**DOKUZ EYLUL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME1252 PROJECT BASED LEARNING – II**

**FINAL REPORT**

**PROJECT – III**

**COLUMNS**

**by**

**Fatma Ceren Akyüz**

**Melih Ekizce**

**Mustafa Eren Isıktaslı**

**Eren Çiçek**

**Lecturers**

**Assoc. Prof. Derya Birant**

**Dr. Göksu Tüysüzoğlu**

**IZMIR**

**27.05.2022**

# CHAPTER ONE

PROGRESS DESCRIPTION

The game is played in 5 columns. Game elements are numbers (1-10). The aim of the game is to reach the highest score by collecting number sets. Under certain transfer conditions, it is tried to reach the goal by moving elements from columns to columns or by taking elements from the box. In each set situation, the player's score increases. The game is over when there are no elements left in the columns or if the player collects 5 sets.

In the first week of the project, the necessary data structures and classes were created and file operations were completed. In the second week, the box implementation was completed and a separate Single Linked List was created for each column, and functions were written on them. In the third week of the project, there was no progress due to the exams. In the fourth week, when the majority of the project was completed, the functions were updated via the Multi-Linked List structure. The tasks of the 'B', 'X', 'Z', and direction keys were completed properly. The necessary functions for the transfer conditions were created. In the last week, endgame checks, some additional improvements, and debugging processes were made. The project was successfully completed.

# CHAPTER TWO

TASK SUMMARY

## Completed Tasks

Fatma Ceren Akyüz : Creating a shuffled box. Designing double linked list, and some other classes. File operations. High score table with new player. End game conditions. Final score calculation.

Melih Ekizce : Some improvements have been made on the appearance of the game. A GIF has been prepared, which explains the gameplay of the game and appears in the Enigma Console. Sound has been added to the menu part of the game and the card handling part.Helped conditions in the intercolumn transfer part.

Mustafa Eren Isıktaslı : Designing the game screen and adding numbers randomly from box to column has been made. The column part has been made. Choosing with the ‘Z’ part has made. Deleting the choosing part from ‘from column’ and adding the choosing part to the ‘to Column’.Some adjustments have been made on controlling if a column had completed.

Eren Çiçek : Creating the Multi-Linked List class, printing the screen; Necessary controls for 'B', 'X' and arrow keys, transfer conditions

## Incomplete Tasks: Reasons and Explanations

The project has been completed in full

## Additional Improvements ~~to the Project~~

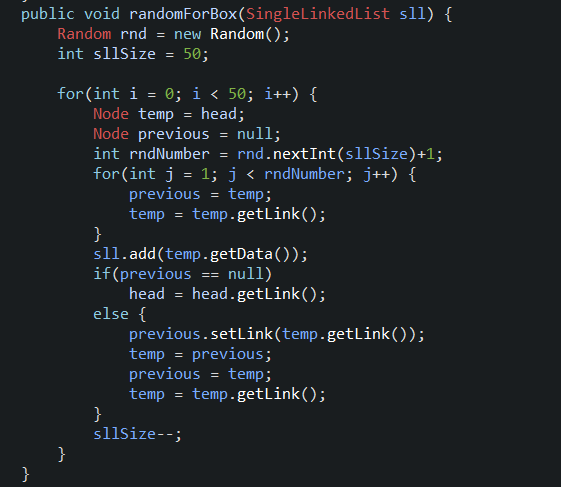
Menu and login screen has been added.

Sound effects has been added.

# CHAPTER THREE

EXPLANATION of algorıthms

## Screenshots



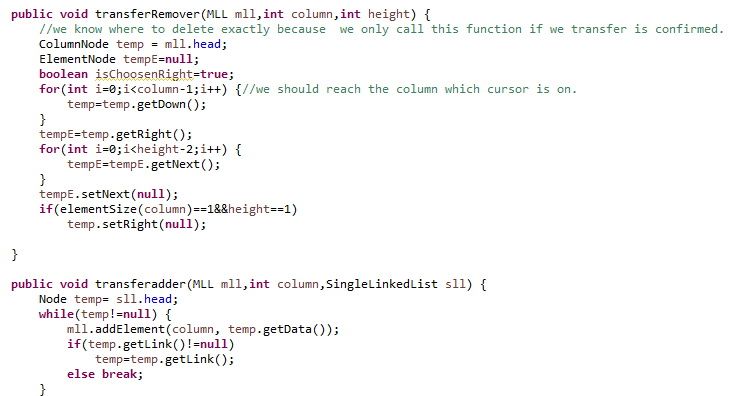
(Figure 1: randomForBox function)



(Figure 2: highScore function)



(Figure 3: isTransfer function)



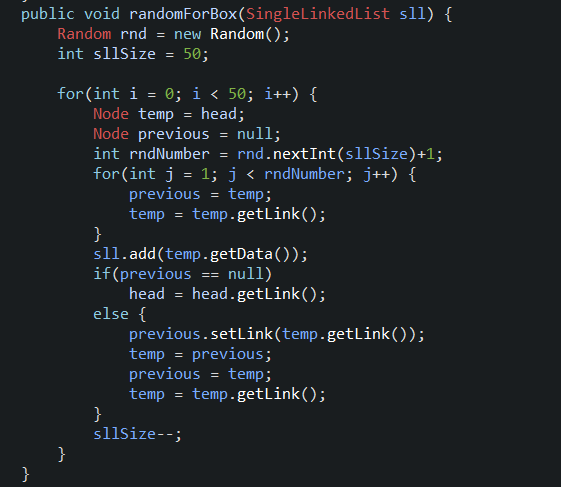
(Figure 4: transferRemover and transferAdder functions)



(Figure 5: isSet10 function)

## Algorithms and Solution Strategies

Fatma Ceren Akyüz : To shuffled the box, an ordered single linked list box is taken within the randomForBox function. In a loop, elements are skipped up to a randomly selected number between 1 and box size, the element in head is thrown into the mixedBox and deleted from the sorted box. The selected number is reduced by one at a time so that it does not exceed the size of the box. The loop continues 50 times.



(Figure 6: randomForBox function)

To read the high score table from the file, within the highScore function, in a loop as many as the number of lines in the file, each line is first thrown into the two-element highScore array. It is then assigned to the score and name of a variable belonging to the contestant class with its score and name. The contestant variable is added to the high score in the double linked list structure. The new player is also added to the table in the finalHighScore function.



(Figure 7: highScore function)

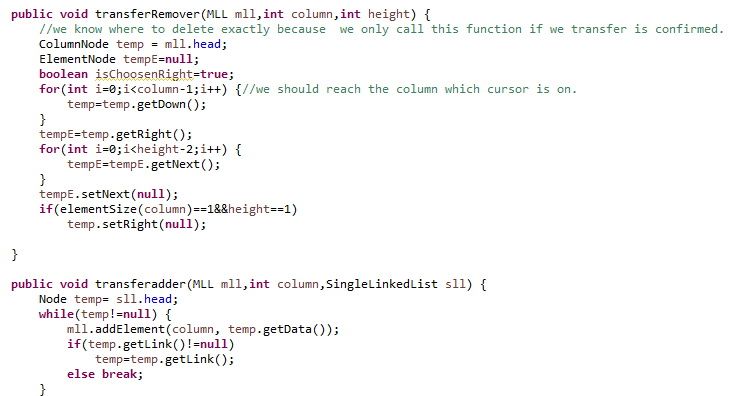
Melih Ekizce :

The algorithm in this section is set up for the accuracy of the transfer condition. The isTransfer function takes column and height parameters, which includes the column to move the elements to, and checks whether the difference between the moved element and the column we will move is 1 or 0. If the column is empty, it pays attention to whether the element to be moved is 1 or 10. Also, the cursor is more In order to optimize, we fill this column with a \* sign when it comes to an empty column, and we check this situation in the first condition of the while loop so that our program does not break because of this sign.



(Figure 8: isTransfer function)

Mustafa Eren Isıktaslı: In column Transfer part SingleLinkedList data Type had been used to keep the chosen part. After a part has chosen if all conditions are provided, Two function works first one goes around our MultiLinkedList and reaches the chosen part by using column and height information and disconnects the chosen part from the SLL and the second function(adder) is also work same but when it reaches the right coordinates it just connects our SLL to MLL’s following column and all transfer operations complete.



(Figure 9: transferRemover and transferAdder functions)

Eren Çiçek : In the function of 10-set control, the columns are checked one by one. In the elements on the controlled column, the difference between 2 elements in the column is taken and it is done for all elements in the column. If the difference is constantly 1 or -1, the set is provided. The function returns which column contains the set.



(Figure 10: isSet10 function)

# CHAPTER FOUR

PROBLEMS ENCOUNTERED

Fatma Ceren Akyüz : Some cast errors occured. But I solved this errors by using IntegerParseInt method. Also sometimes I forgot to use the getData() function and I tried to use Node variable directly. Because of that I take some @Node32342 error. I solved this problem by taking the temp.getData().

Melih Ekizce : The biggest problem I faced was that I misunderstood the homework document given to us, so I had to change the functions I wrote. Also, I encountered very minor roughnesses when adding sound.

Mustafa Eren Isıktaslı: Actually the biggest problem was misunderstandings about the game. Just because the Project is so similar with solitaire I thought the part that will be transferred should be a sequential series but when all parts are done we have seen that any part can be transferred as long as you can find a place to put it. So in this Project, twas were a lot of misunderstandings so we had worked more because of these problems but in the end, all Project have been completed

Eren Çiçek : A sign or a colored icon should have been shown to the player to make it clear that the cursor was there when the empty columns were accessed in the cursor movement. as a solution to this, the "\*" element was added to this column on the MultiLinkedList every time the player goes to an empty column. And when the player left that column, the "\*" element was deleted. But the deleted element "\*" in the game loop was still appearing on the screen. The problem was fixed by deleting the screen after each horizontal movement

# CHAPTER FIVE

conclusıon

Together with the task sharing and cooperation, the goals of the project were completed every week and successfully overcome.

REFERENCES

<https://stackoverflow.com/questions/43991004/playing-audio-file-in-eclipse> for adding sound

<https://patorjk.com/> for ascii characters

**AppendIx A**

Poster/Web page of the Project

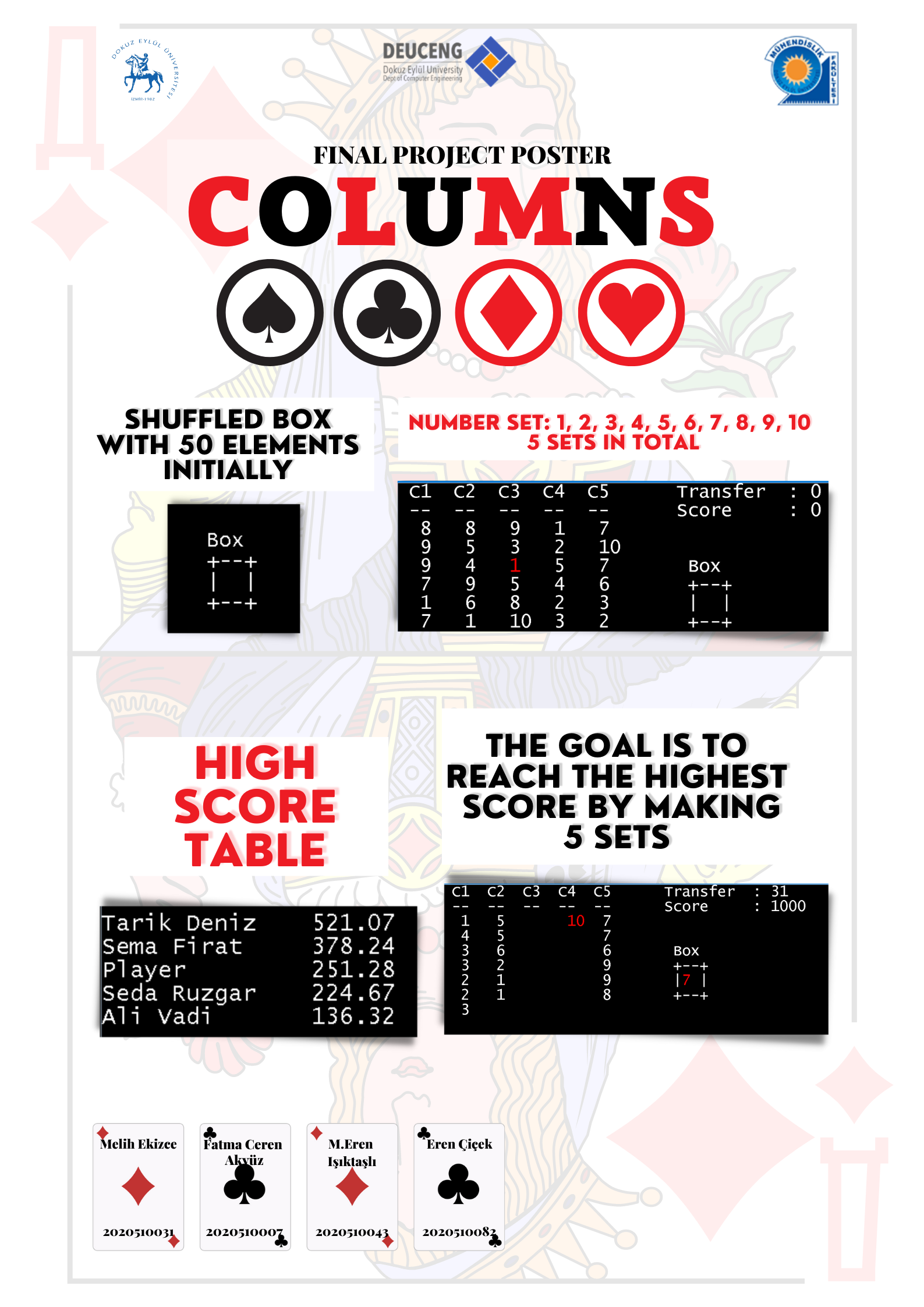


Figure A-1 Example poster**.**

**AppendIx B**

Code of the Project

import java.awt.Color;

import enigma.console.TextAttributes;

import enigma.core.Enigma;

public class AdditionalThings {

public static enigma.console.Console cn = Enigma.getConsole("COLUMNS",50,15,40,7);

public static void LoginScreen() throws InterruptedException{

cn.getTextWindow().setCursorPosition(0, 0);

int i=0;

while(i<5) {

TextAttributes color = new TextAttributes(Color.WHITE,Color.BLACK);

if(i==0) color = new TextAttributes(Color.RED);

else if(i==1) color=new TextAttributes(Color.MAGENTA);

else if(i==2) color=new TextAttributes(Color.PINK);

else if(i==3) color=new TextAttributes(Color.WHITE);

else if(i==4) color=new TextAttributes(Color.YELLOW);

cn.setTextAttributes(color);

System.out.print(" \_\_\_\_\_ \_\_\_\_ \_ \_ \_ \_\_ \_\_ \_ \_ \_\_\_\_\_ \r\n"

+ " / \_\_\_\_/ \_\_ \\| | | | | | \\/ | \\ | |/ \_\_\_\_|\r\n"

+ " | | | | | | | | | | | \\ / | \\| | (\_\_\_ \r\n"

+ " | | | | | | | | | | | |\\/| | . ` |\\\_\_\_ \\ \r\n"

+ " | |\_\_\_| |\_\_| | |\_\_\_| |\_\_| | | | | |\\ |\_\_\_\_) |\r\n"

+ " \\\_\_\_\_\_\\\_\_\_\_/|\_\_\_\_\_\_\\\_\_\_\_/|\_| |\_|\_| \\\_|\_\_\_\_\_/ \r\n"

+ " \r\n"

+ " ");

i++;

Thread.sleep(1000);

Column.clearScreen();

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

}

Column.clearScreen();

}

public static void HowToPlayGIF() throws InterruptedException {

cn.getTextWindow().setCursorPosition(0, 0);

TextAttributes color = new TextAttributes(Color.WHITE,Color.BLACK);

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print("Box");

color = new TextAttributes(Color.WHITE);

cn.getTextWindow().setCursorPosition(0, 1);

System.out.println("+--+");

cn.getTextWindow().setCursorPosition(0, 2);

System.out.println("|");

cn.getTextWindow().setCursorPosition(3, 2);

System.out.println("|");

cn.getTextWindow().setCursorPosition(0, 3);

System.out.println("+--+");

cn.setTextAttributes(color);

System.out.println("Game box containing these playing cards");

Thread.sleep(1500);

Column.clearScreen();

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.println("All Cards");

System.out.println("1 2 3 4 5 6 7 8 9 10 ");

System.out.println("1 2 3 4 5 6 7 8 9 10 ");

System.out.println("1 2 3 4 5 6 7 8 9 10 ");

System.out.println("1 2 3 4 5 6 7 8 9 10 ");

System.out.println("1 2 3 4 5 6 7 8 9 10 ");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

System.out.println("This game box contains 5 each of the numbers 1to10\nAlso, these cards are mixed in the box.\nMixed cards in the box ");

color = new TextAttributes(Color.GREEN);

cn.setTextAttributes(color);

System.out.println("1 3 2 6 8 4 9 10 7 5 ");

System.out.println("3 10 8 1 4 2 6 9 5 7 ");

System.out.println("8 4 2 5 1 7 9 10 6 3 ");

System.out.println("10 1 5 2 4 8 3 7 6 9 ");

System.out.println("3 4 10 1 2 8 6 7 9 2");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

Thread.sleep(2500);

Column.clearScreen();

cn.getTextWindow().setCursorPosition(0, 0);

//System.out.println("These cards are then randomly distributed to the columns in the game, 6 in each column.");

//System.out.println();

System.out.println("C1 C2 C3 C4 C5 ");

System.out.println("-- -- -- -- -- ");

System.out.print("1 3 2 6 8\n4 9 10 7 5 \n3 10 ");

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print("8 ");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

System.out.print(" 1 4 \n2 6 9 5 7 \n8 4 2 5 1 ");

System.out.println();System.out.println();System.out.println();

System.out.println("Competitor can direct cursor by pressing W-A-S-D keys");

Thread.sleep(2500);

Column.clearScreen();

System.out.println("C1 C2 C3 C4 C5 ");

System.out.println("-- -- -- -- -- ");

System.out.print("1 3 2 6 8\n4 9 10 7 5 \n3 10 8 ");

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print("1 ");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

System.out.print(" 4 \n2 6 9 5 7 \n8 4 2 5 1 ");

cn.getTextWindow().setCursorPosition(0, 0);

Thread.sleep(3500);

Column.clearScreen();

cn.getTextWindow().setCursorPosition(0, 0);

System.out.println("C1 C2 C3 C4 C5 ");

System.out.println("-- -- -- -- -- ");

System.out.print("1 3 2 6 8\n4 9 10 ");

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print("7 ");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

System.out.print(" 5 \n3 10 8 1 4 \n2 6 9 5 7 \n8 4 2 5 1 ");

System.out.println();System.out.println();System.out.println();

System.out.println("Competitor can direct cursor by pressing W-A-S-D keys");

Thread.sleep(3000);

Column.clearScreen();

System.out.println("C1 C2 C3 C4 C5 ");

System.out.println("-- -- -- -- -- ");

System.out.println("1 3 2 6 8\n4 9 10 7 5 \n3 10 8 1 4 \n2 6 9 5 7 \n8 4 2 5 1 ");

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

cn.getTextWindow().setCursorPosition(28, 3);

System.out.print("Box");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

cn.getTextWindow().setCursorPosition(28, 4);

System.out.println("+--+");

cn.getTextWindow().setCursorPosition(28, 5);

System.out.println("|");

cn.getTextWindow().setCursorPosition(31, 5);

System.out.println("|");

cn.getTextWindow().setCursorPosition(28, 6);

System.out.println("+--+");

System.out.println("After the competitor presses the 'B' key, a card begins to appear from the box.");

Thread.sleep(3500);

Column.clearScreen();

System.out.println("C1 C2 C3 C4 C5 ");

System.out.println("-- -- -- -- -- ");

System.out.println("1 3 2 6 8\n4 9 10 7 5 \n3 10 8 1 4 \n2 6 9 5 7 \n8 4 2 5 1 ");

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

cn.getTextWindow().setCursorPosition(28, 3);

System.out.print("Box");

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

cn.getTextWindow().setCursorPosition(28, 4);

System.out.println("+--+");

cn.getTextWindow().setCursorPosition(28, 5);

System.out.println("|");

cn.getTextWindow().setCursorPosition(31, 5);

System.out.println("|");

cn.getTextWindow().setCursorPosition(28, 6);

System.out.println("+--+");

cn.getTextWindow().setCursorPosition(29, 5);

System.out.println("7");

}

}

------------Box Class

public class Box {

private boolean visible;

public Box(boolean visible) {

this.visible = false;

}

public void mixingBox(SingleLinkedList mixedBox) {

SingleLinkedList box = new SingleLinkedList();

for(int i = 0; i < 5; i++) {

for(int j = 1; j <= 10; j++) {

box.add(j);

}

}

box.randomForBox(mixedBox);

}

public boolean isVisible() {

return visible;

}

public void setVisible(boolean visible) {

this.visible = visible;

}

}

---------------------------Column Class

import java.awt.Color;

import enigma.console.TextAttributes;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import enigma.core.Enigma;

import java.util.\*;

import javax.sound.sampled.AudioInputStream;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import javax.sound.sampled.LineUnavailableException;

import javax.sound.sampled.UnsupportedAudioFileException;

public class Column {

public static enigma.console.Console cn = Enigma.getConsole("Game", 100, 40, 20, 2);

//public static KeyListener klis;

public static int keypr; // key pressed?

public static int rkey; // key (for press/release)

KeyListener klis = new KeyListener() {

public void keyTyped(KeyEvent e) {

}

public void keyPressed(KeyEvent e) {

if (keypr == 0) {

keypr = 1;

rkey = e.getKeyCode();

}

}

public void keyReleased(KeyEvent e) {

}

}; // do not touch

static int score = 0;

static int transfer =0;

public static int column = 1;

public static int height = 1;

public static int orderedCount=0;

public void gamePlay() throws IOException, InterruptedException, LineUnavailableException, UnsupportedAudioFileException {//This function is our main game function.

boolean isPressedZ=false;//This boolean will be true as long as we keep a choosen list.

SingleLinkedList transferCarrier= new SingleLinkedList();// Carries the list which will be transfered.

cn.getTextWindow().addKeyListener(klis);

TextAttributes color = new TextAttributes(Color.WHITE,Color.BLACK);

//int column = 1;//column number

//int height = 1;//height number

int transferedColumn=1; // for 'Z' key

int transferedHeight=1;

enigma.console.Console cn = Enigma.getConsole("Game", 100, 40, 20, 2);

Box box = new Box(false);

DoubleLinkedList HighScoreTable = new DoubleLinkedList();

SingleLinkedList mixedBox = new SingleLinkedList();

box.mixingBox(mixedBox);

MLL mll = new MLL();

mll.addColumn(1);mll.addColumn(2);mll.addColumn(3);mll.addColumn(4);mll.addColumn(5);

firstGameScreen(mixedBox,mll);//Prints the first times game Screen

boolean isSelected = false; // This boolean will be used for box.

boolean exitFlag = false;

while(!mll.endGame() && !exitFlag) {

if(box.isVisible()&&!isSelected) {//To show box element's condition correctly

cn.getTextWindow().setCursorPosition(27, 6);

System.out.print(mixedBox.headElement());

}

else if(box.isVisible()&&isSelected) {//To show box element's condition correctly

cn.getTextWindow().setCursorPosition(27, 6);

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print(mixedBox.headElement());

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

}

printGameLines(); //

mll.display(column,height);

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

if(keypr == 1) {

switch(rkey) {//Cursor Movements and other key probability

case KeyEvent.VK\_LEFT: //Moving Left

mll.horizontalMove(0);

clearScreen();

break;

case KeyEvent.VK\_RIGHT: //Moving Right

mll.horizontalMove(1);

clearScreen();

break;

case KeyEvent.VK\_UP: //Moving Up

if(height !=1 ) {

height--;

}

else {

height = mll.elementSize(column);//If there are no elements on the cursor cursor goes end of the column

}

break;

case KeyEvent.VK\_DOWN: //Moving down

if(height != mll.elementSize(column)) {

height++;

}

else {

height =1;//If there are no elements below the cursor, it goes to first element

}

break;

case KeyEvent.VK\_B:

if(mixedBox.headElement()!=null) {

if(!box.isVisible()) {//After pressing 'B' if box was not visiable condition

cn.getTextWindow().setCursorPosition(27, 6);

System.out.print(mixedBox.headElement());

box.setVisible(true);

}

else if(box.isVisible()&& isSelected == false){//If box is visible and user pressed B one more time Chooses the box element

isPressedZ=false;

isSelected = true;

cn.getTextWindow().setCursorPosition(27, 6);

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print(mixedBox.headElement());

}

else if(isSelected == true){ // To set box element as not choosed

isSelected = false;

cn.getTextWindow().setCursorPosition(27, 6);

System.out.print(mixedBox.headElement());

}

}

else {

cn.getTextWindow().setCursorPosition(36, 10);

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.println("There is no element in Box!");

}

break;

case KeyEvent.VK\_X:

if(box.isVisible()&&isSelected==true) { // transfer for the box element

int[] columnNumbersToTransfer = mll.transferFromBox(mixedBox.headElement());

for(int i = 0;i<columnNumbersToTransfer.length;i++){

if(columnNumbersToTransfer[i]==column) {

mll.addElement(column, mixedBox.headElement());

mixedBox.deleteFirstElement();

cn.getTextWindow().setCursorPosition(27, 6);

System.out.print(" ");

isSelected = false;

box.setVisible(false);

transfer++;

break;

}

}

int set10column = mll.isSet10();//This function controlls if there are any 10 set

if(set10column!=0) {

score+=1000;

orderedCount++;

//Thread.sleep(3000);

mll.display(column, height);

Thread.sleep(1500); //A delay for 1,5 seconds to show the user transfer operation

mll.deleteElements(set10column);

}

}

if(isPressedZ&&!isSelected) { // So if we have a choosen part from the 'From column' and if box element was not choosen

if(transferCarrier!=null) {

if(mll.isTransfer(column, height, transferCarrier)) {

Clip clip = AudioSystem.getClip();

AudioInputStream cardsoundeffect = AudioSystem.getAudioInputStream(new File("cardsound.wav"));

clip.open(cardsoundeffect);

clip.start();

mll.transferRemover(mll, transferedColumn, transferedHeight);//deletes the transfered elements from the first box

mll.transferadder(mll, column,transferCarrier); //adds the choosen part to the 'To column'

int set10column = mll.isSet10();

if(set10column!=0) {

score+=1000;

orderedCount++;

//Thread.sleep(3000);

mll.display(column, height);

Thread.sleep(1500);

mll.deleteElements(set10column);

}

transfer++;

transferCarrier=null;

isPressedZ=false;

clearScreen();//Clears the screen

cn.getTextWindow().setCursorPosition(33,5);

System.out.println("Z key: OFF");

//transfer completed

}

else {//Wrong column choosen Transfer Failed!

cn.getTextWindow().setCursorPosition(25, 25);

System.out.println("Please Choose a right place to apply the transfer");

}

}

else {//User did not choose a part which can be transfered.

cn.getTextWindow().setCursorPosition(25, 25);

System.out.println("Please choose the list first!");

}

}

break;

case KeyEvent.VK\_Z:

transferCarrier= mll.transferPartReturner(column, height);//This function Returns the choosen part in a Single Linked List

transferedColumn=column;//this integer keeps the first column coordinate(from columns)

transferedHeight=height;//this integer keeps the first height coordinate

if(transferCarrier!=null) {//If there is a choosen part

cn.getTextWindow().setCursorPosition(33, 5);

System.out.println("Z key: ON ");

isPressedZ=true;

}

else {

isPressedZ = false;

}

break;

case KeyEvent.VK\_E://To exit the game

exitFlag = true;

break;

}

}

keypr= 0;

rkey=0;

}

if(exitFlag== false) {

cn.getTextWindow().setCursorPosition(0, 30);

finalHighScore(score); //printing final high score

}

else {

cn.getTextWindow().setCursorPosition(0, 30);

highScore(HighScoreTable);

HighScoreTable.display();

}

}

public void menu() throws Exception {//A standard menu part for the game, which includes Game Info, game and high score table

Scanner scann = new Scanner(System.in);

String selection = "1"; // menu selection

DoubleLinkedList HighScoreTable = new DoubleLinkedList();

while (!selection.equals("2")){

System.out.println(); System.out.println();

System.out.println("\t1.Game Info\n" +

"\t2.Start Game\n" +

"\t3.High Score Table\n" +

"\t4.Exit\n");

System.out.print("Enter your choice : ");

while(true) {

Clip clip = AudioSystem.getClip();

AudioInputStream menuAudio = AudioSystem.getAudioInputStream(new File("menusound.wav"));

clip.close();

clip.open(menuAudio);

clip.start();

selection=scann.next();

if(selection.equals("1") || selection.equals("2") || selection.equals("3") || selection.equals("4")) {

clip.close();

break;

}System.out.println();

System.out.print("\tPlease enter a number between 1-4 : ");

}

switch(selection){

case "1" :clearScreen();

AdditionalThings.HowToPlayGIF();

clearScreen();

break;

case "2" : clearScreen();gamePlay(); break;//Main game

case "3" :clearScreen();highScore(HighScoreTable); //Score table

HighScoreTable.display();

break;

case "4" :clearScreen(); //End of the game

System.out.println("Thanks for playing! Bye!"); System.exit(1);

default: break;

}

}

}

public static void clearScreen() {

// output spaces to clear the screen

// start with one space less than the screen size, so as not to induce scrolling

char[] buffer = new char[Math.max(0, cn.getTextWindow().getColumns() \* cn.getTextWindow().getRows() - 1)];

Arrays.fill(buffer, ' ');

cn.getTextWindow().setCursorPosition(0, 0);

cn.getTextWindow().output(buffer, 0, buffer.length);

// this positional output does not cause scrolling

cn.getTextWindow().output(cn.getTextWindow().getColumns() - 1, cn.getTextWindow().getRows() - 1, ' ');

// move cursor back to beginning

cn.getTextWindow().setCursorPosition(0, 0);

}

public void firstGameScreen(SingleLinkedList mixedBox,MLL mll) { //distributes elements from the box into columns

Node temp = mixedBox.head;

int columnNum = 1;

for(int i =0;i<5;i++){

for(int j=0;j<6;j++) {

mll.addElement(columnNum, temp.getData());//adds the element to the MLL

mixedBox.deleteFirstElement();//deletes the element from the mixedBox

temp = temp.getLink();

}

columnNum++;

}

}

public void printGameLines() { // prints the constant game lines

cn.getTextWindow().setCursorPosition(1,0);

System.out.println("C1 C2 C3 C4 C5");

cn.getTextWindow().setCursorPosition(1,1);

System.out.println("-- -- -- -- --");

cn.getTextWindow().setCursorPosition(25, 0);

System.out.println("Transfer : " + transfer);

cn.getTextWindow().setCursorPosition(25, 1);

System.out.println("Score : " + score);

cn.getTextWindow().setCursorPosition(26, 4);

System.out.print("Box");

cn.getTextWindow().setCursorPosition(26, 5);

System.out.println("+--+");

cn.getTextWindow().setCursorPosition(26, 6);

System.out.println("|");

cn.getTextWindow().setCursorPosition(29, 6);

System.out.println("|");

cn.getTextWindow().setCursorPosition(26, 7);

System.out.println("+--+");

}

public void highScore(DoubleLinkedList DLL) throws IOException {

//enigma.console.Console cn = Enigma.getConsole("COLUMNS",50,10,40,7);

BufferedReader objReader = new BufferedReader(new FileReader("highscore.txt"));

String row;

Object[] highScore = new Object[2];

row = objReader.readLine();

while (row != null) {

if(!row.equals("")) {

//adds each row to the array

highScore = row.split(" ");

}

//sets the cont's name and score from the highScore array

Contestant cont = new Contestant((String) highScore[0], Double.parseDouble((String)highScore[1]));

DLL.add(cont);

row = objReader.readLine();

}

objReader.close();

}

//adds the new player to the high score table

public void finalHighScore(int score) throws IOException {

DoubleLinkedList HighScoreTable = new DoubleLinkedList();

Column highScore = new Column();

highScore.highScore(HighScoreTable);

Contestant player = new Contestant("Player", calculateScore());

HighScoreTable.add(player);

HighScoreTable.display();

}

public double calculateScore() {

double endGameScore = (100\*orderedCount + ((double)score / (double)transfer));

endGameScore = (double) (Math.round(endGameScore\*100.0)/100.0);

return endGameScore;

}

}

-------ColumnNode Class

public class ColumnNode {

private Object data;

private ColumnNode down;

private ElementNode right;

public ColumnNode(Object data) {

this.data=data;

down = null;

right = null;

}

public Object getData() {

return data;

}

public void setData(Object data) {

this.data = data;

}

public ColumnNode getDown() {

return down;

}

public void setDown(ColumnNode down) {

this.down = down;

}

public ElementNode getRight() {

return right;

}

public void setRight(ElementNode right) {

this.right = right;

}

}

---------ColumnTest Class

import java.io.IOException;

public class ColumnTest {

public static void main(String[] args) throws Exception{

AdditionalThings.LoginScreen();

Column game = new Column();

game.menu();

}

------Contestant Class

}

public class Contestant {

private String name;

private double score;

public Contestant(String name, double score) {

this.name = name;

this.score = score;

}

public String getName() {

return name;

}

public double getScore() {

return score;

}

}

------DoubleLinkedList Class

public class DoubleLinkedList {

private DoubleNode head;

private DoubleNode tail;

public DoubleLinkedList() {

head = null;

tail = null;

}

public void addToTheEnd(Object dataToAdd) {

if(head == null && tail == null) {

DoubleNode newNode = new DoubleNode(dataToAdd);

head = newNode;

tail = newNode;

}

else {

DoubleNode newNode = new DoubleNode(dataToAdd);

newNode.setPrev(tail);

tail.setNext(newNode);

tail = newNode;

}

}

public void add(Contestant cont) {

if(head == null && tail == null) {

DoubleNode newNode = new DoubleNode(cont);

head = newNode;

tail = newNode;

}

else {

DoubleNode newNode = new DoubleNode(cont);

if(cont.getScore() > ((Contestant) head.getData()).getScore()) {

newNode.setNext(head);

head.setPrev(newNode);

head = newNode;

}

else {

DoubleNode temp = head;

while(temp.getNext() != null && cont.getScore() < ((Contestant)temp.getNext().getData()).getScore()) {

temp = temp.getNext();

}

newNode.setPrev(temp);

newNode.setNext(temp.getNext());

if(temp.getNext() != null)

temp.getNext().setPrev(newNode);

else

tail = newNode;

temp.setNext(newNode);

}

}

}

public int size(){

int count = 0;

DoubleNode temp = head;

while(temp != null){

count++;

temp = temp.getNext();

}

return count;

}

public void display() {

if(head == null)

System.out.println("List is empty!");

else {

DoubleNode temp = head;

String space = " ";

while(temp != null) {

int spacenum = 15 - ((Contestant)temp.getData()).getName().length();

System.out.print(((Contestant)temp.getData()).getName() + space.repeat(spacenum) + ((Contestant)temp.getData()).getScore() + "\n");

temp = temp.getNext();

}

}

}

}

-------------DoubleNode Classs

public class DoubleNode {

private Object data;

private DoubleNode prev;

private DoubleNode next;

public DoubleNode(Object dataToAdd) {

data = dataToAdd;

prev = null;

next = null;

}

public Object getData() {

return data;

}

public void setData(Object data) {

this.data = data;

}

public DoubleNode getPrev() {

return prev;

}

public void setPrev(DoubleNode prev) {

this.prev = prev;

}

public DoubleNode getNext() {

return next;

}

public void setNext(DoubleNode next) {

this.next = next;

}

}

---------ElementNode Class

public class ElementNode {

private Object data;

private ElementNode next;

public ElementNode(Object data) {

this.data = data;

next = null;

}

public Object getData() {

return data;

}

public void setData(Object data) {

this.data = data;

}

public ElementNode getNext() {

return next;

}

public void setNext(ElementNode next) {

this.next = next;

}

}

---------------------Mll Class

import java.awt.Color;

import enigma.console.TextAttributes;

import enigma.core.Enigma;

public class MLL {

private ColumnNode head;

//TextAttributes color = new TextAttributes(Color.WHITE,Color.BLACK);

public MLL() {

head = null;

}

public void addColumn(int columnNumber) { // adds a new column

ColumnNode newNode = new ColumnNode(columnNumber);

if(head == null) {

head = newNode;

}

else{

ColumnNode temp = head;

while(temp.getDown()!=null) {

temp = temp.getDown();

}

temp.setDown(newNode);

}

}

public void addElement(int columnNumber , Object element) {

// Adds an element to the column entered as a parameter

if(head == null) {

System.out.println("There is no column");

}

else {

ElementNode newNode = new ElementNode(element);

ColumnNode temp = head;

while(temp!=null && columnNumber != (int)temp.getData()) {

temp = temp.getDown();

}

if(temp!=null) {

if(temp.getRight()==null) {

temp.setRight(newNode);

}

else {

ElementNode tempE = temp.getRight();

while(tempE.getNext()!=null) {

tempE = tempE.getNext();

}

tempE.setNext(newNode);

}

}

else {

System.out.println("The requested column does not exist");

}

}

}

public void display(int column,int height) { // prints columns and heights

enigma.console.Console cn = Enigma.getConsole("Game", 100, 40, 20, 2);

TextAttributes color = new TextAttributes(Color.WHITE,Color.BLACK);

if(head == null) {

System.out.println("there is no data to display");

}

else {

ColumnNode temp = head;

int cursorx =2;

while(temp!=null) {

if(temp.getRight()!=null) {

ElementNode tempE = temp.getRight();

int i=0;

int count = 1;

while(tempE!=null) {

if(tempE.getData()!=null) {

cn.getTextWindow().setCursorPosition(cursorx, i+2);

if((int)temp.getData()==column&&count==height) {

// to write the point we are on in red

color = new TextAttributes(Color.RED);

cn.setTextAttributes(color);

System.out.print(tempE.getData().toString());

color = new TextAttributes(Color.WHITE); // to revert to its original color

cn.setTextAttributes(color);

}

else {

color = new TextAttributes(Color.WHITE);

cn.setTextAttributes(color);

System.out.print(tempE.getData().toString());

}

i++;

count++;

tempE = tempE.getNext();

}

else

break;

}

}

cursorx+=4;

temp = temp.getDown();

}

}

}

public int columnSize() { // return the number of columns

if(head == null) {

System.out.println("there is no column");

return 0;

}

else {

ColumnNode temp = head;

int count = 0;

while(temp!=null) {

count++;

temp = temp.getDown();

}

return count;

}

}

public int elementSize(int columnNumber) { // returns the number of elements on column that entered parameter

if(head == null) {

System.out.println("there is no column");

return 0;

}

else {

int count = 0;

ColumnNode temp = head;

while(temp!=null && (int)temp.getData()!=columnNumber) {

temp = temp.getDown();

}

if(temp!=null) {

count =0;

ElementNode tempE = temp.getRight();

while(tempE!=null) {

count++;

tempE = tempE.getNext();

}

temp = temp.getDown();

}

else {

System.out.println("there is no column like : C"+ columnNumber);

}

return count;

}

}

public int[] transferFromBox(Object data) {

int[] columnNumberstoTransfer = new int[5];

columnNumberstoTransfer[0]=0;

if(head == null) {

System.out.println("there is no column");

return null;

}

else {

int a = 0;

ColumnNode temp = head;

int columnNum = 1;

while(temp!=null) {

if(temp.getRight()!=null && temp.getRight().getData().equals("\*")) {

deleteElements(columnNum); // If there is a "\*" element in the column we are in

// and we want to move an element from the box here, we first delete the "\*" element.

}

if(temp.getRight()==null&&((int)data == 10||(int)data == 1)) {

columnNumberstoTransfer[a] = (int)temp.getData(); // if column is empty and element from box is 1 or 10

a++;

}

else{

if(temp.getRight()!=null) {

ElementNode tempE= temp.getRight();

while(tempE.getNext()!=null) {

tempE = tempE.getNext();

}

int difference = (int)tempE.getData()-(int)data;

if(Math.abs(difference)==1||difference ==0){

columnNumberstoTransfer[a] = (int)temp.getData();

a++;

}

}

}

temp = temp.getDown();

columnNum++;

}

return columnNumberstoTransfer;

}

}

public void deleteElements(int column) { // Deletes all elements in the column entered as a parameter

if(head == null) {

System.out.println("There is no column");

}

else {

ColumnNode temp = head;

while(temp!=null) {

if((int)temp.getData()==column) {

temp.setRight(null);

break;

}

temp = temp.getDown();

}

}

}

public Object columnheadElement(int column) { // Returns the head element of the column

if(head == null) {

System.out.println("There is no column");

return null;

}

else {

Object firstelement = null;

ColumnNode temp = head;

while(temp!=null) {

if((int)temp.getData()==column&&temp.getRight()!=null) {

firstelement = temp.getRight().getData();

}

temp = temp.getDown();

}

return firstelement;

}

}

public void horizontalMove(int direction) { // 0-left , 1-right

enigma.console.Console cn = Enigma.getConsole("Game", 100, 40, 20, 2);

while(true) {

if(elementSize(Column.column)!=0&&columnheadElement(Column.column).equals("\*")) {

deleteElements(Column.column); // If there is a "\*" element in the column we are on and

} //we are going to make a horizontal movement, the "\*" element is deleted first

if(Column.column!=5 && direction == 1) {

if(elementSize(Column.column+1)>= Column.height) {

Column.column++;

break;

}

else if(elementSize(Column.column+1)==0){

Column.column++; // // If the column to go to is empty, the "\*" element is added to that column.

addElement(Column.column,"\*");

Column.height = elementSize(Column.column);

break;

}

else {

Column.height = elementSize(Column.column+1);

Column.column++;

break;

}

}

else if(Column.column == 5 && direction == 1) {

Column.column = 1;

if(elementSize(Column.column)!=0) {

if(elementSize(Column.column)<Column.height) {

Column.height = elementSize(Column.column);

}

break;

}

else {

addElement(Column.column,"\*");

Column.height = elementSize(Column.column);

break;

}

}

else if(Column.column!=1 && direction == 0) {

if(elementSize(Column.column-1)>= Column.height) {

Column.column--;

break;

}

else if(elementSize(Column.column-1)==0){

Column.column--;

addElement(Column.column,"\*"); // If the column to go to is empty, the "\*" element is added to that column.

Column.height = elementSize(Column.column);

break;

}

else {

Column.height = elementSize(Column.column-1);

Column.column--;

break;

}

}

else if(Column.column == 1 && direction == 0) {

Column.column = 5;

if(elementSize(Column.column)!=0) {

if(elementSize(Column.column)<Column.height) {

Column.height = elementSize(Column.column);

}

break;

}

else {

// If the column to go to is empty, the "\*" element is added to that column.

addElement(Column.column,"\*");

Column.height = elementSize(Column.column);

break;

}

}

}

}

public SingleLinkedList transferPartReturner(int column,int height) {

if(head == null) { // When z is pressed, it adds a single linked list the elements under it along with the element we are on.

System.out.println("There is no column");

return null;

}

else {

SingleLinkedList transferList=new SingleLinkedList();

ColumnNode temp = head;

while(temp!=null) {

if(column == (int)temp.getData()) {

if(temp.getRight()!=null) {

if(temp.getRight().getData().equals("\*")) {

transferList=null;

break;

}

ElementNode tempE = temp.getRight();

int count = 1;

while(tempE !=null) {

if(count == height) {

while(tempE!=null) {

transferList.add(tempE.getData());

tempE = tempE.getNext();

}

break;

}

count++;

tempE = tempE.getNext();

}

}

}

temp = temp.getDown();

}

return transferList;

}

}

public void transferRemover(MLL mll,int column,int height) {

//we know where to delete exactly because we only call this function if we transfer is confirmed.

ColumnNode temp = mll.head;

ElementNode tempE=null;

boolean isChoosenRight=true;

for(int i=0;i<column-1;i++) {//we should reach the column which cursor is on.

temp=temp.getDown();

}

tempE=temp.getRight();

for(int i=0;i<height-2;i++) {

tempE=tempE.getNext();

}

tempE.setNext(null);

if(elementSize(column)==1&&height==1)

temp.setRight(null);

}

public void transferadder(MLL mll,int column,SingleLinkedList sll) {

Node temp= sll.head; ////Adds the elements in the single linked list to the column to go to by pressing Z

while(temp!=null) {

mll.addElement(column, temp.getData());

if(temp.getLink()!=null)

temp=temp.getLink();

else break;

}

}

public boolean isTransfer(int column,int height,SingleLinkedList sll) { // transfer controls

if(head == null) {

System.out.println("There is no column");

return false;

}

else {

boolean flag = false;

ColumnNode temp = head;

while(temp!=null && (int)temp.getData()!= column) {

temp = temp.getDown();

}

if(temp.getRight()!=null && temp.getRight().getData().equals("\*")) {

deleteElements(column); // if there is a star in the column to go to

flag = false;

}

if(temp.getRight()==null) {

if((int)sll.headElement()==10 || (int)sll.headElement()==1) {

flag = true;

}

}

else{

ElementNode tempE = temp.getRight();

while(tempE.getNext()!=null) {

tempE = tempE.getNext();

}

int difference = (int)tempE.getData() - (int)sll.headElement();

if(Math.abs(difference)== 1 || difference == 0) {

flag = true;

}

}

return flag;

}

}

public int isSet10() { // it checks if there is a set of 10 by traversing all the columns

if(head == null) {

System.out.println("there is no column");

return 0;

}

else {

boolean flag = false;

ColumnNode temp = head;

int column\_num = 1;

int setColumnNum=0;

while(temp!=null) {

if(elementSize(column\_num)==10) {

ElementNode tempE = temp.getRight();

ElementNode previous = null;

int count = 1;

boolean flagfordecreasing = true;

boolean flagforincreasing = true;

while(tempE.getNext()!=null) {

previous = tempE;

tempE = tempE.getNext();

int difference = (int)previous.getData()-(int)tempE.getData();

if(difference==1 && flagforincreasing==true) {

flagfordecreasing = false;

count++;

}

else if(difference == -1 && flagfordecreasing == true) {

count++;

flagforincreasing = false;

}

if(count == 10) {

flag =true;

break;

}

}

}

if(flag==true) {

setColumnNum = column\_num;

break;

}

temp = temp.getDown();

column\_num++;

}

return setColumnNum;

}

}

public boolean endGame() { // the game is over if the elements in the columns run out

boolean flag = true;

for(int i=1;i<=5;i++) {

if(elementSize(i)!=0) {

flag=false;

}

}

return flag;

}

}

---------------NodeClass

public class Node {

Object data;

Node link;

public Node(Object dataToAdd) {

data = dataToAdd;

link = null;

}

public Object getData() { return data; }

public void setData(Object data) { this.data = data; }

public Node getLink() { return link; }

public void setLink(Node link) { this.link = link; }

}

---------------SingleLinkedList Class

import java.util.Random;

public class SingleLinkedList {

Node head;

//For adding dices

public void add(Object dataToAdd) {

if(head == null) {

Node newNode = new Node(dataToAdd);

head = newNode;

}

else {

Node temp = head;

while(temp.getLink() != null)

temp = temp.getLink();

Node newNode = new Node(dataToAdd);

temp.setLink(newNode);

}

}

public Object headElement() {

if(head != null)

return head.getData();

else

return null;

}

public void randomForBox(SingleLinkedList sll) {

Random rnd = new Random();

int sllSize = 50;

for(int i = 0; i < 50; i++) {

Node temp = head;

Node previous = null;

int rndNumber = rnd.nextInt(sllSize)+1;

for(int j = 1; j < rndNumber; j++) {

previous = temp;

temp = temp.getLink();

}

sll.add(temp.getData());

if(previous == null)

head = head.getLink();

else {

previous.setLink(temp.getLink());

temp = previous;

previous = temp;

temp = temp.getLink();

}

sllSize--;

}

}

public int size() {

if(head == null)

return 0;

else {

int count = 0;

Node temp = head;

while(temp != null) {

temp = temp.getLink();

count++;

}

return count;

}

}

public void display() {

if(head == null)

System.out.println("List is empty!");

else {

Node temp = head;

while(temp != null) {

System.out.print(temp.getData() + " ");

temp = temp.getLink();

}

}

}

public int count(Object data) {

if(head == null) {

System.out.println("List is empty!");

return 0;

}

else {

Node temp = head;

int count = 0;

while(temp != null) {

if((Integer) temp.getData() == (Integer) data)

count++;

temp = temp.getLink();

}

return count;

}

}

public boolean search(Object data) {

if(head == null) {

System.out.println("List is empty!");

return false;

}

else {

Node temp = head;

while(temp != null) {

if((Integer) temp.getData() == (Integer) data)

return true;

temp = temp.getLink();

}

return false;

}

}

public void remove(Object dataToDelete) {

if(head == null)

System.out.println("List is empty");

else {

int count = 0;

while((Integer) head.getData() == (Integer) dataToDelete) {

head = head.getLink();

count++;

}

Node temp = head;

Node previous = null;

while(temp != null) {

if((Integer) temp.getData() == (Integer) dataToDelete) {

previous.setLink(temp.getLink());

temp = previous;

count++;

}

previous = temp;

temp = temp.getLink();

if(count == 1)

break;

}

}

}

public void deleteFirstElement() {

if(head == null) {

System.out.println("list is empty");

}

else {

head = head.getLink();

}

}

public void deleteLastElement() {

if(head == null)

System.out.println("List is empty");

else {

Node temp = head;

Node previous = null;

while(temp.getLink() != null) {

previous = temp;

temp = temp.getLink();

}

previous.setLink(null);

}

}

public boolean isEmpty() {

if(head == null) {

return true;

}

else

return false;

}

}