CSCI-C 212 Final Exam Corrections

Score: 70/100

What I wrote down:

World.java

This perfectly resembles my World.java from my project

————————————————-

import java.awt.\*;

interface World {

public boolean hasEnded();

public void draw(Graphics g);

public void update();

public void keh(String key);

}

—————————————————

BigBang.java

BigBang is supposed to extend a Component and implement an ActionListener and a KeyListener.

————————————————-

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

class BigBang extends ActionListener implements ActionEvent,

KeyEvent {

World world;

Timer timer;

public BigBang(World w) {

super(); // unnecessary

this.world = w;

this.timer = new Timer(1000 “forot to add a ‘, this) );

}

public void start() {

this.timer.start();

update(); //needs to repaint

}

public void update() { //this method is useless

repaint();

}

public void draw (Graphics g) { // not needed

this.world.draw(g);

}

public void keyPressed(KeyEvent e) {

int code;

switch (code) {

case 37: this.world.keh( "left" );

break;

case 38: this.world.keh( "right" );

break;

case 39: this.world.keh( "up" );

break;

case 40: this.world.keh( "down" );

break;

case 41: this.world.keh( "rotate" );

break;

}

}

public void keyReleased(KeyEvent e) { }

public void keyReleased(KeyEvent e) { } // need to implement keyTyped()

//Need a paintComponent and actionPerformed methods

}

—————————————————

Tetris.java

————————————————-

import java.awt.\*;

import javax.swing.\*;

import java.util.\*;

public class Tetris implements World {

Tetromino current;

public Tetris() {

super();

this.current = new T (100, 0);

}

public boolean hasEnded() {

return false;

}

public void draw(Graphics g) {

this.current.draw(g);

}

public void update() {

this.keh("down");

}

public void keh(String key) {

if (key.equals("left")) {

this.current.moveLeft();

} else if (key.equals("right")) {

this.current.moveRight();

} else if (key.equals("up")) {

this.current.moveUp();

} else if (key.equals("down")) {

this.current.moveDown();

} else if (key.equals("rotate")) {

this.current.rotate();

} else {

this.current.moveDown();

}

}

public static void main(String[] args) {

JFrame a = new JFrame();

a.setSize(200, 500);

BigBang b = new BigBang( new Tetris() );

a.getContentPane().add( b );

a.addKeyListener( b );

b.start();

a.setVisibility(true); //setVisible not Visibility

}

}

—————————————————

Tetromino.java

————————————————-

import java.awt.\*;

import java.util.\*;

public abstract class Tetromino extends ArrayList<Block> {

int x, y;

public Tetromino(int x, int y) {

this.x = x;

this.y = y;

}

public void draw(Graphics g) {

//need b.draw(g) or something

}

public void moveLeft() {

this.x -= Block.SIZE;

}

public void moveRight() {

this.x += Block.SIZE;

}

public void moveUp() {

this.y -= Block.SIZE;

}

public void moveDown() {

this.y += Block.SIZE;

}

public void rotate(); //needs to be abstract

}

—————————————————

Block.java

This is fine

————————————————-

import java.awt.\*;

import java.util.\*;

public class Block {

final static int SIZE = 30;

int x, y, w, h;

Color c;

public Block(int x, int y, Color c) {

this.x = x;

this.y = y;

this.w = Block.SIZE;

this.h = Block.SIZE;

this.c = c;

}

public void moveRight() { this.x += Block.SIZE; }

public void moveLeft( ) { this.x -= Block.SIZE; }

public void moveUp( ) { this.y -= Block.SIZE; }

public void moveDown( ) { this.y += Block.SIZE; }

}

—————————————————

T.java

————————————————-

import java.awt.\*;

class T extends Tetromino {

final int[][][] shape =

{

{ { 0, 1, 0 },

{ 1, 1, 1 },

{ 0, 0, 0 }

},

{ { 0, 1, 0 },

{ 0, 1, 1 },

{ 0, 1, 0 }

},

{ { 0, 0, 0 },

{ 1, 1, 1 },

{ 0, 1, 0 }

},

{ { 0, 1, 0 },

{ 1, 1, 0 },

{ 0, 1, 0 }

},

};

public void draw(Graphics g) { //not needed

g.setColor(Red);

g.fillRect(this.x, this.y, this.w, this.h);

g.setColor(Black);

g.makeRect(this.x, this.y, this.w, this.h);

}

public void moveLeft() {

this.x -= Block.SIZE;

for (Block b : this) b.moveLeft();

}

public void moveRight() {

this.x += Block.SIZE;

for (Block b : this) b.moveRight();

}

public void moveUp() {

this.y -= Block.SIZE;

for (Block b : this) b.moveUp();

}

public void moveDown() {

this.y += Block.SIZE;

for (Block b : this) b.moveDown();

}

public void rotate() { //needs finished

}

public T(int x, int y) {

super(); //super(x,y)

this.x = x; //this.reset();

this.y = y;

}

}

—————————————————

Corrections:

This is my stage two of my project

World.java

————————————————-

import java.awt.\*;

interface World {

public boolean hasEnded();

public void draw(Graphics g);

public void update();

public void keh(String key);

}

—————————————————

BigBang.java

————————————————-

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

public class BigBang extends JComponent

implements ActionListener,

KeyListener {

World world;

Timer timer;

public BigBang(World w) {

this.world = w;

this.timer = new Timer(1000, this);

}

public BigBang(int delay, World w) {

this.world = w;

this.timer = new Timer(delay, this);

}

public void start() {

this.timer.start();

this.repaint();

}

public void keyPressed(KeyEvent e) {

int code = e.getKeyCode();

switch (code) {

case 37: // left

this.world.keh( "left" );

break;

case 38: // up

this.world.keh( "up" );

break;

case 39: // right

this.world.keh( "right" );

break;

case 40: // down

this.world.keh( "down" );

break;

default: // what?!

this.world.keh( "rotate" );

break;

}

this.repaint();

}

public void keyReleased(KeyEvent e) { }

public void keyTyped(KeyEvent e) { }

public void actionPerformed(ActionEvent e) {

this.world.update();

this.repaint();

}

public void paintComponent(Graphics g) {

this.world.draw(g);

}

}

—————————————————

Tetris.java

————————————————-

import java.awt.\*;

import javax.swing.\*;

public class Tetris implements World {

Tetromino current;

public Tetris() {

super();

this.current = new T (250, 0);

}

// Returns a random integer between min (included) and max (included)

// Using Math.round() will give you a non-uniform distribution!

public void newPiece(int min, int max) {

int pieceChoice = (int) Math.floor(Math.random() \* (max - min + 1)) + min;

if (pieceChoice == 1) {

this.current = new Z (20, 50);

} else if (pieceChoice == 2) {

this.current = new T (20, 50);

} else if (pieceChoice == 3) {

this.current = new S (20, 50);

} else if (pieceChoice == 4) {

this.current = new O (20, 50);

} else if (pieceChoice == 5) {

this.current = new L (20, 50);

} else if (pieceChoice == 6) {

this.current = new J (20, 50);

} else if (pieceChoice == 7) {

this.current = new I (20, 50);

} else {

}

}

public boolean hasEnded() {

return false;

}

public void draw(Graphics g) {

this.current.draw(g);

}

private int count;

public void update() {

this.count += 1;

this.keh("down");

System.out.println( count );

}

public void keh(String key) {

if (key.equals("left")) {

this.current.moveLeft();

} else if (key.equals("right")) {

this.current.moveRight();

} else if (key.equals("up")) {

this.current.moveUp();

} else if (key.equals("down")) {

this.current.moveDown();

} else if (key.equals("rotate")) {

this.current.rotate();

} else {

}

}

public static void main(String[] args) {

JFrame a = new JFrame();

a.setSize(500, 500);

a.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

BigBang b = new BigBang( new Tetris() );

a.addKeyListener( b );

a.getContentPane().add( b );

a.setVisible(true);

b.start();

}

}

—————————————————

Tetromino.java

————————————————-

import java.awt.\*;

import java.util.\*;

public abstract class Tetromino extends ArrayList<Block> {

protected int x, y;

protected int direction; // will take values 0, 1, 2, 3

public Tetromino(int x, int y) {

super(); // since I am in fact an ArrayList of Blocks

this.x = x;

this.y = y;

}

public void draw(Graphics g) {

for (Block b : this)

b.draw(g);

}

public void moveLeft() {

this.x -= Block.SIZE;

for (Block b : this) b.moveLeft();

}

public void moveRight() {

this.x += Block.SIZE;

for (Block b : this) b.moveRight();

}

public void moveUp() {

this.y -= Block.SIZE;

for (Block b : this) b.moveUp();

}

public void moveDown() {

this.y += Block.SIZE;

for (Block b : this) b.moveDown();

System.out.println("this.y = " + this.y);

System.out.println("this.x = " + this.x);

}

public abstract void rotate();

}

—————————————————

Block.java

————————————————-

import java.awt.\*;

public class Block {

final static int SIZE = 20;

private int x, y, w, h;

private Color c;

public Block(int x, int y, Color c) {

this.x = x;

this.y = y;

this.w = Block.SIZE;

this.h = Block.SIZE;

this.c = c;

}

public void draw(Graphics g) {

g.setColor(this.c);

g.fillRect(this.x, this.y, this.w, this.h);

g.setColor(Color.BLACK);

g.drawRect(this.x, this.y, this.w, this.h);

}

public void moveLeft( ) { this.x -= Block.SIZE; }

public void moveRight() { this.x += Block.SIZE; }

public void moveUp( ) { this.y -= Block.SIZE; }

public void moveDown( ) { this.y += Block.SIZE; }

}

—————————————————

T.java

————————————————-

import java.awt.\*;

public class T extends Tetromino {

final static int[][][] shape =

{

{ { 0, 1, 0 },

{ 1, 1, 1 },

{ 0, 0, 0 }

}, // index 0 in the shape array

{ { 0, 1, 0 },

{ 0, 1, 1 },

{ 0, 1, 0 }

}, // index 0 in the shape array

{ { 0, 0, 0 },

{ 1, 1, 1 },

{ 0, 1, 0 }

}, // index 0 in the shape array

{ { 0, 1, 0 },

{ 1, 1, 0 },

{ 0, 1, 0 }

}, // index 0 in the shape array

};

public void rotate() {

this.direction += 1; // reminder a op= b; is the same as a = a op b;

this.direction %= 4;

// what else

this.reset();

}

private void reset() {

this.clear();

for (int i = 0; i < shape[direction].length; i++)

for (int j = 0; j < shape[direction][i].length; j++)

if (shape[direction][i][j] == 1)

this.add( new Block( this.x + Block.SIZE \* j,

this.y + Block.SIZE \* i,

Color.MAGENTA

)

);

}

public T(int x, int y) {

super(x, y);

this.reset();

}

}

—————————————————