北京邮电大学软件学院

实验报告

课程名称： Java SE 程 序 设 计

项目名称： Java程序编制（基础练习）

项目完成人：

姓名： 王衔飞

学号：

指导教师： 程保中 孙艺

日 期： 2019 年 10 月 13 日

（正文文字，请使用【正文缩进】样式）

（标题编号请使用自动编号。在样式表中选中标题级别样式即可自动编号。）

# 实验概述

## 实验目的

学生通过使用Java语言进行基本程序的开发，掌握Java通用IDE，练习类的封装使用、Java基本类库的使用、Java GUI编程和事件驱动编程、利用UML进行简单建模。

## 实验内容

|  |  |
| --- | --- |
|  | 任务说明 |
|  | 设计一个小车竞赛的GUI程序 |
|  | 设计一个计算不同人群的税率的控制台程序 |
|  | 设计一个寻找最小公倍数的控制台程序 |
|  |  |
|  | 设计一个24点牌游戏的GUI程序 |
|  |  |
|  |  |

## 实验环境

### 硬件环境

不限

### 网络环境

不需要

### 软件环境

JDK1.8+IDEA 2019

## 实验结果

完成实验

# 实验内容

# Project 1分析设计

## N.1问题分析

### N.1.1 目标

设计一个GUI程序实现小车比赛

### N.1.2功能

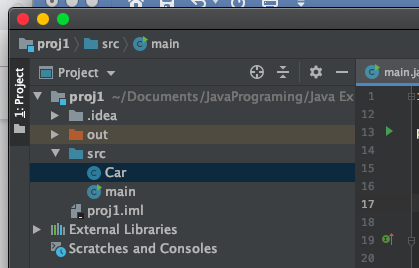
可以自定义四个小车的速度，并让这几个小车跑起来

### N.1.3性能

各项功能均无肉眼可见的等待时间

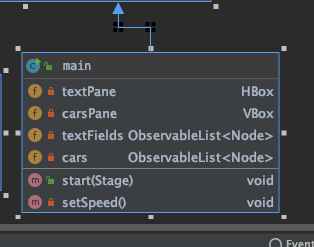
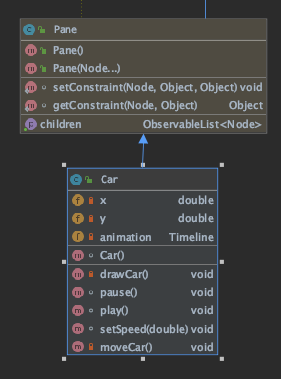
## N.2设计方案

### N.2.1模块划分



本项目包含两个模块，分别是Car和main

### N.2.2类图

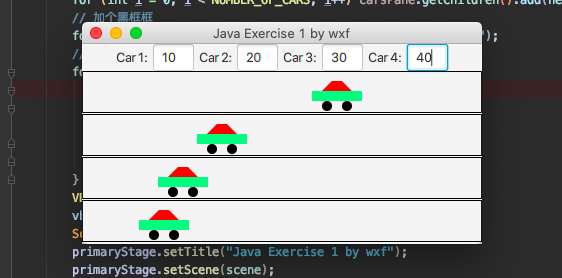
注：Car类继承Pane，Main类继承Application

### N.2.3算法

#### N.2.3.1算法一：让小车动起来：创建一个Timeline，关联moveCar函数，使用Timeline类中的setRate方法改变小车的运动速度，再使用取倒数的方法使得1位最快，越高越慢。

## N.3实验结果

试验完成，可以达到预期效果，输入相应的速度值按下回车后，截图如下。



## N.4 调试心得

没有遇到明显问题

## N.5源程序

/\* 文件名：main.java

\*/

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.geometry.Pos;

import javafx.collections.ObservableList;

import javafx.scene.Node;

import javafx.scene.input.KeyCode;

public class main extends Application {

private HBox textPane = new HBox(5);

private VBox carsPane = new VBox(1);

private ObservableList<Node> textFields = textPane.getChildren(); //包含文字及输入框，size=8

private ObservableList<Node> cars = carsPane.getChildren(); //包含所有车 size=4

@Override public void start(Stage primaryStage) {

final int NUMBER\_OF\_CARS = 4;

textPane.setAlignment(Pos.CENTER);

// 把标签和输入框放在pane里面

for (int i = 1; i <= NUMBER\_OF\_CARS; i++) textPane.getChildren().addAll(

new Label("Car " + i + ":"), new TextField());

// 设置输入框的大小

for (int i = 1; i < textFields.size(); i+= 2)

((TextField)textFields.get(i)).setPrefColumnCount(2);

// 把车放在pane里面

for (int i = 0; i < NUMBER\_OF\_CARS; i++) carsPane.getChildren().add(new Car());

// 加个黑框框

for (Node car: cars) car.setStyle("-fx-border-color: black");

// 设置监听键盘事件

for (int i = 1; i < textFields.size(); i += 2) {

(textFields.get(i)).setOnKeyPressed(e -> {

if (e.getCode() == KeyCode.ENTER) {

setSpeed();

}

});

}

VBox vbox = new VBox();

vbox.getChildren().addAll(textPane,carsPane);

Scene scene = new Scene(vbox, 400, 200);

primaryStage.setTitle("Java Exercise 1 by wxf");

primaryStage.setScene(scene);

primaryStage.show();

}

private void setSpeed() {

for (int i = 1, j = 0; i < textFields.size(); i += 2, j++) {

if (((TextField)textFields.get(i)).getText().length() > 0) {

// 设置速度 强行让1最快（手动doge

((Car)cars.get(j)).setSpeed(1.0/Double.parseDouble(((TextField)textFields.get(i)).getText())\*100);

((Car)cars.get(j)).play();

} else {

((Car)cars.get(j)).pause();

}

}

}

}

/\* 文件名：Car.java

\*/

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.scene.layout.Pane;

import javafx.scene.shape.Polygon;

import javafx.scene.shape.Rectangle;

import javafx.scene.shape.Circle;

import javafx.scene.paint.Color;

import javafx.util.Duration;

public class Car extends Pane {

private double x = 0;

private double y = 40;

private Timeline animation;

Car() {

drawCar();

animation = new Timeline(new KeyFrame(Duration.millis(100), e -> moveCar())); // 每50ms移动车辆

animation.setCycleCount(Timeline.INDEFINITE); // 动画无限循环

}

// 绘制车辆

private void drawCar() {

getChildren().clear();

Rectangle rectangle = new Rectangle(x, y - 20, 50, 10);

rectangle.setFill(Color.SPRINGGREEN);

Polygon polygon = new Polygon(x + 10, y - 20, x + 20, y - 30, x + 30,

y - 30, x + 40, y - 20);

polygon.setFill(Color.RED);

Circle circle1 = new Circle(x + 15, y - 5, 5);

Circle circle2 = new Circle(x + 35, y - 5, 5);

this.getChildren().addAll(rectangle, circle1, circle2, polygon);

}

void pause() {

animation.pause();

}

void play() {

animation.play();

}

void setSpeed(double speed) {

animation.setRate(speed);

}

private void moveCar() {

if (x <= getWidth()) {

x += 1;

}

else

x = 0;

drawCar();

}

}

# Project 2分析设计

## N.1问题分析

### N.1.1 目标

设计一个控制台应用，计算不同种类人群的税率

### N.1.2功能

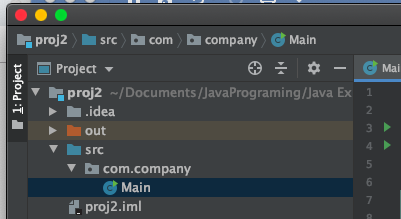
输出50000-60000之间收入四种不同人群所应缴纳的税率，以50位一档

### N.1.3性能

各项功能均无肉眼可见的等待时间

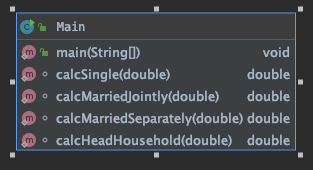
## N.2设计方案

### N.2.1模块划分



本项目只有一个模块

### N.2.2类图

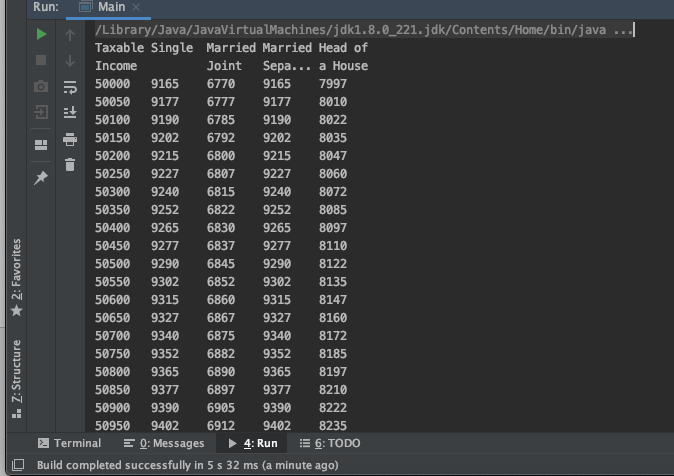


### N.2.3算法

#### N.2.3.1算法一：输出不同人群税率：按照表格进行计算。

## N.3实验结果

试验完成，可以达到预期效果，前20行输出截图如下。



## N.4 调试心得

没有遇到明显问题

## N.5源程序

/\* 文件名：Main.java

\*/

public class Main {

public static void main(String[] args) {

System.out.print("Taxable\tSingle\tMarried\tMarried\tHead of\t\n");

System.out.print("Income\t \tJoint \tSepa...\ta House\t\n");

for(int i=50000;i<=60000;i+=50){

System.out.printf("%d\t%d\t%d\t%d\t%d\t\n",i,(int)calcSingle(i),

(int)calcMarriedJointly(i),(int)calcMarriedSeparately(i),(int)calcHeadHousehold(i));

}

}

static double calcSingle(double income){

if(income<7300)return income\*0.1;

if(income<29700)return 730+(income-7300)\*0.15;

if(income<71950)return 4090+(income-29700)\*0.25;

if(income<150150)return 14652.5+(income-71950)\*0.28;

if(income<326450)return 36548.5+(income-150150)\*0.33;

return 94727.5+(income-326450)\*0.35;

}

static double calcMarriedJointly(double income){

if(income<14600)return income\*0.1;

if(income<59400)return 1460+(income-14600)\*0.15;

if(income<119950)return 8180+(income-59400)\*0.25;

if(income<182800)return 23317.5+(income-119950)\*0.28;

if(income<326450)return 40915.5+(income-182800)\*0.33;

return 88320.0+(income-326450)\*0.35;

}

static double calcMarriedSeparately(double income){

if(income<7300)return income\*0.1;

if(income<29700)return 730+(income-7300)\*0.15;

if(income<59975)return 4090+(income-29700)\*0.25;

if(income<91400)return 11658.75+(income-59975)\*0.28;

if(income<163225)return 20457.75+(income-91400)\*0.33;

return 44160+(income-163225)\*0.35;

}

static double calcHeadHousehold(double income){

if(income<10450)return income\*0.1;

if(income<39800)return 1045+(income-10450)\*0.15;

if(income<102800)return 5447.5+(income-39800)\*0.25;

if(income<166450)return 21197.5+(income-102800)\*0.28;

if(income<326450)return 39019.5+(income-166450)\*0.33;

return 91819.5+(income-326450)\*0.35;

}

}

# Project 3分析设计

## N.1问题分析

### N.1.1 目标

设计一个控制台应用，寻找最小公倍数

### N.1.2功能

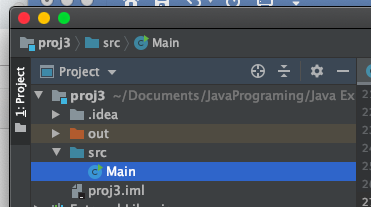
输出给定两数的最小公倍数以及得到某数分解的二维数组

### N.1.3性能

各项功能均无肉眼可见的等待时间

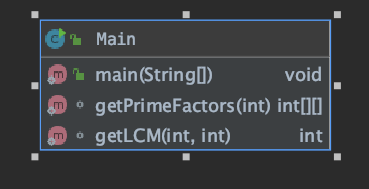
## N.2设计方案

### N.2.1模块划分



本项目只有一个模块

### N.2.2类图

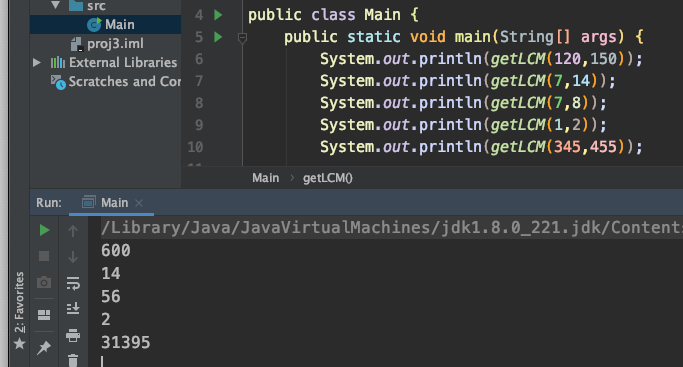


### N.2.3算法

#### N.2.3.1算法一：使用哈希表数据结构，将因数与存放两个数该因数个数的二维数组构成哈希表，以降低时间及空间复杂度。具体算法与指导书上相同。

## N.3实验结果

试验完成，可以达到预期效果，测试数据及结果截图如下。



## N.4 调试心得

没有遇到明显问题

## N.5源程序

/\* 文件名：Main.java

\*/

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

System.out.println(getLCM(120,150));

System.out.println(getLCM(7,14));

System.out.println(getLCM(7,8));

System.out.println(getLCM(1,2));

System.out.println(getLCM(345,455));

}

static int[][] getPrimeFactors(int num){

HashMap hashMap=new HashMap();

int ret[][];

for(int i=2;i<=num;i++){

if(num%i==0)hashMap.put(i,new int[2]);

while(num%i==0){

((int[])hashMap.get(i))[0]++;

num/=i;

}

}

int i=0;

ret= new int[hashMap.size()][2];

for (Object o :hashMap.entrySet()) {

ret[i][0]=(int)((Map.Entry)o).getKey();

ret[i][1]=((int[])(((Map.Entry)o).getValue()))[2];

i++;

}

return ret;

}

static int getLCM(int num1,int num2){

HashMap hashMap=new HashMap();

for(int i=2;i<=num1;i++){

if(num1%i==0)hashMap.put(i,new int[2]);

while(num1%i==0){

((int[])hashMap.get(i))[0]++;

num1/=i;

}

}

for(int i=2;i<=num2;i++){

if(num2%i==0&&!hashMap.containsKey(i))hashMap.put(i,new int[2]);

while(num2%i==0){

((int[])hashMap.get(i))[1]++;

num2/=i;

}

}

int i=1;

for (Object o :hashMap.entrySet()) {

i\*= Math.pow((int)((Map.Entry)o).getKey() ,Math.max(((int[])(((Map.Entry)o).getValue()))[0]

,((int[])(((Map.Entry)o).getValue()))[1]));

}

return i;

}

}

# Project 4分析设计

## N.1问题分析

### N.1.1 目标

设计一个Java程序，设计一个矩形类和一个时间类

### N.1.2功能

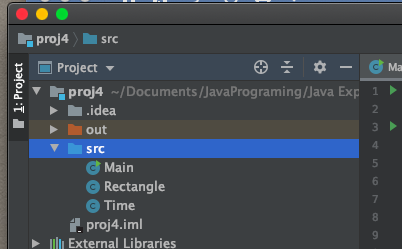
矩形可以计算面积和周长，时间可以由mills化成年月日时分秒

### N.1.3性能

各项功能均无肉眼可见的等待时间

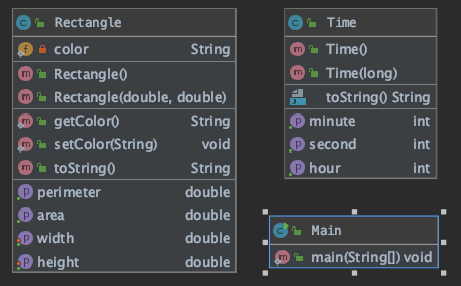
## N.2设计方案

### N.2.1模块划分



该项目有三个类。

### N.2.2类图

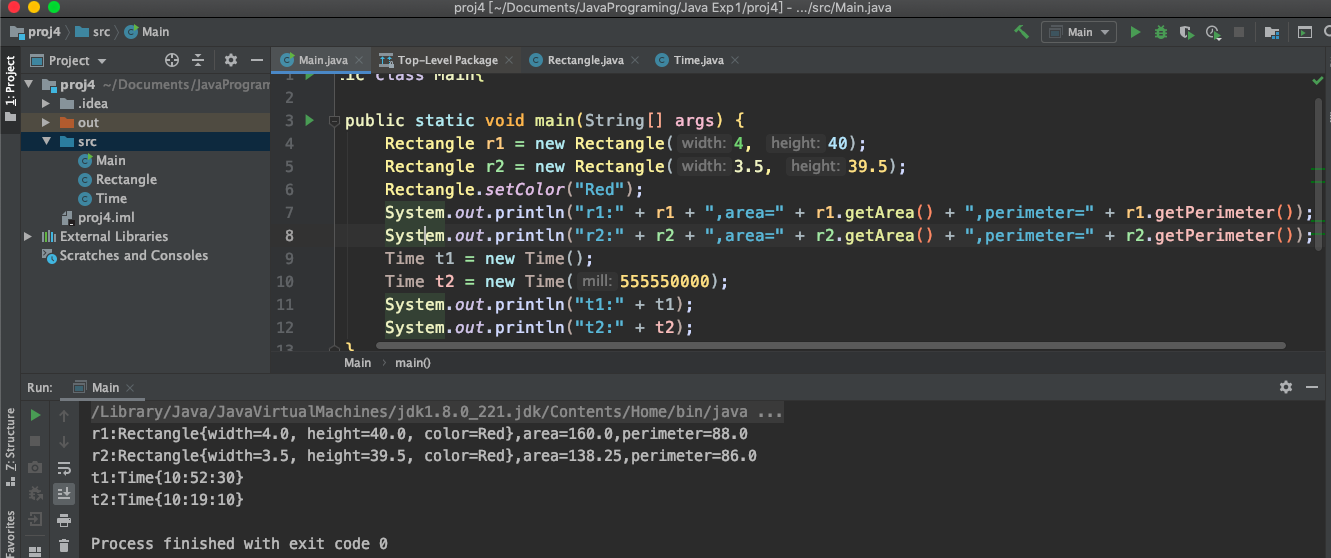


### N.2.3算法

#### 无特殊算法

## N.3实验结果

试验完成，可以达到预期效果，测试数据及结果截图如下。



## N.4 调试心得

没有遇到明显问题

## N.5源程序

/\* 文件名：Main.java

\*/

public class Main{

public static void main(String[] args) {

Rectangle r1 = new Rectangle(4, 40);

Rectangle r2 = new Rectangle(3.5, 39.5);

Rectangle.setColor("Red");

System.out.println("r1:" + r1 + ",area=" + r1.getArea() + ",perimeter=" + r1.getPerimeter());

System.out.println("r2:" + r2 + ",area=" + r2.getArea() + ",perimeter=" + r2.getPerimeter());

Time t1 = new Time();

Time t2 = new Time(555550000);

System.out.println("t1:" + t1);

System.out.println("t2:" + t2);

}

}

/\* Rectangle.java

\*/

public class Rectangle {

private double width;

private double height;

private static String color;

public Rectangle() {

this(150, 100);

}

public Rectangle(double width, double height) {

this.width = width;

this.height = height;

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

this.width = width;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

public static String getColor() {

return color;

}

public static void setColor(String color) {

Rectangle.color = color;

}

public double getArea() {

return width \* height;

}

public double getPerimeter(){

return 2 \* (width + height);

}

@Override

public String toString() {

return "Rectangle{" +

"width=" + width +

", height=" + height +

", color=" + color +

'}';

}

}

/\* Time.java

\*/

import java.text.DecimalFormat;

public class Time {

int hour;

int minute;

int second;

public Time() {

this(System.currentTimeMillis());

}

public Time(long mill){

mill /= 1000;

int m = (int) (mill % (60 \* 60 \* 24));

hour = m / 60 / 60;

second = m % 60;

minute = (m % (60 \* 60)) / 60;

}

public int getHour() {

return hour;

}

public int getMinute() {

return minute;

}

public int getSecond() {

return second;

}

@Override

public String toString() {

DecimalFormat d = new DecimalFormat("00");

return "Time{" +

d.format(hour) +

":" + d.format(minute) +

":" + d.format(second) +

'}';

}

}

# Project 5分析设计

## N.1问题分析

### N.1.1 目标

设计一个GUI程序，实现24点牌游戏

### N.1.2功能

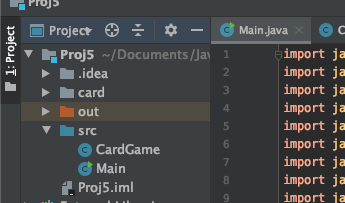
友好的显示图形见面，随机产生四张卡牌，并可以自动寻找正确答案，以及判断用户输入答案是否正确。

### N.1.3性能

各项功能均无肉眼可见的等待时间

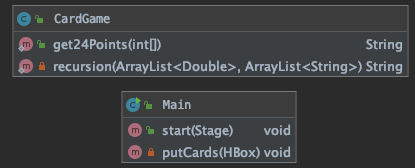
## N.2设计方案

### N.2.1模块划分



该项目有两个类。

### N.2.2类图



### N.2.3算法

#### N.2.3.1寻找结果算法：递归分治，第一次从四张牌中按顺序取两数分别尝试+-\*/运算，放入新数组中，第二次从这三个数中继续进行如上操作，进行递归

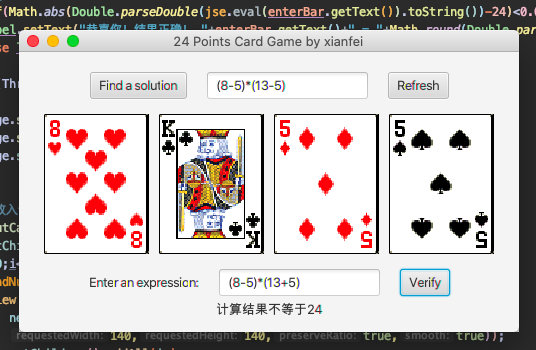
#### N.2.3.2判断表达式是否正确：使用js脚本解释器引擎当做表达式求值。

## N.3实验结果

试验完成，可以达到预期效果，截图如下。



当用户输入的表达式计算结果不为24时：



当用户输入的表达式使用的不是牌上的数时：



当用户输入的表达式不合法时：



## N.4 调试心得

没有遇到明显问题

## N.5源程序

/\* 文件名：Main.java

\*/

import javafx.application.Application;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.geometry.Insets;

import javafx.stage.Stage;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javax.script.ScriptEngine;

import javax.script.ScriptEngineManager;

import java.io.File;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Main extends Application {

private int[] nums =new int[4]; // 存储四张牌是啥

// 创建js解释器

private ScriptEngine jse = new ScriptEngineManager().getEngineByName("JavaScript");

@Override public void start(Stage primaryStage){

// 顶端区域

HBox upBox = new HBox(20);

upBox.setPadding(new Insets(5,3,10,3));

upBox.setAlignment(Pos.CENTER);

TextField foundBar=new TextField();

Button but1=new Button("Refresh");

Button but2=new Button("Find a solution");

upBox.getChildren().addAll(but2,foundBar,but1);

// 卡牌区域

HBox cardBox = new HBox(10);

cardBox.setAlignment(Pos.CENTER);

cardBox.setPadding(new Insets(0,5,10,5));

putCards(cardBox);

// 底部区域

HBox botBox = new HBox(20);

botBox.setAlignment(Pos.CENTER);

TextField enterBar=new TextField();

Button but3=new Button("Verify");

botBox.getChildren().addAll(new Label("Enter an expression:"),enterBar,but3);

Label label = new Label("Welcome!");

// 把这些东东放到一起去

VBox vbox = new VBox(5);

vbox.setAlignment(Pos.CENTER);

vbox.getChildren().addAll(upBox,cardBox,botBox,label);

Scene scene=new Scene(vbox,500,280);

// 注册按钮响应事件

but1.setOnAction(event -> putCards(cardBox));

but2.setOnAction(event -> foundBar.setText(CardGame.get24Points(nums)));

but3.setOnAction(event -> {

Pattern p = Pattern.compile("\\d+"); // 匹配数字

Matcher m = p.matcher(enterBar.getText());

// 验证表达式是否使用到牌中四个数

for(int i=0;m.find( )&&i<4;i++) {

boolean matched = false;

for (int j = 0; j < 4; j++) {

String str = m.group();

if (Integer.parseInt(str) == nums[j]) matched = true;

}

if(!matched){

label.setText("表达式中数字与牌上数字不匹配！");

return;

}

}

// 使用js脚本引擎计算表达式是否成立

try{ if(Math.abs(Double.parseDouble(jse.eval(enterBar.getText()).toString())-24)<0.000001)

label.setText("恭喜你！结果正确！ "+enterBar.getText()+" = "+Math.round(Double.parseDouble(jse.eval(enterBar.getText()).toString())));

else label.setText("计算结果不等于24");

}

catch (Throwable obj){label.setText("表达式不合法！");}

});

primaryStage.setTitle("24 Points Card Game by xianfei");

primaryStage.setScene(scene);

primaryStage.show();

}

// 向hbox中随机放入卡片 并更新nums数组

private void putCards(HBox cardBox){

cardBox.getChildren().clear();

for(int i=0;i<4;i++) {

int randNum=(int) (Math.random() \* 52 + 1);

ImageView iv=new ImageView(new Image(

new File("card/" + randNum+ ".png").toURI().toString(),

140,140,true,true));

cardBox.getChildren().addAll(iv);

if(randNum%13==0)nums[i]=13;else nums[i]=randNum%13;

}

}

}

/\* 文件名 CardGame.java

\*/

import java.util.ArrayList;

public class CardGame {

public static String get24Points(int[] nums) {

ArrayList<Double> numArrayList = new ArrayList<>(); // 用于存放数字

ArrayList<String> stringArrayList = new ArrayList<>(); // 用于存放数字的字符串形式

for (int num:nums) { // 将数组转换为ArrayList

numArrayList.add((double)num);

stringArrayList.add(Integer.toString(num));

}

String result = recursion(numArrayList, stringArrayList);

if (result == null) return "Can't found!";

return result;

}

// 递归，数组中取出两个数计算后结果放回数组

private static String recursion(ArrayList<Double> numArrayList, ArrayList<String> stringArrayList) {

String result = null;

int n = numArrayList.size();

if (n == 1) {

if (Math.abs(numArrayList.get(0) - 24.0) <= 0.000001) result = stringArrayList.get(0); // 若数组只有一个数，且为24，则将算式字符串返回

return result;

}

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

for (int j = i + 1; j < n; j++) {

ArrayList<Double> newNumArrayList = new ArrayList<Double>();// 用于递归的新数组

ArrayList<String> newStringArrayList = new ArrayList<String>();// 用于存放算式的新数组

// 任意取出两个数的不同组合情况

for (int r = 0; r < n; r++) {

if (r != i && r != j) {

newNumArrayList.add(numArrayList.get(r));

newStringArrayList.add(stringArrayList.get(r));

}

}

newNumArrayList.add(numArrayList.get(i) + numArrayList.get(j));// 加入取出的两数计算结果 a+b

newStringArrayList.add("(" + stringArrayList.get(i) + "+" + stringArrayList.get(j) + ")");// 记录两数计算的算式

result = recursion(newNumArrayList, newStringArrayList);//递归

if (result != null) return result; //若已有结果，则返回

newNumArrayList.remove(newNumArrayList.size() - 1);// 移除上次添加的两数计算结果

newStringArrayList.remove(newStringArrayList.size() - 1);

newNumArrayList.add(numArrayList.get(i) - numArrayList.get(j));//a-b

newStringArrayList.add("(" + stringArrayList.get(i) + "-" + stringArrayList.get(j) + ")");

result = recursion(newNumArrayList, newStringArrayList);

if (result != null) return result; //若已有结果，则返回

newNumArrayList.remove(newNumArrayList.size() - 1);

newStringArrayList.remove(newStringArrayList.size() - 1);

newNumArrayList.add(numArrayList.get(i) \* numArrayList.get(j));//a\*b

newStringArrayList.add(stringArrayList.get(i) + "\*" + stringArrayList.get(j));

result = recursion(newNumArrayList, newStringArrayList);

if (result != null) return result; //若已有结果，则返回

newNumArrayList.remove(newNumArrayList.size() - 1);

newStringArrayList.remove(newStringArrayList.size() - 1);

newNumArrayList.add(numArrayList.get(i) / numArrayList.get(j));//a/b

newStringArrayList.add(stringArrayList.get(i) + "/" + stringArrayList.get(j));

result = recursion(newNumArrayList, newStringArrayList);

if (result != null) return result; //若已有结果，则返回

newNumArrayList.remove(newNumArrayList.size() - 1);

newStringArrayList.remove(newStringArrayList.size() - 1);

newNumArrayList.add(numArrayList.get(j) - numArrayList.get(i));//b-a

newStringArrayList.add("(" + stringArrayList.get(j) + "-" + stringArrayList.get(i) + ")");

result =recursion(newNumArrayList, newStringArrayList);

if (result != null) return result; //若已有结果，则返回

newNumArrayList.remove(newNumArrayList.size() - 1);

newStringArrayList.remove(newStringArrayList.size() - 1);

newNumArrayList.add(numArrayList.get(j) / numArrayList.get(i));//b/a

newStringArrayList.add(stringArrayList.get(j) + "/" + stringArrayList.get(i));

result = recursion(newNumArrayList, newStringArrayList);

if (result != null) return result; //若已有结果，则返回

}

}

return null;

}

}