package model;  
  
import java.awt.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
import static userInterface.GameArea.*COLORS*;  
  
public enum Figure {  
 // каждый элемент - экземпляр класса Figure  
 *I1*(0,1, 1,1, 2,1, 3,1),// ----  
 *I2*(1,0, 1,1, 1,2, 1,3),  
 *J1*(1,0, 1,1, 0,2, 1,2),//Г  
 *J2*(0,0, 0,1, 1,1, 2,1),  
 *J3*(1,0, 2,0, 1,1, 1,2),  
 *J4*(0,1, 1,1, 2,1, 2,2),  
 *L1*(1,0, 1,1, 1,2, 2,2),//L  
 *L2*(0,1, 1,1, 2,1, 0,2),  
 *L3*(0,0, 1,0, 1,1, 1,2),  
 *L4*(2,0, 0,1, 1,1, 2,1),  
 *O1*(0,0, 1,0, 0,1, 1,1),//square  
 *S1*(1,1, 2,1, 0,2, 1,2),//z  
 *S2*(0,0, 0,1, 1,1, 1,2),  
 *T1*(0,1, 1,1, 2,1, 1,2),//T  
 *T2*(1,0, 0,1, 1,1, 1,2),  
 *T3*(1,0, 0,1, 1,1, 2,1),  
 *T4*(1,0, 1,1, 2,1, 1,2),  
 *Z1*(0,1, 1,1, 1,2, 2,2),//z  
 *Z2*(2,0, 1,1, 2,1, 1,2);  
  
 // для каждой фигуры в списке хранятся свои координаты  
 public List<Coordinate> dots;  
 public final Coordinate top;  
 public final Coordinate bottom;  
  
  
 // в перечислении enum могут быть только конструкторы с модификатором доступа private  
 Figure(int... coordinates) {  
 dots = new ArrayList<Coordinate>();  
 for (int j = 0; j < coordinates.length; j += 2) {  
 dots.add(new Coordinate(coordinates[j], coordinates[j+1]));  
 }  
 top = setTop();  
 bottom = setBottom();  
 }  
  
 private Coordinate setTop() {  
 int x = dots.get(0).x;  
 int y = dots.get(0).y;  
 for(Coordinate coordinate: dots) {  
 if(x > coordinate.x) x = coordinate.x;  
 if(y > coordinate.y) y = coordinate.y;  
 }  
 return new Coordinate(x,y);  
 }  
  
 private Coordinate setBottom() {  
 int x = dots.get(0).x;  
 int y = dots.get(0).y;  
 for(Coordinate coordinate: dots) {  
 if(x < coordinate.x) x = coordinate.x;  
 if(y < coordinate.y) y = coordinate.y;  
 }  
 return new Coordinate(x,y);  
 }  
  
 public Figure rotateRight() {  
 switch(this) {  
 case *I1*: return *I2*;  
 case *I2*: return *I1*;  
 case *J1*: return *J2*;  
 case *J2*: return *J3*;  
 case *J3*: return *J4*;  
 case *J4*: return *J1*;  
 case *O1*: return *O1*;  
 case *L1*: return *L2*;  
 case *L2*: return *L3*;  
 case *L3*: return *L4*;  
 case *L4*: return *L1*;  
 case *S1*: return *S2*;  
 case *S2*: return *S1*;  
 case *T1*: return *T2*;  
 case *T2*: return *T3*;  
 case *T3*: return *T4*;  
 case *T4*: return *T1*;  
 case *Z1*: return *Z2*;  
 case *Z2*:  
 default: return *Z1*;  
 }  
 }  
  
 public Figure rotateLeft() {  
 return rotateRight().rotateRight().rotateRight();  
 }  
  
 public static Figure getRandom() {  
 switch ((int)(Math.*random*()\*19)) {  
 case 0: return *J1*;  
 case 1: return *J2*;  
 case 2: return *J3*;  
 case 3: return *J4*;  
 case 4: return *I1*;  
 case 5: return *I2*;  
 case 6: return *Z1*;  
 case 7: return *Z2*;  
 case 8: return *T1*;  
 case 9: return *T2*;  
 case 10: return *T3*;  
 case 11: return *T4*;  
 case 12: return *S1*;  
 case 13: return *S2*;  
 case 14: return *O1*;  
 case 15: return *L1*;  
 case 16: return *L2*;  
 case 17: return *L3*;  
 case 18:  
 default: return *L4*;  
 }  
 }  
  
 public int getFigureColor() {  
 switch(this) {  
 case *I1*:  
 case *I2*: return 2;  
 case *J1*:  
 case *J2*:  
 case *J3*:  
 case *J4*: return 3;  
 case *O1*: return 4;  
 case *L1*:  
 case *L2*:  
 case *L3*:  
 case *L4*: return 5;  
 case *S1*:  
 case *S2*: return 6;  
 case *T1*:  
 case *T2*:  
 case *T3*:  
 case *T4*: return 7;  
 case *Z1*:  
 case *Z2*:  
 default: return 8;  
 }  
 }  
}

package model;  
  
public class Coordinate {  
 public final int x;  
 public final int y;  
  
 public Coordinate(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
  
 public Coordinate plus(int cx, int cy) {  
 return new Coordinate(x+cx, y+cy);  
 }  
}

package model;  
  
public interface Mapable {  
  
 int getBoxColor(int x, int y);  
}

package controller;  
  
import model.Coordinate;  
import model.Figure;  
import model.Mapable;  
import userInterface.GameArea;  
import userInterface.GameWindow;  
  
import java.awt.\*;  
  
public class FallFigure {  
  
 private Figure figure; // активная фигура  
 private Coordinate coordinate;  
 private boolean landed;  
 private int ticks;  
 Mapable map;  
  
 public Figure getFigure() {  
 return figure;  
 }  
  
 public Coordinate getCoordinate() {  
 return coordinate;  
 }  
  
 public boolean isLanded() {  
 return landed;  
 }  
  
 public FallFigure(Mapable map) {  
 this.map = map;  
 figure = Figure.*getRandom*();  
 coordinate = new Coordinate(GameArea.*WIDTH* / 2 - 2,-1);//figure.top.y -figure.bottom.y - 1  
 landed = false;  
 ticks = 2;  
 }  
  
 public boolean canPlaceFigure() {  
 return canMoveFigure(figure,0,0);  
 }  
  
 private boolean canMoveFigure(Figure figure, int cx, int cy) {  
 if(coordinate.x + cx + figure.top.x < 0) return false;  
 if(coordinate.x + cx + figure.bottom.x >= GameArea.*WIDTH*) return false;  
 //if(coordinate.y + cy + figure.top.y < 0) return false;  
 if(coordinate.y + cy + figure.bottom.y >= GameArea.*HEIGHT*) return false;  
 for(Coordinate dot: figure.dots) {  
 if (map.getBoxColor(coordinate.x + dot.x + cx, coordinate.y + dot.y + cy) > 0)  
 return false;  
 }  
 return true;  
 }  
  
 public void moveFigure(int cx, int cy) {  
 if(canMoveFigure(figure,cx,cy))  
 coordinate = coordinate.plus(cx,cy);  
 else if (cy == 1) {  
 if(ticks > 1) ticks--; //>0  
 else landed = true;  
 }  
 }  
  
 public int colorFigure() {  
 return figure.getFigureColor();  
 }  
  
 public void turnFigure(int direction) {  
 Figure rotated = direction == 1 ? figure.rotateRight() : figure.rotateLeft();  
 if(canMoveFigure(rotated,0,0)) {  
 figure = rotated;  
 } else if(canMoveFigure(rotated,1,0)) {  
 figure = rotated;  
 moveFigure(1,0);  
 } else if(canMoveFigure(rotated,-1,0)) {  
 figure = rotated;  
 moveFigure(-1,0);  
 return;  
 } else if (canMoveFigure(rotated,0,-1)) {  
 figure = rotated;  
 moveFigure(0,-1);  
 }  
 }  
}

package userInterface;  
  
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
  
public class Box extends JPanel {  
  
 private int color;  
  
 public int getColor() {  
 return color;  
 }  
  
 public Box(int x, int y) {  
 color = 0;  
 setBounds(x \* GameArea.*SIZE*, y \* GameArea.*SIZE*,  
 GameArea.*SIZE*,GameArea.*SIZE*);  
 setBackground(GameArea.*COLORS*[0]);  
 }  
  
 private void setBoxBack() throws IOException {  
 BufferedImage image = ImageIO.*read*(new File("background.jpg"));  
 JLabel label = new JLabel(new ImageIcon(image ));  
 this.add(label);  
 }  
  
 public void setColor(int color) {  
 this.color = color;  
 if(color >= 0 && color < GameArea.*COLORS*.length)  
 setBackground(GameArea.*COLORS*[color]);  
 }  
}

package userInterface;  
  
import java.awt.\*;  
  
public class GameArea {  
 public static final int *WIDTH* = 10;  
 public static final int *HEIGHT* = 20;  
  
 public static final int *SIZE* = 30;  
  
 public static final Color[] *COLORS* = {  
 Color.*DARK\_GRAY*,  
 Color.*GREEN*,  
 Color.*CYAN*,  
 Color.*BLUE*,  
 Color.*ORANGE*,  
 Color.*YELLOW*,  
 Color.*GREEN*,  
 Color.*MAGENTA*,  
 Color.*RED* };  
}

package userInterface;  
  
import controller.FallFigure;  
import model.Coordinate;  
import model.Figure;  
import model.Mapable;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
  
public class GameWindow extends JFrame implements Runnable, Mapable {  
  
 private Box[][] box;  
 private FallFigure fallfigure;  
  
 public GameWindow() {  
 box = new Box[GameArea.*WIDTH*][GameArea.*HEIGHT*];  
 initForm();  
 initBox();  
 addKeyListener(new KeyAdapter());  
 TimeAdapter timeAdapter = new TimeAdapter();  
 Timer timer = new Timer(180, timeAdapter);  
 timer.start();  
 }  
  
 public void addFigure() {  
 fallfigure = new FallFigure(this);  
 if (fallfigure.canPlaceFigure())  
 showFigure();  
 else {  
 setVisible(false);  
 dispose();  
 }  
 }  
  
 private void initForm() { //инициализация формы  
 setSize(GameArea.*WIDTH* \* GameArea.*HEIGHT* + 118,  
 GameArea.*HEIGHT* \* GameArea.*SIZE* + 47);  
  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
 setTitle("TETRIS");  
 setLayout(null);  
 setVisible(true);  
 }  
  
 private void initBox() {  
 for(int x =0 ; x < GameArea.*WIDTH*; x++) {  
 for(int y = 0; y < GameArea.*HEIGHT*; y++) {  
 box[x][y] = new Box(x, y);  
 add(box[x][y]);  
 }  
 }  
 }  
  
 @Override  
 public void run() {  
 repaint();  
 }  
  
 private void showFigure() {  
 int colorNumber = fallfigure.colorFigure();  
 showFigure(colorNumber);  
  
 //showFigure(1);  
 }  
  
 private void hideFigure() {  
 showFigure(0);  
 }  
  
 private void showFigure(int color) { //чтобы показать/скрыть фигурку  
 for (Coordinate dot: fallfigure.getFigure().dots) {  
 setBoxColor(fallfigure.getCoordinate().x + dot.x, fallfigure.getCoordinate().y + dot.y,color);  
 }  
 }  
  
 private void setBoxColor(int x, int y, int color) {  
 if(x < 0 || x>= GameArea.*WIDTH*) return;  
 if(y < 0 || y>= GameArea.*HEIGHT*) return;  
 box[x][y].setColor(color);  
 }  
  
 public int getBoxColor(int x, int y) {  
 if(x < 0 || x>= GameArea.*WIDTH*) return -1;  
 if(y < 0 || y>= GameArea.*HEIGHT*) return -1;  
 return box[x][y].getColor();  
 }  
  
 private void moveInFall(int cx, int cy) {  
 hideFigure();  
 fallfigure.moveFigure(cx,cy);  
 showFigure();  
 }  
  
 private void turnInFall(int direction) {  
 hideFigure();  
 fallfigure.turnFigure(direction);  
 showFigure();  
 }  
  
 // обработка нажатых клавиш  
 class KeyAdapter implements KeyListener {  
 @Override  
 public void keyTyped(KeyEvent e) {}  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch (e.getKeyCode()) {  
 case KeyEvent.*VK\_LEFT*: moveInFall(-1,0); break;  
 //case KeyEvent.VK\_A: moveInFall(-1,0); break;  
 case KeyEvent.*VK\_RIGHT*: moveInFall(+1,0); break;  
 //case KeyEvent.VK\_D: moveInFall(+1,0); break;  
 case KeyEvent.*VK\_DOWN*: moveInFall(0,+1); break;  
 //case KeyEvent.VK\_S: moveInFall(0,+1); break;  
 case KeyEvent.*VK\_SPACE*: turnInFall(1); break;  
 case KeyEvent.*VK\_UP*: turnInFall(2); break;  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {}  
 }  
  
 private void deleteLines() {  
 for(int y = GameArea.*HEIGHT*-1; y>= 0; y--) {  
 if (isCompleteLine(y)) {  
 dropLine(y);  
 }  
 }  
 }  
  
 private void dropLine(int y) {  
 for (int movey = y - 1; movey >= 0; movey--) {  
 for(int x = 0; x < GameArea.*WIDTH*; x++)  
 setBoxColor(x,movey+1,getBoxColor(x,movey)); //в то, что ниже, копируем то, что выше  
 }  
 for(int x = 0; x < GameArea.*WIDTH*; x++)  
 setBoxColor(x,0,0);  
 }  
  
 private boolean isCompleteLine(int y) {  
 for (int x = 0; x < GameArea.*WIDTH*; x++)  
 if (getBoxColor(x, y) != 2 && getBoxColor(x, y) != 3 &&  
 getBoxColor(x, y) != 4 && getBoxColor(x, y) != 5 &&  
 getBoxColor(x, y) != 6 && getBoxColor(x, y) != 7 &&  
 getBoxColor(x, y) != 8 && getBoxColor(x, y) != 9) return false;  
 return true;  
 }  
  
 class TimeAdapter implements ActionListener {  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 moveInFall(0,1);  
 if (fallfigure.isLanded()) {  
 deleteLines();  
 addFigure();  
 }  
 }  
 }  
}

package userInterface;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class GameAbout extends JFrame implements Runnable {  
 private JPanel AboutPanel;  
 private JLabel HeaderJLabel;  
 private JTextArea text;  
  
 private Image image;  
  
 public GameAbout() {  
  
 image = Toolkit.*getDefaultToolkit*().getImage("D:\\IntelliJ IDEA\\Tetris2D\\about.jpg");  
 this.setContentPane(new JPanel() {  
 @Override  
 public void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 g.drawImage(image, -100, 0, this);  
 }  
 });  
  
 AboutPanel = new JPanel();  
 AboutPanel.setLayout(new BoxLayout(AboutPanel, BoxLayout.*Y\_AXIS*));  
 AboutPanel.setBackground(new Color(255, 255, 255, 0));  
  
 HeaderJLabel = new JLabel("ABOUT");  
 HeaderJLabel.setFont(new Font("Microsoft JhengHei", Font.*BOLD*,32));  
 HeaderJLabel.setForeground(Color.*BLACK*);  
 HeaderJLabel.setAlignmentX(JComponent.*LEFT\_ALIGNMENT*);  
  
 text = new JTextArea("This Tetris game was written by me, Ryzhkova Daria, the IKPI-94 group, in the Java programming language");  
 text.setFont(new Font("Microsoft JhengHei", Font.*BOLD*,25));  
 text.setAlignmentX(JComponent.*LEFT\_ALIGNMENT*);  
 text.setSize(270,600);  
 text.setBackground(new Color(255, 255, 255, 127));  
 text.setLineWrap(true);  
 text.setEditable(false);  
  
 //Добавляем label на панель  
 AboutPanel.add(HeaderJLabel);  
 AboutPanel.add(text);  
  
 AboutPanel.setBorder(BorderFactory.*createEmptyBorder*(50, 25, 0, 0));  
  
 //Добавляем панель на главное окно  
 add(AboutPanel);  
  
 //Устанавливаем размеры главного окна  
 setPreferredSize(new Dimension(350, 600));  
 setSize(350,600);  
 setTitle("TETRIS ABOUT");  
 setDefaultCloseOperation(WindowConstants.*DISPOSE\_ON\_CLOSE*);  
 //Делаем окно по центру  
 setLocationRelativeTo(null);  
 //Делаем видимым главное окно  
 pack();  
 setVisible(true);  
 }  
  
 @Override  
 public void run() { repaint(); }  
}

package userInterface;  
  
import javax.swing.\*;  
  
public class GameOver {  
 private JTable table1;  
 private JPasswordField passwordField1;  
 private JButton button1;  
 private JButton button2;  
 private JButton button3;  
}

package userInterface;  
  
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
  
import static com.sun.deploy.uitoolkit.ToolkitStore.*dispose*;  
  
public class StartForm extends JFrame implements Runnable {  
 private JPanel mainPanel;  
 private JButton startGameButton;  
 private JButton leaderboardButton;  
 private JTextArea TETRISTextArea;  
 private JLabel mainJLabel;  
 private JButton aboutButton;  
 private JButton exitButton;  
 private Image image;  
  
 public StartForm() {  
 // устанавливаем GIF на фон стартовой панели  
 image = Toolkit.*getDefaultToolkit*().getImage("D:\\IntelliJ IDEA\\Tetris2D\\tetris\_main\_back.gif");  
 this.setContentPane(new JPanel() {  
 @Override  
 public void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 g.drawImage(image, -235, 0, this);  
 }  
 });  
  
 //Создаем объект панели, на котором будет отображаться кнопка  
 mainPanel = new JPanel();  
 //Добавляем BoxLayout  
 mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.*Y\_AXIS*));  
 mainPanel.setBackground(new Color(255, 255, 255, 0));  
  
 Image label = Toolkit.*getDefaultToolkit*().getImage("D:\\IntelliJ IDEA\\Tetris2D\\tetris\_label.png");//it must be an image file, otherwise you'll get an exception  
 mainJLabel = new JLabel();  
 mainJLabel.setIcon(new ImageIcon(label));  
 //mainJLabel.setPreferredSize(new Dimension(219,110));  
 //mainJLabel.setSize(300, 100);  
 mainJLabel.setAlignmentX(JComponent.*CENTER\_ALIGNMENT*);  
  
  
 //Создаем кнопку  
 startGameButton = new JButton("START GAME");  
 //Устанавливаем размеры кнопки  
 startGameButton.setPreferredSize(new Dimension(300,100));  
 startGameButton.setBackground(new Color(255, 255, 255, 0));  
 //Выравниваем по центру по оси X  
 startGameButton.setAlignmentX(JComponent.*CENTER\_ALIGNMENT*);  
 //Обрабатываем событие при нажатии на кнопку  
 startGameButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent arg0) {  
 //Создаем диалог  
 GameWindow window = new GameWindow();  
 // Делаем видимым второе окно  
 window.setVisible(true);  
 javax.swing.SwingUtilities.*invokeLater*(window);  
 window.addFigure();  
 setVisible(false);  
 dispose();  
 }  
 });  
  
 leaderboardButton = new JButton("HIGH SCORE");  
 leaderboardButton.setPreferredSize(new Dimension(300,100));  
 leaderboardButton.setBackground(new Color(255, 255, 255, 0));  
 leaderboardButton.setAlignmentX(JComponent.*CENTER\_ALIGNMENT*);  
  
  
  
 aboutButton = new JButton("ABOUT");  
 aboutButton.setPreferredSize(new Dimension(300,100));  
 aboutButton.setBackground(new Color(255, 255, 255, 0));  
 aboutButton.setAlignmentX(JComponent.*CENTER\_ALIGNMENT*);  
  
 aboutButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent arg0) {  
 GameAbout about = new GameAbout();  
 about.setVisible(true);  
 javax.swing.SwingUtilities.*invokeLater*(about);  
 setVisible(false);  
 dispose();  
 }  
 });  
  
 exitButton = new JButton("EXIT");  
 exitButton.setPreferredSize(new Dimension(300,100));  
 exitButton.setBackground(new Color(255, 255, 255, 0));  
 exitButton.setAlignmentX(JComponent.*CENTER\_ALIGNMENT*);  
  
 exitButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent arg0) {  
 setVisible(false);  
 System.*exit*(0);  
 }  
 });  
  
 //Добавляем кнопку на панель  
 mainPanel.add(mainJLabel);  
 mainPanel.add(startGameButton);  
 mainPanel.add(leaderboardButton);  
 mainPanel.add(aboutButton);  
 mainPanel.add(exitButton);  
 mainPanel.setBorder(BorderFactory.*createEmptyBorder*(50, 0, 0, 0));  
 //startGameButton.setBorder(BorderFactory.createEmptyBorder(50, 0, 50, 0));  
  
  
 //Устанавливаем размеры главного окна  
 setPreferredSize(new Dimension(350, 600));  
 setSize(350,600);  
 //Дабавляем панель на главное окно  
 add(mainPanel);  
  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 //Делаем окно по центру  
 setLocationRelativeTo(null);  
 //Делаем видимым главное окно  
 setVisible(true);  
  
 }  
  
 @Override  
 public void run() {  
 repaint();  
 }  
}

import model.Coordinate;  
import model.Figure;  
import userInterface.GameWindow;  
import userInterface.StartForm;  
  
public class Tetris {  
 public static void main(String[] args) {  
 StartForm startForm = new StartForm();  
 javax.swing.SwingUtilities.*invokeLater*(startForm);  
  
 //GameWindow window = new GameWindow();  
 //javax.swing.SwingUtilities.invokeLater(window);  
 //window.addFigure();  
 }  
}