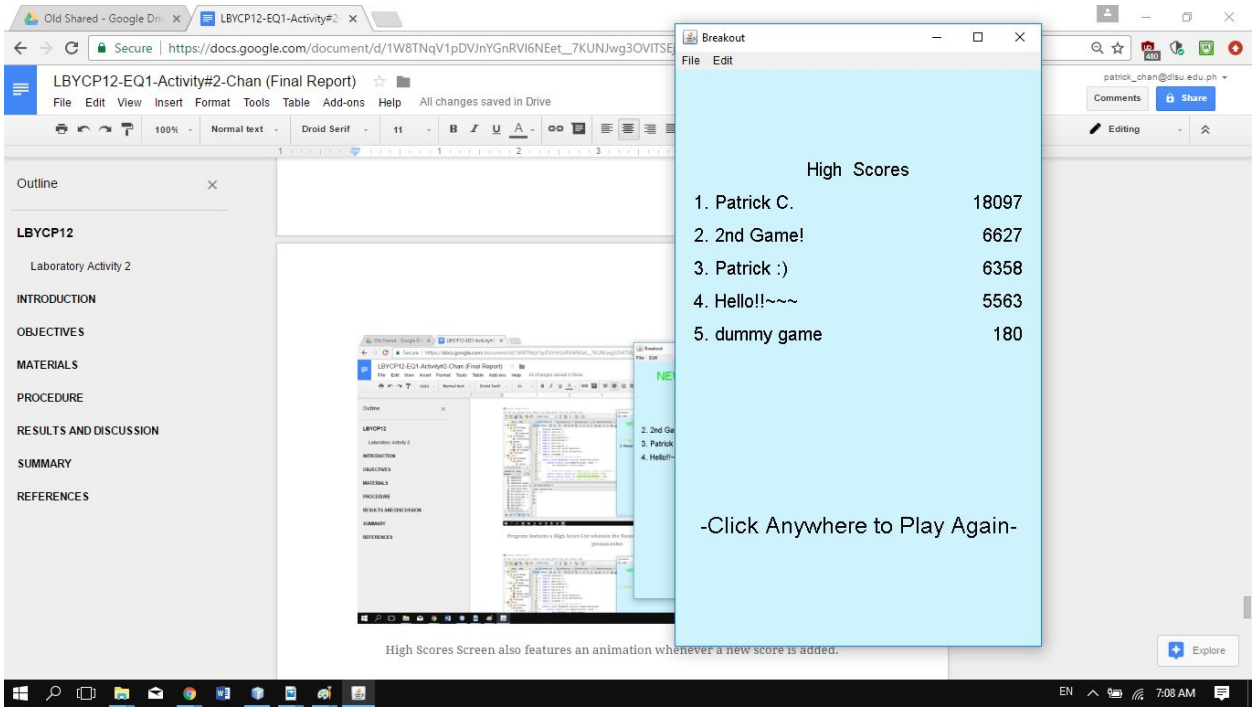
Program features a High Score List wherein the Names can be typed in, until the user presses enter.



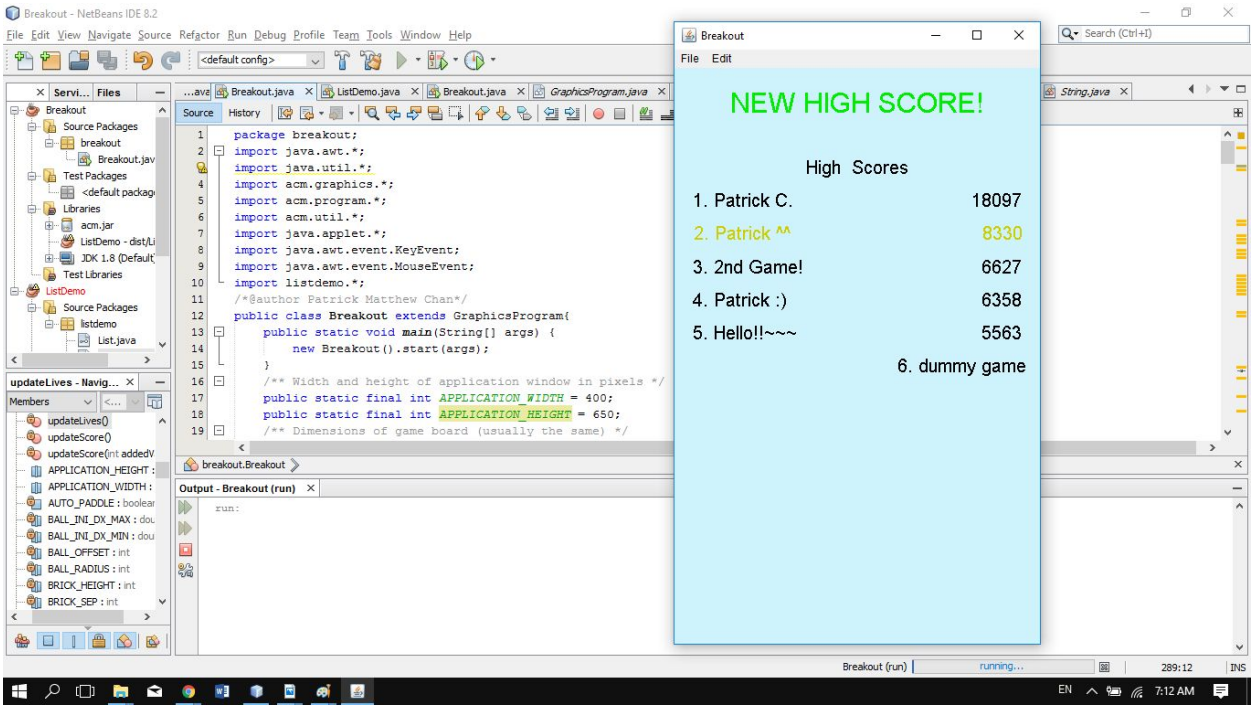
High Scores Screen waits for user to finish viewing high scores.



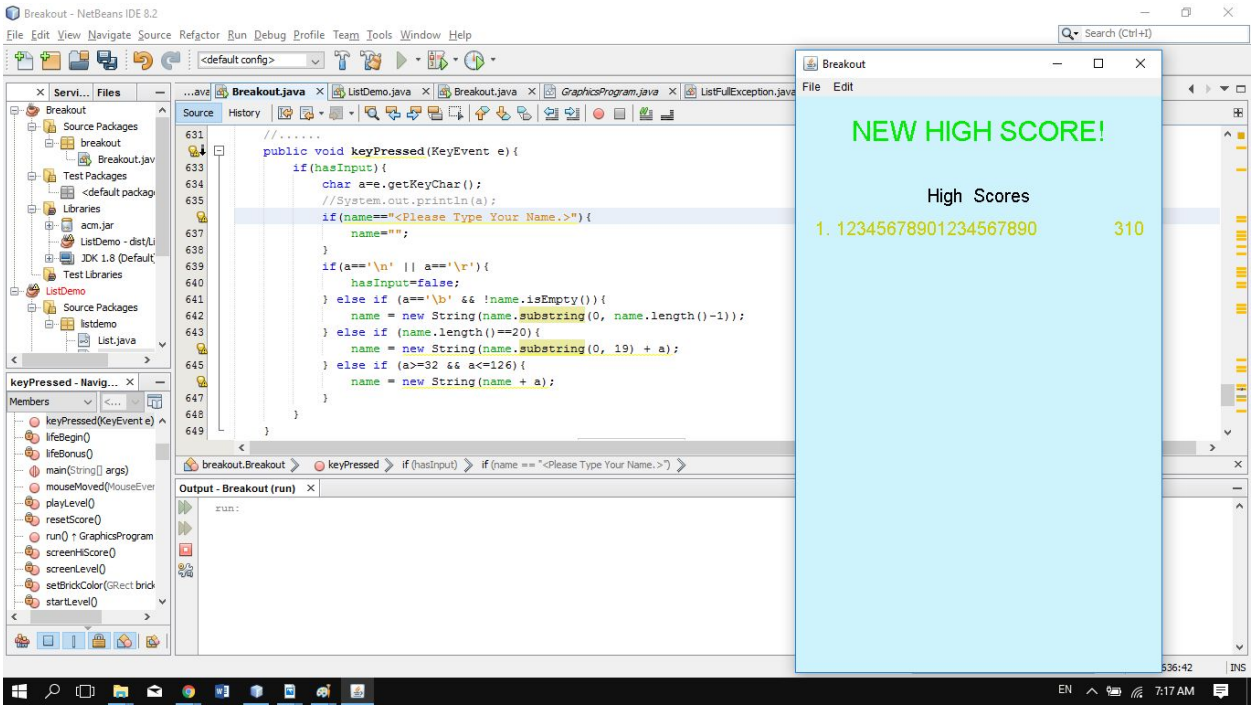
High Scores Screen also features an animation whenever a new score is added.



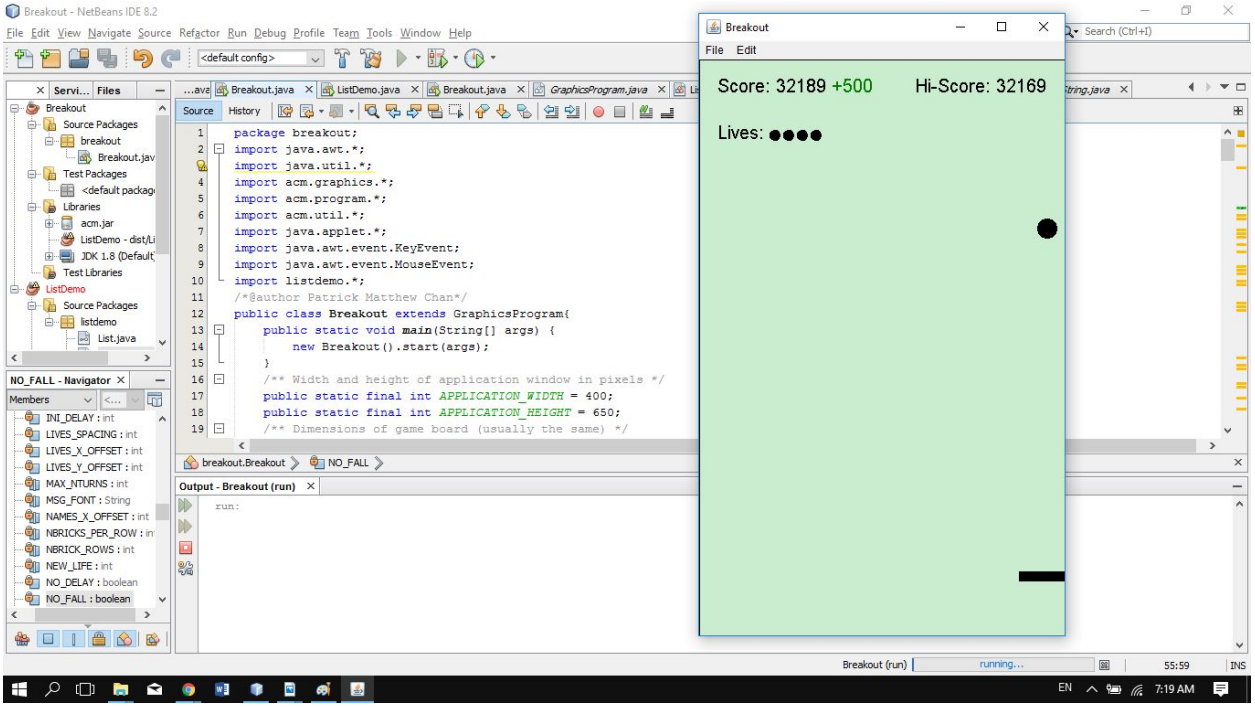
“NEW HIGH SCORE!” not displayed when score is lower than last entry.



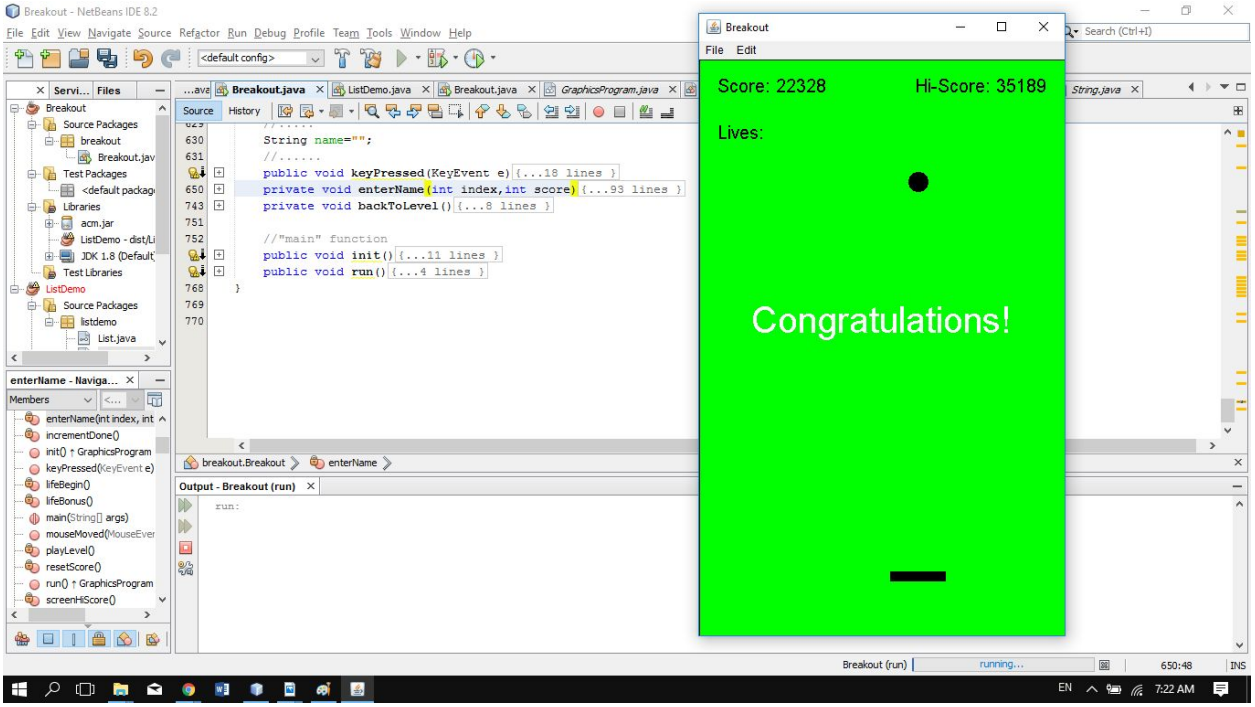
6th entry is automatically removed when a higher score puts it out of place.



Names are limited to 20 characters. Any new input replaces the last character, unless the user presses backspace, or until the user presses enter.

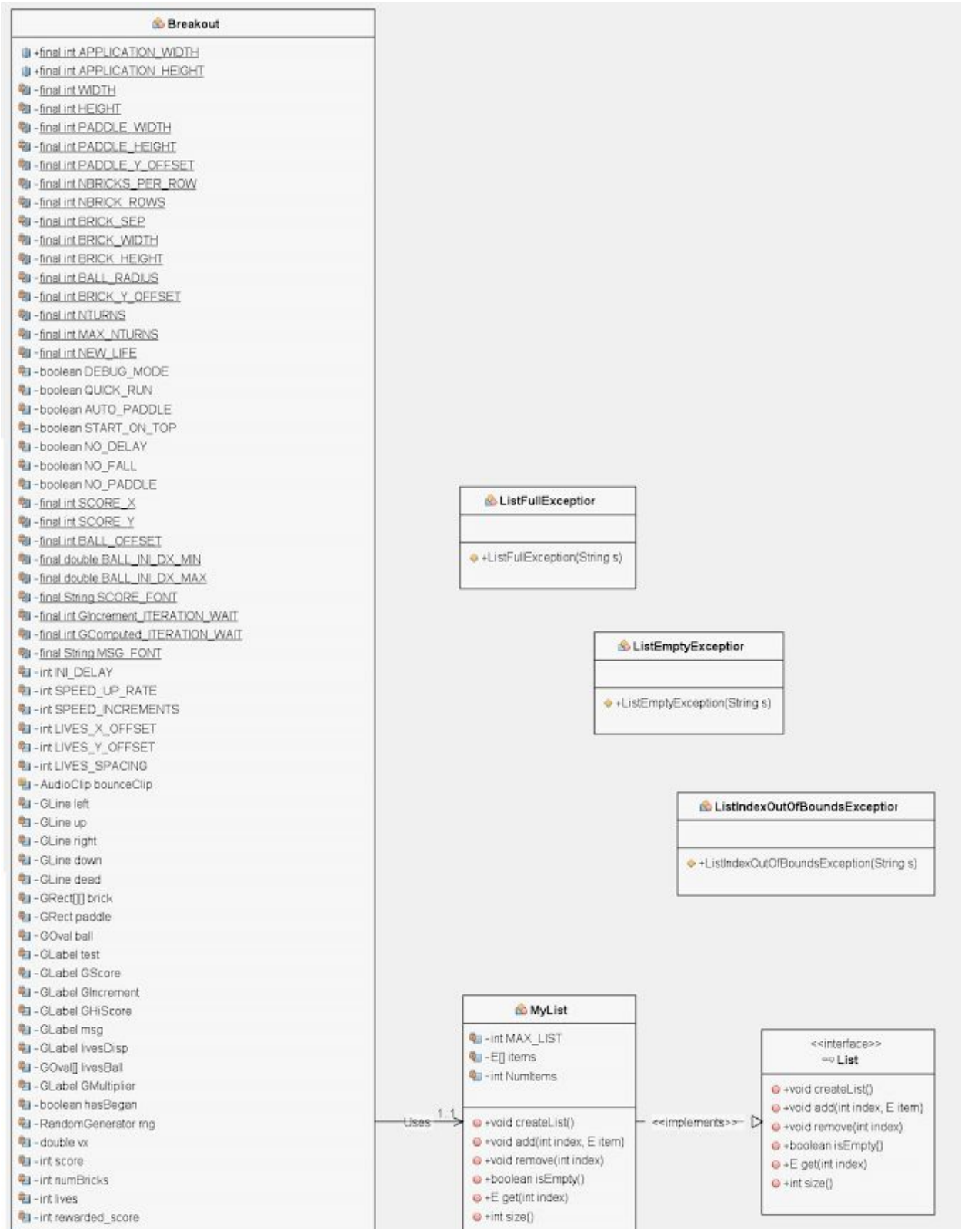


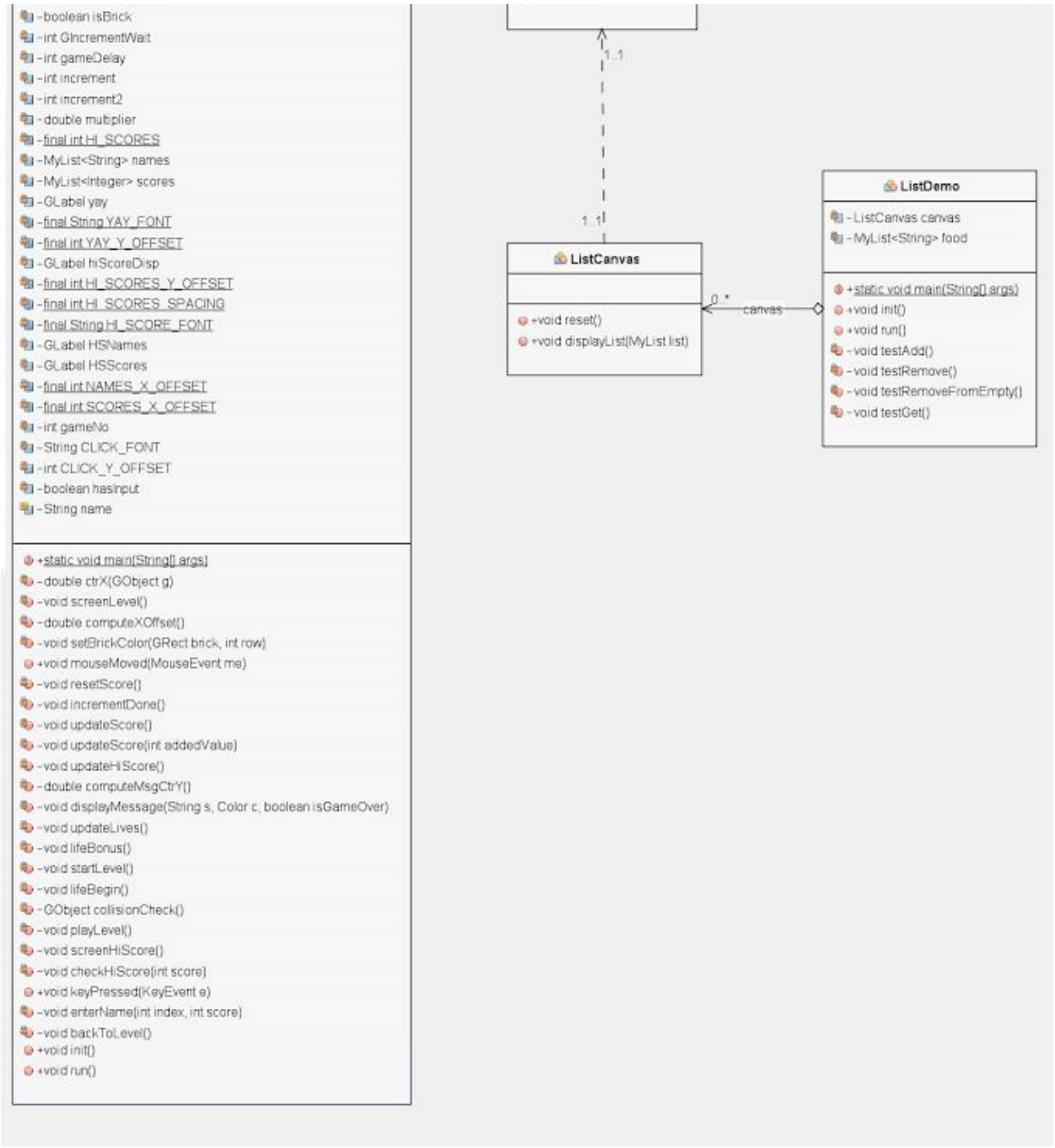
Extra Lives at the end are added towards the score.



“Congratulations!” is shown instead, if the player successfully completes the game.

The UML:


Breakout
+final int APPLICATION_WIDTH
+final int APPLICATION_HEIGHT
-final int WIDTH
-final int HEIGHT
-final int PADDLE_WIDTH
-final int PADDLE_HEIGHT
-final int PADDLE_Y_OFFSET
-final int NBRICKS_PER_ROW
-final int NBRICK_ROWS
-final int BRICK_SEP
-final int BRICK_WIDTH
-final int BRICK_HEIGHT
-final int BALL_RADIUS
-final int BRICK_Y_OFFSET
-final int NTURNS
-final int MAX_NTURNS
-final int NEW_LIFE
-boolean DEBUG_MODE
-boolean QUICK_RUN
-boolean AUTO_PADDLE
-boolean START_ON_TOP
-boolean NO_DELAY
-boolean NO_FALL
-boolean NO_PADDLE
-final int SCORE_X
-final int SCORE_Y
-final int BALL_OFFSET
-final double BALL_INI_DX_MIN
-final double BALL_INI_DX_MAX
-final String SCORE_FONT
-final int GIncrement_ITERATION_WAIT
-final int GComputed_ITERATION_WAIT
-final String MSG_FONT
-int INI_DELAY
-int SPEED_UP_RATE
-int SPEED_INCREMENTS
-int LIVES_X_OFFSET
-int LIVES_Y_OFFSET
-int LIVES_SPACING
-AudioClip bounceClip
-GLine left
-GLine up
-GLine right
-GLine down
-GLine dead
-GRect[][] brick
-GRect paddle
-GOval ball
-GLabel test
-GLabel GScore
-GLabel GIncrement
-GLabel GHiScore
-GLabel msg
-GLabel livesDisp
-GOval[] livesBall
-GLabel GMultiplier
-boolean hasBegan
-RandomGenerator rng
-double vx
-int score
-int numBricks
-int lives
-int rewarded_score
ListFullException
+ListFullException(String s)
ListEmptyException
+ListEmptyException(String s)
ListIndexOutOfBoundsException
+ListIndexOutOfBoundsException(String s)
MyList
-int MAX_LIST
-E[] items
-int NumItems
+void createList()
+void add(int index, E item)
+void remove(int index)
+boolean isEmpty()
+E get(int index)
+int size()
Uses
1..1
<<implements>>
<<interface>>
List
+void createList()
+void add(int index, E item)
+void remove(int index)
+boolean isEmpty()
+E get(int index)
+int size()



SUMMARY

This activity proved to be quite time-consuming to continuously improve, but it was a fun experience, as it was able to introduce me to the field of game development. It was very nice to see how dividing a problem into smaller pieces could actually get me

very far, from simply drawing rows of bricks, to a fully functional game that is quite fun to play.

One thing I found amusing, was how the collision detection was simply done by checking the corners of a bounding rectangle for the ball. Then, it was also nice to know that reflecting the ball, was as simple as negating the velocity of the corresponding axis where the ball hit.

In this activity, I was able to somewhat get the hang of manipulating GObjects in the acm Library in Java. In here, I was able to apply the procedural part of programming that I have learned from C in moving the graphics for the game. Also, it was the first time, that I was able to use an ADT into my program, and apply what I have learned from my DATASAL class in using the given implementation for the List ADT. Overall, it was an interesting and challenging experience that helped me learn more about object-oriented programming.

APPENDIX

A) Breakout.java

```
package ph.edu.dlsu.chan.breakout;
import java.awt.*;
import java.util.*;
import acm.graphics.*;
import acm.program.*;
import acm.util.*;
import java.applet.*;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JFrame;
/*@author Patrick Matthew Chan*/
public class Breakout extends GraphicsProgram{
```

```
   public static void main(String[] args) {
      new Breakout().start(args);
   }
   /** Width and height of application window in pixels */
   public static final int APPLICATION_WIDTH = 400;
   public static final int APPLICATION_HEIGHT = 650;
   /** Dimensions of game board (usually the same) */
   private static final int WIDTH = APPLICATION_WIDTH;
   private static final int HEIGHT = APPLICATION_HEIGHT;
   ////////////////////////////LEVEL SCREEN/////////////////////////////////
   /** Dimensions of the paddle */
   private static final int PADDLE_WIDTH = 60;
   private static final int PADDLE_HEIGHT = 10;
   /** Offset of the paddle up from the bottom */
   private static final int PADDLE_Y_OFFSET = 80;
   /** Number of bricks per row */
   private static final int NBRICKS_PER_ROW = 10;
   /** Number of rows of bricks */
   private static final int NBRICK_ROWS = 10;
   /** Separation between bricks */
   private static final int BRICK_SEP = 4;
   /** Width of a brick */
   private static final int BRICK_WIDTH =
     (WIDTH - (NBRICKS_PER_ROW - 1) * BRICK_SEP) / NBRICKS_PER_ROW;
   /** Height of a brick */
   private static final int BRICK_HEIGHT = 8;
   /** Radius of the ball in pixels */
   private static final int BALL_RADIUS = 10;
   /** Offset of the top brick row from the top */
   private static final int BRICK_Y_OFFSET = 70;
   /** Number of turns/lives */
   private static final int NTURNS = 3;
   private static final int MAX_NTURNS_DRAWN = 10;
   private static final int NEW_LIFE = 10000;
   /** Mine **/
   //cheats
   private final boolean DEBUG_MODE = false;    //(0 mask) cheats enabled
      //should only work if DEBUG_MODE is true
      private final boolean DELAY_ZERO = DEBUG_MODE &&  true;
      private final boolean QUICK_RUN = DEBUG_MODE && true; //(1 mask)
         private final boolean NO_PAUSE = DEBUG_MODE && (QUICK_RUN||false);
         private final boolean AUTO_PADDLE = DEBUG_MODE && (QUICK_RUN||true);
         private final boolean START_ON_TOP = DEBUG_MODE && (QUICK_RUN||true);
         private final boolean DELAY_ONE = DEBUG_MODE && !DELAY_ZERO &&
(QUICK_RUN||true);
      private final boolean NO_FALL = DEBUG_MODE && true;
      private final boolean NO_PADDLE = DEBUG_MODE && true;
```

```
    //global constants
    private static final int SCORE_X = 10;
    private static final int SCORE_Y = 10;
    private static final int BALL_OFFSET = 10;
    private static final double BALL_INI_DX_MIN=1.0;
    private static final double BALL_INI_DX_MAX=3.0;
    private static final String SCORE_FONT = "SansSerif-20";
    private static final int GIncrement_ITERATION_WAIT=60;
    private static final int GComputed_ITERATION_WAIT=20;//must be less than gincrement
iteration
    private static final String MSG_FONT = "SansSerif-40";
    private final int INI_DELAY=7;
    private final int SPEED_UP_RATE=1;
    private final int SPEED_INCREMENTS=6;
    private final int LIVES_X_OFFSET=20;
    private final int LIVES_Y_OFFSET=20;
    private final int LIVES_SPACING=5;
    //global GObjects
    AudioClip bounceClip = MediaTools.loadAudioClip("bounce.au");
    AudioClip bgm = MediaTools.loadAudioClip("bgm.au");
    //this can either be in Breakout\build\classes\ or Brakout project folder
    private GLine left= new GLine(-1,-1,-1,HEIGHT);
    private GLine up= new GLine(-1,-1,WIDTH,-1);
    private GLine right= new GLine(WIDTH,-1,WIDTH,HEIGHT);
    private GLine down= new GLine(-1,HEIGHT,WIDTH,HEIGHT);
    private GLine dead= new
GLine(-1,HEIGHT+3*BALL_RADIUS,WIDTH,HEIGHT+3*BALL_RADIUS);
    private GRect[][] brick = new GRect[NBRICK_ROWS][NBRICKS_PER_ROW];
    private GRect paddle = new GRect(PADDLE_WIDTH, PADDLE_HEIGHT);
    private GOval ball = new GOval(2*BALL_RADIUS,2*BALL_RADIUS);
    private GLabel test;
    private GLabel GScore = new GLabel("Score: 0");
    private GLabel GIncrement = new GLabel(" ");
    private GLabel GHiScore = new GLabel("Hi-Score: 0");
    private GLabel msg;
    private GLabel livesDisp = new GLabel("Lives: ");
    private GOval[] livesBall = new GOval[MAX_NTURNS_DRAWN];
    private GLabel GMultiplier = new GLabel(" ");
    //global variables
    private boolean hasBegan=false;
    private RandomGenerator rng = RandomGenerator.getInstance();
    private double vx=0,vy=0;
    private int score=0;
    private int numBricks=0;
    private int lives=0;
    private int rewarded_score=0;   //avoid double rewarding of lives
    //private boolean nonwallCheck=false;    //true if not wall nor GLabel
```

```
private boolean isBrick=true;
private int GIncrementWait=-1;
private int gameDelay=INI_DELAY;
private int increment=0;
private int increment2 = 0; //(temporary)
private double multiplier=1;
public boolean isMouseInScreen=true;
////////////////////////////PAUSE SCREEN/////////////////////////////////
private GRect overlay=new GRect(WIDTH,HEIGHT);
private final String CLICK1_FONT="SansSerif-30";
private final String CLICK2_FONT="SansSerif-12";
private final static int CLICK12_OFFSET=5;
private boolean isPaused=false;
////////////////////////////HI-SCORE SCREEN/////////////////////////////////
private static final int HI_SCORES=5;
private MyList<String> names=new MyList<String>();
private MyList<Integer> scores=new MyList<Integer>();
private double ctrX(GObject g){
   return (WIDTH-g.getWidth())/2;
}
private GLabel yay = new GLabel("NEW HIGH SCORE!");
private final static String YAY_FONT="SansSerif-30";
private final static int YAY_Y_OFFSET=10;
private GLabel hiScoreDisp = new GLabel("High  Scores");
private final static int HI_SCORES_Y_OFFSET=90;
private final static int HI_SCORES_SPACING=10;
private final static String HI_SCORE_FONT="SansSerif-20";
private GLabel HSNames[] = new GLabel[HI_SCORES+1];//lets just make [0] do nothing
private GLabel HSScores[] = new GLabel[HI_SCORES+1];//lets just make [0] do nothing
private final static int NAMES_X_OFFSET=20;
private final static int SCORES_X_OFFSET=WIDTH-20;
private int gameNo=1;
private final String CLICK_FONT="SansSerif-25";
private final int CLICK_Y_OFFSET=110;
private boolean hasInput=false;




////////////////////////////LEVEL SCREEN/////////////////////////////////
//--------------------------functions---------------------------------//
//display screen
private void screenLevel(){
   setBackground(new Color(202,236,207));
   removeAll();
   startLevel();
}
```

```
   //bricks
   private double computeXOffset() {
      return
0.5*(WIDTH-(NBRICKS_PER_ROW-1)*BRICK_SEP-BRICK_WIDTH*NBRICKS_PER_ROW);
   }
   private void setBrickColor(GRect brick, int row) {
      switch (row%10) {
         case 0:
         case 1:
            brick.setFillColor(Color.RED);
            break;
         case 2:
         case 3:
            brick.setFillColor(Color.ORANGE);
            break;
         case 4:
         case 5:
            brick.setFillColor(Color.YELLOW);
            break;
         case 6:
         case 7:
            brick.setFillColor(Color.GREEN);
            break;
         case 8:
         case 9:
            brick.setFillColor(Color.CYAN);
      }
   }
   //paddle
   public void mouseMoved(MouseEvent me){
      if (hasBegan&&!isPaused){
         float x = me.getX();
         if(x < 0.5 * PADDLE_WIDTH){
            paddle.setLocation(0,paddle.getY());
         } else if (x < (WIDTH - 0.5 * PADDLE_WIDTH)){
            paddle.setLocation(me.getX()-0.5 * PADDLE_WIDTH,paddle.getY());
         } else {
            paddle.setLocation(WIDTH-PADDLE_WIDTH,paddle.getY());
         }
         if(isMouseInScreen && (x>WIDTH+3 || x<-3)){//3 is the allowance
            //System.out.println("mouse exited playing field");
            isMouseInScreen=false;
         } else if (!isMouseInScreen) {
            //System.out.println("mouse entered playing field");
            isMouseInScreen=true;
         }
      }
```

```
}
//pause
public void mouseExited(MouseEvent me){
    if(hasBegan && isMouseInScreen && !isPaused){
        //System.out.println("mouse exited");
        isMouseInScreen=(!isAutoPause)||false;//!isAutoPause bitmask
    }
}
public void mouseEntered(MouseEvent me){
    if(!isMouseInScreen){
        //System.out.println("mouse entered");
        isMouseInScreen=true;
    }
}
public void mouseClicked(MouseEvent me){
    if(isPaused){
        float x = me.getX();
        if(!(x>WIDTH+3 || x<-3)){//3 is the allowance
            //System.out.println("mouse entered playing field");
            if(x < 0.5 * PADDLE_WIDTH){
                paddle.setLocation(0,paddle.getY());
            } else if (x < (WIDTH - 0.5 * PADDLE_WIDTH)){
                paddle.setLocation(me.getX()-0.5 * PADDLE_WIDTH,paddle.getY());
            } else {
                paddle.setLocation(WIDTH-PADDLE_WIDTH,paddle.getY());
            }
            isPaused=false;
            isMouseInScreen=true;
        }
    }
}
//score
private void resetScore(){//to zero
    score=0;
    increment=0;
    multiplier=1.0;
    rewarded_score=0;
}
private void incrementDone(){
    if(GIncrementWait==-1){
        ;
    } else if(GIncrementWait==0){
        score=score+increment2;
        increment2=0;
        remove(GIncrement);
        remove(GScore);
        GScore=new GLabel("Score: " + score);
```

```
        GScore.setFont(SCORE_FONT);
        add(GScore,SCORE_X,SCORE_Y+GScore.getHeight());

        while(score>=rewarded_score+NEW_LIFE && hasBegan){
           lives++;
           rewarded_score+=NEW_LIFE;
           updateLives();
        }

        GIncrementWait--;
     } else if(GIncrementWait==GComputed_ITERATION_WAIT){
        increment=(int)(double)(increment*multiplier);
        increment2+=increment;
        increment=0;
        multiplier=1;
        remove(GMultiplier);
        remove(GIncrement);
        GIncrement=new GLabel(" +" + increment2);
        GIncrement.setFont(SCORE_FONT);
        GIncrement.setColor(new Color(0,128,0));
        add(GIncrement,SCORE_X+GScore.getWidth(),SCORE_Y+GIncrement.getHeight());
        GIncrementWait--;
     } else if (GIncrementWait==GIncrement_ITERATION_WAIT-1){
        score=score+increment2;
        increment2=0;
     } else {
        GIncrementWait--;
     }
  }
  private void updateScore(){
     remove(GScore);
     remove(GIncrement);
     remove(GMultiplier);
     GScore=new GLabel("Score: " + score);
     GScore.setFont(SCORE_FONT);
     add(GScore,SCORE_X,SCORE_Y+GScore.getHeight());
  }
  private void updateScore(int addedValue){
     remove(GScore);
     remove(GIncrement);
     remove(GMultiplier);
     if(increment>0){
        multiplier+=0.05;
     }
     increment+=addedValue;
     GScore=new GLabel("Score: " + score);
     GScore.setFont(SCORE_FONT);
```

```
      add(GScore,SCORE_X,SCORE_Y+GScore.getHeight());
      if(addedValue>0){
         GIncrement=new GLabel(" +" + increment);
         GIncrement.setFont(SCORE_FONT);
         GIncrement.setColor(new Color(0,128,0));
         add(GIncrement,SCORE_X+GScore.getWidth(),SCORE_Y+GIncrement.getHeight());
         GIncrementWait=GIncrement_ITERATION_WAIT;
         if(multiplier>1){
            GMultiplier=new GLabel(" x" + (float)multiplier);
            GMultiplier.setFont(SCORE_FONT);
            GMultiplier.setColor(new Color(128,0,0));
            add(GMultiplier,GIncrement.getX()+GIncrement.getWidth(),GIncrement.getY());
         }
      }
   }
   //hi-score
   private void updateHiScore(){
      remove(GHiScore);
      int hiscore=score;
      if(!(scores.isEmpty()) && scores.get(1)>score){
         GHiScore=new GLabel("Hi-Score: " + scores.get(1));
      } else {
         GHiScore=new GLabel("Hi-Score: " + score);
      }
      GHiScore.setFont(SCORE_FONT);

add(GHiScore,WIDTH-GHiScore.getWidth()-SCORE_X,SCORE_Y+GHiScore.getHeight());
   }
   //message
   private double computeMsgCtrY(){
      double TOP_SPACE=(BRICK_Y_OFFSET+(BRICK_HEIGHT*NBRICK_ROWS+
            BRICK_SEP*NBRICK_ROWS));
      double
WHITE_SPACE=HEIGHT-(TOP_SPACE+(PADDLE_HEIGHT+PADDLE_Y_OFFSET));
      return TOP_SPACE+0.5*WHITE_SPACE;
   }
   private void displayMessage(String s,Color c,boolean isGameOver){
      double msg_Y=0;
      msg=new GLabel(s);
      msg.setFont(MSG_FONT);
      msg.setColor(c);
      if(isGameOver){
         msg_Y=computeMsgCtrY()-0.5*msg.getHeight();
      } else{
         msg_Y=(HEIGHT-msg.getHeight())/2;
      }
      add(msg,(WIDTH-msg.getWidth())/2,msg_Y);
```

```
}
//lives
private void updateLives(){
    double curX=0;
    double curY=0;
    for(int i=0;i<MAX_NTURNS_DRAWN;i++){
        remove(livesBall[i]);
    }
    if(lives>MAX_NTURNS_DRAWN){
        livesDisp.setLabel("("+lives+")");
        curX=livesDisp.getX()+livesDisp.getWidth();
        curY=livesDisp.getY()-BALL_RADIUS;
        for(int i=0;i<MAX_NTURNS_DRAWN;i++){
            add(livesBall[i],curX,curY);
            curX=curX+BALL_RADIUS+LIVES_SPACING;
        }
    } else {
        if(!livesDisp.getLabel().equals("Lives: ")){
            livesDisp.setLabel("Lives: ");
        }
        curX=livesDisp.getX()+livesDisp.getWidth();
        curY=livesDisp.getY()-BALL_RADIUS;
        for(int i=0;i<lives;i++){
            add(livesBall[i],curX,curY);
            curX=curX+BALL_RADIUS+LIVES_SPACING;
        }
    }
    //remove
}
private void lifeBonus(){
    if(lives>0){
        double dist=(livesDisp.getY()-GScore.getY())-2*(GScore.getHeight());
        for(int j=0;j<=dist;j++){
            livesDisp.move(0,-1);
            if(lives>MAX_NTURNS_DRAWN){
                for(int i=0;i<MAX_NTURNS_DRAWN;i++){
                    livesBall[i].move(0,-1);
                }
                GHiScore.move(0,1);
            } else {
            for(int i=0;i<lives;i++){
                livesBall[i].move(0,-1);
            }
            }
            pause(1);
        }
        while(lives>0){
```

```
            if(lives==MAX_NTURNS_DRAWN){
               updateLives();
            }
            if(lives<=MAX_NTURNS_DRAWN){
            remove(livesBall[lives-1]);
            lives--;
               updateScore(700);
            pause(700);
            } else {
               lives--;
               updateScore(700);
               livesDisp.setLabel("("+lives+")");
               pause(1);
         }
         }
         GIncrementWait=GComputed_ITERATION_WAIT;
         incrementDone();
         pause(400);
         GIncrementWait=0;
         incrementDone();
         pause(500);
      }
   }
   //--------------------------initializer---------------------------------//
   private void startLevel(){
      hasBegan=false;
      removeAll();
      //wall
      add(up);
      add(down);
      add(left);
      add(right);
      dead.setVisible(false);
      add(dead);
      //bricks
      numBricks=0;
      for (int i = 0; i < NBRICK_ROWS; i++){//row
         for (int j = 0; j < NBRICKS_PER_ROW; j++){//col
            double x, y; // brick location
            brick[i][j] = new GRect(BRICK_WIDTH, BRICK_HEIGHT);
            x = computeXOffset() + j * (BRICK_WIDTH + BRICK_SEP);
            y = BRICK_Y_OFFSET + i * (BRICK_HEIGHT + BRICK_SEP);
            brick[i][j].setFilled(true);
            setBrickColor(brick[i][j],i);
            if(y+2*BRICK_HEIGHT<dead.getY() && x+BRICK_WIDTH<right.getX()){
            add(brick[i][j], x, y);
            numBricks++;
```

```
            //pause(20);
         }
            if(!DELAY_ONE && !DELAY_ZERO){
               pause(10);
      }
         }
      }
      //score
      GIncrementWait=-1;
      updateScore();
      increment=0;
      increment2=0;
      multiplier=1;
      rewarded_score=0;
      //hi-score
      updateHiScore();
      //lives
      lives=NTURNS;
      livesDisp.setFont(SCORE_FONT);
      add(livesDisp,LIVES_X_OFFSET,HEIGHT-LIVES_Y_OFFSET-livesDisp.getHeight());
      updateLives();
      //paddle
      paddle.setFilled(true);
      add(paddle, 0.5*(WIDTH - PADDLE_WIDTH), HEIGHT - PADDLE_Y_OFFSET -
PADDLE_HEIGHT);
      //ball
      ball.setFilled(true);
      add(ball, 0.5*WIDTH - BALL_RADIUS , paddle.getY() - PADDLE_HEIGHT -
BALL_RADIUS - BALL_OFFSET );
      if(START_ON_TOP){
         ball.setLocation(0.5*WIDTH - BALL_RADIUS,BRICK_Y_OFFSET/2);
      }
      //game delay
      gameDelay=INI_DELAY;
      //use a life
      bgm.loop();
      lifeBegin();
   }
   private void lifeBegin(){
      hasBegan=false;
      if(lives>0){
         //paddle
         if(NO_PADDLE){//cheats
            remove(paddle);
         }
         paddle.setLocation(0.5*(WIDTH - PADDLE_WIDTH), HEIGHT - PADDLE_Y_OFFSET
- PADDLE_HEIGHT);
```

```
        //ball
        ball.setLocation(0.5*WIDTH - BALL_RADIUS , paddle.getY() - PADDLE_HEIGHT -
BALL_RADIUS - BALL_OFFSET);
        if(START_ON_TOP){
           ball.setLocation(0.5*WIDTH - BALL_RADIUS,BRICK_Y_OFFSET/2);
        }
        //ini speed
        vx = rng.nextDouble(BALL_INI_DX_MIN, BALL_INI_DX_MAX);
        if (rng.nextBoolean(0.5)){
           vx = -vx;
        }
        vy = -3.0;
        //waitforclick
        GLabel click=new GLabel("-Click Anywhere to Begin-");
        click.setFont(CLICK_FONT);
        add(click,ctrX(click),HEIGHT-click.getHeight()-CLICK_Y_OFFSET);
        waitForClick();//////////////////////////remove this if program fails
        //use life
        lives--;
        updateLives();
        //done
        remove(click);
        hasBegan=true;
        playLevel();
     }else {
        setBackground(Color.red);
        displayMessage("Game Over",Color.WHITE,true);
        pause(400);
        screenHiScore();
     }
  }
  //-----------------------playing the level-----------------------------//
  private GObject collisionCheck(){
     //nonwallCheck=false;
     double x=ball.getX();
     double y=ball.getY();
     int d=2*BALL_RADIUS;//ball diameter
     int corner=1;//ball corner [1 2;3 4]
     //checker
     double bx=x,by=y;   //colliding boundary
     GObject coll=getElementAt(bx,by);//colliding object
     if(coll==null){
        bx=x+d;
        corner=2;
        coll=getElementAt(bx,by);
        if (coll==null){
           by=y+d;
```

```
            corner=4;
            coll=getElementAt(bx,by);
            if (coll==null){
               bx=x;
               corner=3;
               coll=getElementAt(bx,by);
            }
         }
      }
      if(coll instanceof GLabel || coll instanceof GOval){
         double ax=coll.getX();
         double ay=coll.getY();
         remove(coll);
         GObject temp=collisionCheck();
         add(coll,ax, ay);
         return temp;
      }
      //the "reaction"
      //walls (also to prevent stuck)
      if((x<=0 && vx<0)||(x+d>=WIDTH && vx>0)){//vertical walls
         vx=-vx;
      }
      if(y<=0 && vy<0){
         vy=-vy;
      }
      if(NO_FALL && (y+d>=HEIGHT + 2*BALL_RADIUS && vy>0)){//cheat
         vy=-vy;
      }
      //isBrick check
      isBrick=true;
      try{
         GRect brick = (GRect)coll;
      } catch(ClassCastException e){
         isBrick=false;
      } finally {
         if(coll==null || coll==paddle){
            isBrick=false;
         }
      }
      //based on coll
      if(isBrick){  //brick only
         double cx=coll.getX();
         double cy=coll.getY();
         double c2x=cx+coll.getWidth();
         double c2y=cy+coll.getHeight();
         //nonwallCheck=true; //bounceClip.play();
         if((bx>=cx && bx<=cx+BALL_INI_DX_MAX) || (bx<=c2x &&
```

```
bx>=c2x-BALL_INI_DX_MAX)){
            //if(vx>0){
            vx=-vx;
            //}
         }
         if((by>=cy && by<=cy+3.0) || (by<=c2y && by>=c2y-3.0)){
            vy=-vy;
         }
         /*{
         test=new GLabel("x="+x+" y="+y+" bx="+bx+" by="+by+" cx="+cx+" cy="+cy+"
c2x="+c2x+" c2y="+c2y+"."+coll);
         add(test,50,50);
         waitForClick();
         remove(test);
         }*/
      } else if(coll==paddle){
         if(GIncrementWait!=-1){
            GIncrementWait=GComputed_ITERATION_WAIT;
         }
         double cx=coll.getX();
         double cy=coll.getY();
         double c2x=cx+coll.getWidth();
         double c2y=cy+coll.getHeight();
         if(corner==3 || corner==4){
            if((by>=cy && by<=cy+3.0) && ((bx>=cx && bx<=cx+BALL_INI_DX_MAX) ||
(bx<=c2x && bx>=c2x-BALL_INI_DX_MAX))){
               vx=-vx;
            }
            if(paddle.contains(bx, by)){
               if(vy>0){
                  vy=-vy;
               }
               ball.setLocation(x,paddle.getY()-d);
            }
         } else if (paddle.contains(bx,by)){
            vy=-vy;
         }
         /*
         double xx=x+BALL_RADIUS;
         double yy=y+BALL_RADIUS;
         //double cyy=coll.getY()+0.5*PADDLE_HEIGHT;
         if(paddle.contains(xx,yy)){
            ball.move(0, -1.5*d);
         }*/
      }
      return coll;
   }
```

```
private void playLevel(){
   while(hasBegan && numBricks>0){

      if(!isMouseInScreen && !NO_PAUSE){
         screenPause();
      }

      ball.move(vx, vy);
      GObject coll=collisionCheck();//collider

      //reaction...++
      if(coll==dead){
         if(GIncrementWait!=-1){
            GIncrementWait=GComputed_ITERATION_WAIT;
            incrementDone();
            pause(400);
            GIncrementWait=0;
            incrementDone();
            pause(200);
         }
         hasBegan=false;
         lifeBegin();
      } else if (isBrick){
         GRect brick = (GRect)coll;
         if(brick.getFillColor()==Color.RED){
            updateScore(100);
         } else if(brick.getFillColor()==Color.ORANGE){
            updateScore(80);
         } else if(brick.getFillColor()==Color.YELLOW){
            updateScore(60);
         } else if(brick.getFillColor()==Color.GREEN){
            updateScore(40);
         } else if(brick.getFillColor()==Color.CYAN){
            //setBackground(Color.BLUE);
            updateScore(20);
         }
            bounceClip.play();
         remove(coll);
         numBricks--;
         /*{test=new GLabel("SCORE: " + score);
         add(test,50,50);
         waitForClick();
         remove(test);}*/
      }
      //delay
      if(AUTO_PADDLE){
         paddle.setLocation(ball.getX()-PADDLE_WIDTH/2+BALL_RADIUS,
```

```
paddle.getY());//hax/cheat
          }
          if(DELAY_ONE){
             pause(1);
          } else if (!DELAY_ZERO){
             gameDelay=INI_DELAY-((int)((1-((float)numBricks/(NBRICKS_PER_ROW
                   *NBRICK_ROWS)))*(SPEED_INCREMENTS-1))*(SPEED_UP_RATE));;
             if(gameDelay<1){
                gameDelay=1;
             }
             pause (gameDelay);
          }
          incrementDone();
          updateHiScore();
       }
       if(numBricks==0){
          if(GIncrementWait!=-1){
             GIncrementWait=GComputed_ITERATION_WAIT;
             incrementDone();
             pause(400);
             GIncrementWait=0;
             incrementDone();
             pause(200);
          }
          hasBegan=false;
          displayMessage("Clear Bonus: +5000",Color.BLUE,false);
          updateScore(5000);
          //increment score for a while...
          GIncrementWait=GComputed_ITERATION_WAIT;
          incrementDone();
          pause(700);
          GIncrementWait=0;
          incrementDone();
          pause(500);
          remove(msg);
          //but dont include life bonus in multiplier
          lifeBonus();
          setBackground(Color.green);
          displayMessage("Congratulations!",Color.WHITE,false);
          pause(700);
          screenHiScore();
       }
    }


    ////////////////////////////PAUSE SCREEN/////////////////////////////////
    private void screenPause(){
```

```
    add(overlay);
    bgm.stop();
    GLabel click1=new GLabel("- PAUSED -");
    GLabel click2=new GLabel("Click Anywhere to Resume");
    click1.setFont(CLICK1_FONT);
    click1.setColor(Color.WHITE);
    click2.setFont(CLICK2_FONT);
    click2.setColor(Color.WHITE);
    add(click1,ctrX(click1),(HEIGHT-click2.getHeight()-CLICK12_OFFSET
          +click1.getHeight())/2);
    add(click2,ctrX(click2),(HEIGHT+click1.getHeight()+CLICK12_OFFSET
          +click2.getHeight())/2);
    isPaused=true;
    while(isPaused){//waitforclick
       pause(0);
    }
    //waitForClick();
    isPaused=false;
    isMouseInScreen=true;
    remove(overlay);
    remove(click1);
    remove(click2);
    bgm.loop();
}



////////////////////////////HI-SCORE SCREEN////////////////////////////////
private void screenHiScore() throws HiScoreListSyncException{
    bgm.stop();
    //HS File Read (creation if none)
    names.createList();
    scores.createList();
    String filePath=new File("").getAbsolutePath();
    FileReader in;
    BufferedReader br = null;
    try{
       in=new FileReader(filePath+"/Scores.dat");
       br=new BufferedReader(in);
    } catch(IOException e){
       File a=new File(filePath+"/Scores.dat");
       try{
          a.createNewFile();
          in=new FileReader(filePath+"/Scores.dat");
          br=new BufferedReader(in);
       } catch (IOException ee){
```

```
            System.err.println("FILE READ ERROR\n");
            ee.printStackTrace();
         }
      }
      MyList<String> buff=new MyList<>();
      buff.createList();
      String a="";
      try{
         a=br.readLine();
      } catch (IOException e){
         System.err.println("File read error");
         e.printStackTrace();
      }
      while(a!=null){
         try{
            buff.add(buff.size()+1,a);
            a=br.readLine();
         } catch (IOException e){
            System.err.println("File read error");
            e.printStackTrace();
         }
      }
      for(int i=1;i<=buff.size()/2;i++){
         names.add(i,buff.get(i));
      }
      for(int i=1;i<=buff.size()/2;i++){
         try{
            scores.add(i,Integer.parseInt(buff.get(buff.size()/2+i)));
         } catch (NumberFormatException e){
            System.err.println(buff.get(i)+"parse int error");
            scores.add(i,0);
         }
      }



      setBackground(new Color(206,243,253));
      removeAll();
      if(names.size()!=scores.size()){
         throw new HiScoreListSyncException("ERROR: HiScore List Size Mismatch.");
      } else if (names.size()>HI_SCORES){
         throw new HiScoreListSyncException("ERROR: HiScore List Size Exceeds
"+HI_SCORES+".");
      }
      checkHiScore(score);
   }
   //exception
```

```
public class HiScoreListSyncException extends RuntimeException{
    public HiScoreListSyncException(String s){
        super(s);
    }//end constructor
}
//functions
private void checkHiScore(int score){
    int size=scores.size();
    if(size==0){
        enterName(1,score);
        //return;
    } else {
        int i=size;
        for(;i>0&&score>scores.get(i);i--){//"outputs" index of higher score
        }
        enterName(i+1,score);
        /*if(i<HI_SCORES){
            enterName(i+1,score);
            return;
        }*/
    }
    //enterName(HI_SCORES+1,score);
}
//.....
String name="";
//......
public void keyPressed(KeyEvent e){
    if(hasInput){
        char a=e.getKeyChar();
        //System.out.println(a);
        if(name=="<Please Type Your Name>"){
            name="";
        }
        if(a=='\n' || a=='\r'){
            hasInput=false;
        } else if (a=='\b' && !name.isEmpty()){
            name = new String(name.substring(0, name.length()-1));
        } else if (name.length()==20){
            name = new String(name.substring(0, 19) + a);
        } else if (a>=32 && a<=126){
            name = new String(name + a);
        }
    }
}
private void enterName(int index,int score){//index is where to insert entry
    yay.setVisible(false);
    yay.setFont(YAY_FONT);
```

```
yay.setColor(new Color(0,237,0));
add(yay,ctrX(yay),YAY_Y_OFFSET+yay.getHeight());
hiScoreDisp.setFont(HI_SCORE_FONT);
double fontSizeY=hiScoreDisp.getHeight();
double curY=HI_SCORES_Y_OFFSET+fontSizeY;//current Y
add(hiScoreDisp,ctrX(hiScoreDisp),curY);
curY+=HI_SCORES_SPACING+fontSizeY;
for(int i=1;i<=names.size();i++){
   if(i<index){
      HSNames[i]=new GLabel(i+". "+names.get(i));
      HSScores[i]=new GLabel(scores.get(i)+"");
      HSNames[i].setFont(HI_SCORE_FONT);
      HSScores[i].setFont(HI_SCORE_FONT);
      add(HSNames[i],NAMES_X_OFFSET,curY);
      add(HSScores[i],SCORES_X_OFFSET-HSScores[i].getWidth(),curY);
   } else {
      HSNames[i]=new GLabel((i+1)+". "+names.get(i));
      HSScores[i]=new GLabel(scores.get(i)+"");
      HSNames[i].setFont(HI_SCORE_FONT);
      HSScores[i].setFont(HI_SCORE_FONT);
      add(HSNames[i],NAMES_X_OFFSET,curY);
      add(HSScores[i],SCORES_X_OFFSET-HSScores[i].getWidth(),curY);
   }

   curY+=HI_SCORES_SPACING+fontSizeY;
}
if(index<=HI_SCORES){
   yay.setVisible(true);
   //if(!names.isEmpty()){
   for(int i=names.size();i>=index;i--){
      curY=HSNames[i].getY();
      for(int j=0;j<=HI_SCORES_SPACING+fontSizeY;j++){
         HSNames[i].move(0,1);
         HSScores[i].move(0,1);
         pause(1);
      }
   }
   //}
   /*test=new GLabel("ENTER NAME");
   add(test,200,200);
   GCanvas canvas = getGCanvas();
   canvas.addKeyListener(this);
   hasInput=true;
   while(hasInput){
      pause(10);
   }
   remove(test);*/
```

```
        //String name="Game#"+gameNo;//"HI--PMC";//temporary
        name="<Please Type Your Name>";
        GLabel newName=new GLabel(index+". "+name);
        //gameNo++;
        GLabel newScore=new GLabel(score+"");
        newName.setFont(HI_SCORE_FONT);
        newScore.setFont(HI_SCORE_FONT);
        add(newScore,-newScore.getWidth(),curY);
        add(newName,NAMES_X_OFFSET-SCORES_X_OFFSET,curY);
        newName.setColor(new Color(206,206,0));
        newScore.setColor(new Color(206,206,0));
        for(int i=0;i<=SCORES_X_OFFSET;i++){
            newName.move(1, 0);
            newScore.move(1, 0);
            pause(1);
        }
        //entry name
        hasInput=true;
        while(hasInput){
            double x=newName.getX();
            double y=newName.getY();
            newName.setLabel(index+". "+name);
            remove(newName);
            add(newName,x,y);
            pause(10);
        }
        //done input
        names.add(index, name);
        scores.add(index, score);
        if(names.size()>HI_SCORES){
            names.remove(6);
            scores.remove(6);
            while(HSNames[5].getX()<WIDTH){
                HSNames[5].move(1, 0);
                HSScores[5].move(1, 0);
                pause(1);
            }
            remove(HSNames[5]);
            remove(HSScores[5]);
        }
    }



    //HS File Overwrite
    String filePath=new File("").getAbsolutePath();
```

```
    FileWriter out;
    BufferedWriter bw = null;
    try{
       out=new FileWriter(filePath+"/Scores.dat");
       bw=new BufferedWriter(out);
    } catch(IOException e){
       System.err.println("FILE READ ERROR\n");
       e.printStackTrace();
    }
    MyList<String> buff=new MyList<>();
    buff.createList();
    for(int i=1;i<=names.size();i++){
       buff.add(i,names.get(i));
    }
    for(int i=1;i<=scores.size();i++){
       buff.add(buff.size()+1,scores.get(i).toString());
    }
    for(int i=1;i<=buff.size();i++){
       try{
          bw.append(buff.get(i));
          bw.newLine();
       } catch (IOException e){
          System.out.println("File Write Error");
          e.printStackTrace();
       }
    }
    try{//necessary to update file
       bw.flush();
       bw.close();
    } catch(IOException e){
       ;
    }//System.out.println(filePath);
    backToLevel();
}
private void backToLevel(){
    resetScore();
    GLabel click=new GLabel("-Click Anywhere to Play Again-");
    click.setFont(CLICK_FONT);
    add(click,ctrX(click),HEIGHT-click.getHeight()-CLICK_Y_OFFSET);
    waitForClick();
    screenLevel();
}

//"main" function
public void init(){
    addMouseListeners();
    GCanvas canvas = getGCanvas();
```

```
canvas.addKeyListener(this);
for(int i=0;i<MAX_NTURNS_DRAWN;i++){
   livesBall[i] = new GOval(BALL_RADIUS,BALL_RADIUS);
   livesBall[i].setFilled(true);
}
names.createList();
scores.createList();

//pause screen
overlay.setFilled(true);
overlay.setFillColor(new Color(0,0,0,200));

//getting high scores list
//HS File Read (creation if none)
names.createList();
scores.createList();
String filePath=new File("").getAbsolutePath();
FileReader in;
BufferedReader br = null;
try{
   in=new FileReader(filePath+"/Scores.dat");
   br=new BufferedReader(in);
} catch(IOException e){
   File a=new File(filePath+"/Scores.dat");
   try{
      a.createNewFile();
      in=new FileReader(filePath+"/Scores.dat");
      br=new BufferedReader(in);
   } catch (IOException ee){
      System.err.println("FILE READ ERROR\n");
      ee.printStackTrace();
   }
}
MyList<String> buff=new MyList<>();
buff.createList();
String a="";
try{
   a=br.readLine();
} catch (IOException e){
   System.err.println("File read error");
   e.printStackTrace();
}
while(a!=null){
   try{
      buff.add(buff.size()+1,a);
      a=br.readLine();
   } catch (IOException e){
```

```
            System.err.println("File read error");
            e.printStackTrace();
         }
      }
      for(int i=1;i<=buff.size()/2;i++){
         names.add(i,buff.get(i));
      }
      for(int i=1;i<=buff.size()/2;i++){
         try{
            scores.add(i,Integer.parseInt(buff.get(buff.size()/2+i)));
         } catch (NumberFormatException e){
            System.err.println(buff.get(i)+"parse int error");
            scores.add(i,0);
         }
      }
   }
   public void run(){
      screenLevel();
      screenHiScore();
   }


   //additional for integration
   private static boolean isAutoPause=true;
   public void pauseGame(){
      isMouseInScreen=false;
}
   /**
    * @param flag
    * if this is set to true, the game pauses once the mouse leaves the
    *  play area. The game can also be paused using the pauseGame() method.
    */
   public void setAutoPause(boolean flag){
      isAutoPause=flag;
   }
   @Override
   public void stop() {
      super.stop();
      //I'll just add methods here
      bgm.stop();
      bounceClip.stop();
   }

}
```

B) List.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.breakout;

/*
 * File: List.java
 * ------------------
 * This is the List ADT definition
 */


public interface List<E>{


  public void createList();
  // precondition: none
  // postcondition: Create an empty list


  public void add(int index, E item) throws ListIndexOutOfBoundsException, ListFullException;
// precondition: index (to be added) is within the position of the list of items,
1<=index<=size()+1
// postcondition: Insert item at position index of a list
// if 1<=index<= size()+1. If index <= size(), items
// at position index onwards are shifted one position
// to the right. Throws an exception when index is out of range
// or if the item cannot be placed on the list (list full).


 // public void replace(int index, E item) throws ListIndexOutOfBoundsException,
ListEmptyException;
// precondition: index (to be replaced) is within the position of the list of items,
1<=index<=size()+1
// postcondition: Replace item at position index of a list
// if 1<=index<= size()+1. If index <= size(), items
// at position index onwards are shifted one position
// to the right. Throws an exception when index is out of range
// or if the item cannot be placed on the list (list empty).


  public void remove(int index) throws ListIndexOutOfBoundsException;
```

```
// precondition: index (to be removed) is within the position of the list of items,
1<=index<=size()
// postcondition: Remove item at position index of a list
// if 1<=index<= size(). Items at position
// index+1 onwards are shifted one position to the left
// Throws an exception when index is out of range, or if list is empty.



 public boolean isEmpty();
// precondition: none
// postcondition: Determine if a list is empty



public E get(int index) throws ListIndexOutOfBoundsException;
// precondition: index is within the position of the list of items, 1<=index<=size()
// postcondition: Returns item at position index of
// a list if 1<=index<=size(). Throws an exception if index is out of range.



public int size();
// precondition: none
// postcondition: Returns number of items in a list

}
```

C) ListEmptyException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.breakout;

/**
 *
 * @author Administrator
 */
class ListEmptyException extends RuntimeException{
    public ListEmptyException(String s){
        super(s);
```

```
    }//end constructor
}
```

D) ListFullException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.breakout;

/**
 *
 * @author Administrator
 */
public class ListFullException extends RuntimeException{
    public ListFullException(String s){
        super(s);
    }//end constructor
} //end ListException
```

E) ListIndexOutOfBounds.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.breakout;

/**
 *
 * @author Administrator
 */
public class ListIndexOutOfBoundsException extends IndexOutOfBoundsException{
      public ListIndexOutOfBoundsException(String s){
        super(s);
      }//end constructor
} //end ListIndexOutofBoundsExceptio
```

F) MyList.java

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.breakout;

/*
 * File: MyList.java
 * ------------------
 * This is the List ADT implementation
 */


import acm.program.*;
import acm.util.*;


public class MyList<E> implements List<E>{

    /// private data fields
    private final int MAX_LIST = 10;     // max length of list
    private E[] items;                   // array of list items
    private int NumItems;                // current size of list

    /// list items are already allocated above with T items[MAX_LIST]
    @SuppressWarnings("unchecked")
    public void createList(){
          items = (E[])new Object[MAX_LIST];
          NumItems = 0;
    }

    public void add(int index, E item) throws ListIndexOutOfBoundsException,
ListFullException{
        if ( index > 0 && index <= NumItems + 1){
            if (NumItems == MAX_LIST){
                throw new ListFullException("ERROR: List Already Full");
            }
            else { // insert the element
                int j = NumItems;
                while(j >= index){
                    items[j] = items[j - 1];
                    j--;
                }
                items[index-1] = item;
                NumItems++;
            }
        }
```

```
        else
          throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of Bounds");
    }

    public void remove(int index) throws ListIndexOutOfBoundsException{
        if ( index > 0 && index <= NumItems){
                for(int i = index; i < NumItems; i++){
                    items[i-1] = items[i];
                }
                NumItems--;
           }
        else
          throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of Bounds");
    }

    public boolean isEmpty(){
          return NumItems == 0;
    }

    public E get(int index) throws ListIndexOutOfBoundsException{
        if ( index > 0 && index <= NumItems){
            return items[index-1];
        }
        else
          throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of Bounds");
    }

    public int size(){
          return NumItems;
    }

}
```

REFERENCES

1. E Roberts. *Art and Science of Java*. Pearson; 2013.
2. E Roberts, M Sahami, and M Stepp, *CS 106A: Programming Methodology (Java) Handouts*, Stanford University.