**Java实验报告**

本文是Java实验的实验报告。

**习题1 正八边形类**

**题目描述**：写一个正八边形类Octagon，要求从课本的GeometricObject类继承，并实现 Comparable和 Cloneable接口。写一个测试类，测试Octagon类的所有可用方法。

**解题思路**：首先在Octagon类内实现GeometricObject类中的getArea()和getPerimeter()方法，然后覆盖clone和compareTo这两个方法。在测试类中进行相应的测试。

**源代码**：

（1）抽象类文件

//package test;  
import java.util.Date;  
  
abstract class GeometricObject {  
 private String color = "white";  
 private boolean filled = false;  
 private Date dateCreated;  
  
 protected GeometricObject() {  
 this.dateCreated = new Date();  
 }  
  
 protected GeometricObject(String color, boolean filled) {  
 this.dateCreated = new Date();  
 this.color = color;  
 this.filled = filled;  
 }  
  
 public String getColor() {  
 return this.color;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 public boolean isFilled() {  
 return this.filled;  
 }  
  
 public void setFilled(boolean filled) {  
 this.filled = filled;  
 }  
  
 public Date getDateCreated() {  
 return this.dateCreated;  
 }  
  
 @Override  
 public String toString() {  
 return "created on " + dateCreated + "\ncolor: " + color +  
 " and filled: " + filled;  
 }  
  
 public abstract double getArea();  
  
 public abstract double getPerimeter();  
  
}

（2）继承

class Octagon extends GeometricObject implements Comparable, Cloneable {  
 double side;  
  
 public Octagon() { }  
 public Octagon(double side) {  
 this.side = side;  
 }  
 public Octagon(double side, String color, boolean filled) {  
 this.side = side;  
 setColor(color);  
 setFilled(filled);  
 }  
 public double getSide() {  
 return side;  
 }  
 public void setSide(double side) {  
 this.side = side;  
 }  
  
 @Override  
 public double getArea() {  
 return (2 + 4 / Math.*sqrt*(2)) \* side \* side;  
 }  
  
 @Override  
 public double getPerimeter() {  
 return 8 \* side;  
 }  
  
 @Override  
 public Object clone() throws CloneNotSupportedException {  
 return super.clone();  
 }  
  
 @Override  
 public int compareTo(Object o) {  
 if(this.side > ((Octagon)o).side)  
 return 1;  
 else if(this.side < ((Octagon)o).side)  
 return -1;  
 else  
 return 0;  
 }  
  
}

（3）测试

public class Test1 {  
 public static void main(String[] args) {  
 Octagon o = new Octagon(5);  
  
 System.*out*.println("Area: " + o.getArea());  
 System.*out*.println("Perimeter: " + o.getPerimeter());  
  
 Octagon o1 = new Octagon();  
 try {  
 o1 = (Octagon)o.clone();  
 }  
 catch(CloneNotSupportedException ex) {  
 System.*out*.println("CloneNotSupportedException");  
 }  
  
 System.*out*.println(o.compareTo(o1));  
 }  
}

**测试样例**：

构建一个边长为5的正八边形，计算其面积和周长，并克隆一个对象，比较两个对象。具体结果如下：

Area: 120.71067811865476

Perimeter: 40.0

0

**习题2 MyRational**

**题目描述**：改写课本Listing 13.13的分数类Rational，将分子和分母从类型long改为BigInteger，新版分数类名为MyRational。写一个测试类，测试MyRational。

**解题思路**：将课本上的源代码中的long类型数据全部换成BigInteger，同时使用BigInteger的静态方法进行四则运算。

**源代码**：

import java.math.BigInteger;  
  
public class MyRational extends Number implements Comparable{  
 private BigInteger numerator = new BigInteger("0");  
 private BigInteger denominator = new BigInteger("1");  
  
 public MyRational() {  
 this(new BigInteger("0"), new BigInteger("1"));  
 }  
  
 public MyRational(BigInteger numerator, BigInteger denominator) {  
 BigInteger gcd = *gcd*(numerator, denominator);  
 this.numerator = ((new BigInteger(denominator.compareTo(BigInteger.*ZERO*) + "")).multiply(numerator)).divide(gcd);  
 this.denominator = denominator.abs().divide(gcd);  
 }  
  
 private static BigInteger gcd(BigInteger n, BigInteger d) {  
 BigInteger n1 = n.abs();  
 BigInteger n2 = d.abs();  
 BigInteger gcd = BigInteger.*ONE*;  
  
 for(BigInteger k = BigInteger.*ONE*; k.compareTo(n1) <= 0 && k.compareTo(n2) <= 0; k = k.add(BigInteger.*ONE*)) {  
 if(n1.mod(k).compareTo(BigInteger.*ZERO*) == 0 && n2.mod(k).compareTo(BigInteger.*ZERO*) == 0)  
 gcd = k;  
 }  
  
 return gcd;  
 }  
  
 public BigInteger getNumerator() {  
 return numerator;  
 }  
  
 public BigInteger getDenominator() {  
 return denominator;  
 }  
  
 public MyRational add(MyRational secondRational) {  
 BigInteger n = numerator.multiply(secondRational.getDenominator()).add(  
 denominator.multiply(secondRational.getNumerator()));  
 BigInteger d = denominator.multiply(secondRational.getDenominator());  
 return new MyRational(n, d);  
 }  
  
 public MyRational subtract(MyRational secondRational) {  
 BigInteger n = numerator.multiply(secondRational.getDenominator()).subtract(  
 denominator.multiply(secondRational.getNumerator()));  
 BigInteger d = denominator.multiply(secondRational.getDenominator());  
 return new MyRational(n, d);  
 }  
  
 public MyRational multiply(MyRational secondRational) {  
 BigInteger n = numerator.multiply(secondRational.getNumerator());  
 BigInteger d = denominator.multiply(secondRational.getDenominator());  
 return new MyRational(n, d);  
 }  
  
 public MyRational divide(MyRational secondRational) {  
 BigInteger n = numerator.multiply(secondRational.getDenominator());  
 BigInteger d = denominator.multiply(secondRational.getNumerator());  
 return new MyRational(n, d);  
 }  
  
 @Override  
 public String toString() {  
 if(denominator.compareTo(BigInteger.*ONE*) == 0)  
 return numerator.toString() + "";  
 else  
 return numerator.toString() + "/" + denominator.toString();  
 }  
  
 @Override  
 public boolean equals(Object parm1) {  
 if((this.subtract((MyRational)(parm1))).getNumerator().compareTo(BigInteger.*ZERO*) == 0)  
 return true;  
 else  
 return false;  
 }  
  
 @Override  
 public int intValue() {  
 return (int)doubleValue();  
 }  
  
 @Override  
 public float floatValue() {  
 return (float)doubleValue();  
 }  
  
 @Override  
 public double doubleValue() {  
 return numerator.doubleValue() \* 1.0 / denominator.doubleValue();  
 }  
  
 @Override  
 public long longValue() {  
 return (long)doubleValue();  
 }  
  
 @Override  
 public int compareTo(Object o) {  
 if((this.subtract((MyRational)o)).getNumerator().compareTo(BigInteger.*ZERO*) == 1)  
 return 1;  
 else if((this.subtract((MyRational)o)).getNumerator().compareTo(BigInteger.*ZERO*) == -1)  
 return -1;  
 else  
 return 0;  
 }  
}

测试：

//package testjavafx;  
  
import java.io.\*;  
import java.util.Random;  
import java.util.jar.\*;  
import javafx.application.Application;  
import javafx.geometry.HPos;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Label;  
import javafx.scene.image.\*;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.GridPane;  
import javafx.stage.Stage;  
  
public class Test3 extends Application{  
 public static ImageView readImageFromJar(String jarname, String picname) {  
 ImageView imageView = null;  
 try {  
 JarFile jarFile = new JarFile(jarname);  
 JarEntry entry = jarFile.getJarEntry(picname);  
 InputStream in = jarFile.getInputStream(entry);  
 imageView = new ImageView(new Image(in));  
 in.close();  
 jarFile.close();  
 }  
 catch (IOException e) {  
 System.*err*.println("read file error.");  
 }  
 return imageView;  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 BorderPane pane = new BorderPane();  
  
 Random random = new Random((int)(System.*currentTimeMillis*()));  
 int img1Number = random.nextInt(54) + 1;  
 String img1Address = "cards/" + img1Number + ".png";  
  
 int img2Number;  
 do {  
 img2Number = random.nextInt(54) + 1;  
 }  
 while(img2Number == img1Number);  
 String img2Address = "cards/" + img2Number + ".png";  
  
 int img3Number;  
 do {  
 img3Number = random.nextInt(54) + 1;  
 }  
 while(img3Number == img1Number || img3Number == img2Number);  
 String img3Address = "cards/" + img3Number + ".png";  
  
 int img4Number;  
 do {  
 img4Number = random.nextInt(54) + 1;  
 }  
 while(img4Number == img1Number || img4Number == img2Number || img4Number == img3Number);  
 String img4Address = "cards/" + img4Number + ".png";  
  
 ImageView img1View = *readImageFromJar*("cards.jar", img1Address);  
 ImageView img2View = *readImageFromJar*("cards.jar", img2Address);  
 ImageView img3View = *readImageFromJar*("cards.jar", img3Address);  
 ImageView img4View = *readImageFromJar*("cards.jar", img4Address);  
  
 String card1Name = getName(img1Number);  
 String card2Name = getName(img2Number);  
 String card3Name = getName(img3Number);  
 String card4Name = getName(img4Number);  
  
 Label label1 = new Label(card1Name);  
 Label label2 = new Label(card2Name);  
 Label label3 = new Label(card3Name);  
 Label label4 = new Label(card4Name);  
  
 GridPane.*setHalignment*(label1, HPos.*CENTER*);  
 GridPane.*setHalignment*(label2, HPos.*CENTER*);  
 GridPane.*setHalignment*(label3, HPos.*CENTER*);  
 GridPane.*setHalignment*(label4, HPos.*CENTER*);  
  
 GridPane grid = new GridPane();  
 grid.setPadding(new Insets(5, 5, 5, 5));  
 grid.setHgap(10);  
 grid.setVgap(10);  
 grid.setAlignment(Pos.*CENTER*);  
 grid.add(img1View, 0, 0);  
 grid.add(img2View, 1, 0);  
 grid.add(img3View, 2, 0);  
 grid.add(img4View, 3, 0);  
 grid.add(label1, 0, 1);  
 grid.add(label2, 1, 1);  
 grid.add(label3, 2, 1);  
 grid.add(label4, 3, 1);  
 pane.setCenter(grid);  
  
 Scene scene = new Scene(pane, 400, 200);  
 primaryStage.setTitle("Four Cards");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 public String getName(int number) {  
 String name = "";  
  
 switch((number - 1) / 13) {  
 case 0: name = name + "黑桃"; break;  
 case 1: name = name + "红心"; break;  
 case 2: name = name + "方块"; break;  
 case 3: name = name + "梅花"; break;  
 default: {  
 if(number == 53)  
 name = name + "小 王";  
 else if(number == 54)  
 name = name + "大 王";  
 }  
 }  
  
 if(1 <= number && number <= 52) {  
 if((number - 1) % 13 == 0)  
 name = name + "A";  
 else if(1 <= (number - 1) % 13 && (number - 1) % 13 <= 9)  
 name = name + ((number - 1) % 13 + 1) + "";  
 else if((number - 1) % 13 == 10)  
 name = name + "J";  
 else if((number - 1) % 13 == 11)  
 name = name + "Q";  
 else  
 name = name + "K";  
 }  
  
 return name;  
 }  
}

**测试样例**：

按照测试类内的数据，输出的结果为：

r1 + r2 + r3 = 3/10000

r1 - r2 - r3 = -1/10000

r1 \* r2 \* r3 = 1/1000000000000

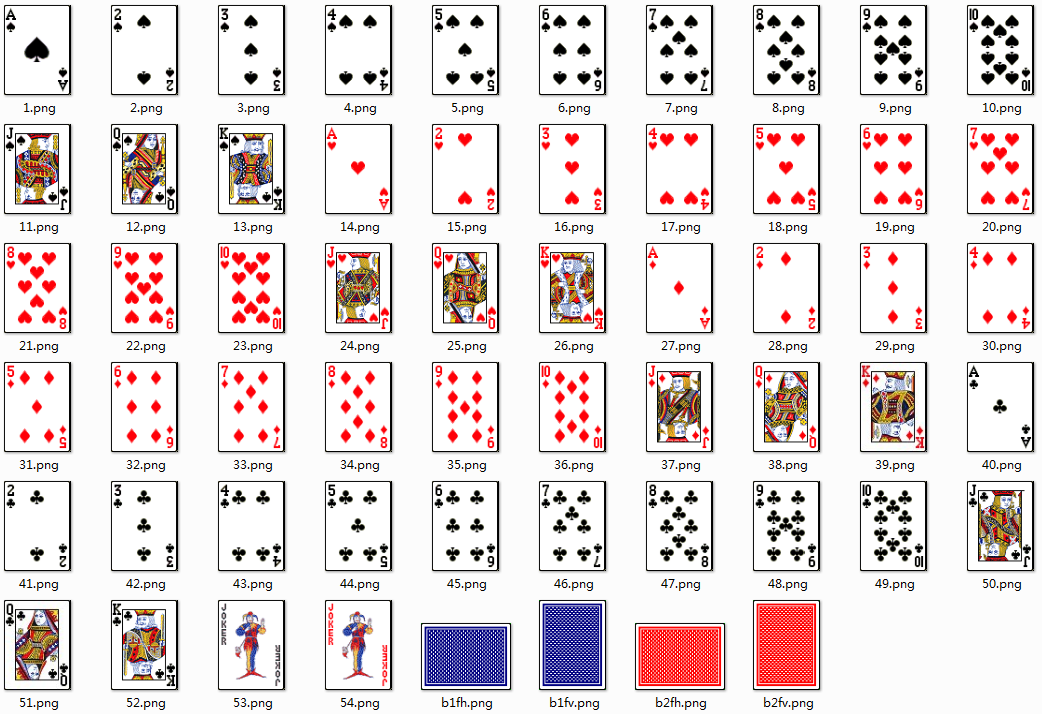
r1 / r2 / r3 = 10000

**习题3 发扑克牌**

**题目描述**：编程实现程序每次启动后，在界面上随机显示54张扑克牌中的4张，并在其下方显示花色名（如图）。扑克牌图片已打包到cards.jar文件中，命名规则见后。



效果图



扑克牌命名规则

**解题思路**：首先利用随机数，随机抽取四张不相同的扑克牌，然后从jar文件中打开相应的图片，并利用函数求出该扑克牌的名称，最后在窗口内予以显示。

**源代码**：

//package testjavafx;  
  
import java.io.\*;  
import java.util.Random;  
import java.util.jar.\*;  
import javafx.application.Application;  
import javafx.geometry.HPos;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Label;  
import javafx.scene.image.\*;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.GridPane;  
import javafx.stage.Stage;  
  
public class Test3 extends Application{  
 public static ImageView readImageFromJar(String jarname, String picname) {  
 ImageView imageView = null;  
 try {  
 JarFile jarFile = new JarFile(jarname);  
 JarEntry entry = jarFile.getJarEntry(picname);  
 InputStream in = jarFile.getInputStream(entry);  
 imageView = new ImageView(new Image(in));  
 in.close();  
 jarFile.close();  
 }  
 catch (IOException e) {  
 System.*err*.println("read file error.");  
 }  
 return imageView;  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 BorderPane pane = new BorderPane();  
  
 Random random = new Random((int)(System.*currentTimeMillis*()));  
 int img1Number = random.nextInt(54) + 1;  
 String img1Address = "cards/" + img1Number + ".png";  
  
 int img2Number;  
 do {  
 img2Number = random.nextInt(54) + 1;  
 }  
 while(img2Number == img1Number);  
 String img2Address = "cards/" + img2Number + ".png";  
  
 int img3Number;  
 do {  
 img3Number = random.nextInt(54) + 1;  
 }  
 while(img3Number == img1Number || img3Number == img2Number);  
 String img3Address = "cards/" + img3Number + ".png";  
  
 int img4Number;  
 do {  
 img4Number = random.nextInt(54) + 1;  
 }  
 while(img4Number == img1Number || img4Number == img2Number || img4Number == img3Number);  
 String img4Address = "cards/" + img4Number + ".png";  
  
 ImageView img1View = *readImageFromJar*("cards.jar", img1Address);  
 ImageView img2View = *readImageFromJar*("cards.jar", img2Address);  
 ImageView img3View = *readImageFromJar*("cards.jar", img3Address);  
 ImageView img4View = *readImageFromJar*("cards.jar", img4Address);  
  
 String card1Name = getName(img1Number);  
 String card2Name = getName(img2Number);  
 String card3Name = getName(img3Number);  
 String card4Name = getName(img4Number);  
  
 Label label1 = new Label(card1Name);  
 Label label2 = new Label(card2Name);  
 Label label3 = new Label(card3Name);  
 Label label4 = new Label(card4Name);  
  
 GridPane.*setHalignment*(label1, HPos.*CENTER*);  
 GridPane.*setHalignment*(label2, HPos.*CENTER*);  
 GridPane.*setHalignment*(label3, HPos.*CENTER*);  
 GridPane.*setHalignment*(label4, HPos.*CENTER*);  
  
 GridPane grid = new GridPane();  
 grid.setPadding(new Insets(5, 5, 5, 5));  
 grid.setHgap(10);  
 grid.setVgap(10);  
 grid.setAlignment(Pos.*CENTER*);  
 grid.add(img1View, 0, 0);  
 grid.add(img2View, 1, 0);  
 grid.add(img3View, 2, 0);  
 grid.add(img4View, 3, 0);  
 grid.add(label1, 0, 1);  
 grid.add(label2, 1, 1);  
 grid.add(label3, 2, 1);  
 grid.add(label4, 3, 1);  
 pane.setCenter(grid);  
  
 Scene scene = new Scene(pane, 400, 200);  
 primaryStage.setTitle("Four Cards");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 public String getName(int number) {  
 String name = "";  
  
 switch((number - 1) / 13) {  
 case 0: name = name + "黑桃"; break;  
 case 1: name = name + "红心"; break;  
 case 2: name = name + "方块"; break;  
 case 3: name = name + "梅花"; break;  
 default: {  
 if(number == 53)  
 name = name + "小 王";  
 else if(number == 54)  
 name = name + "大 王";  
 }  
 }  
  
 if(1 <= number && number <= 52) {  
 if((number - 1) % 13 == 0)  
 name = name + "A";  
 else if(1 <= (number - 1) % 13 && (number - 1) % 13 <= 9)  
 name = name + ((number - 1) % 13 + 1) + "";  
 else if((number - 1) % 13 == 10)  
 name = name + "J";  
 else if((number - 1) % 13 == 11)  
 name = name + "Q";  
 else  
 name = name + "K";  
 }  
  
 return name;  
 }  
}

**测试样例**：

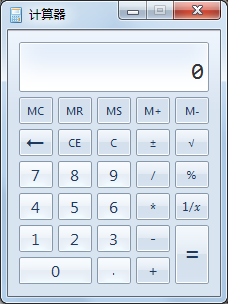
输出效果为：





**习题4 我的计算器**

**题目描述**：编程设计如图所示的界面。要求最后的界面布局尽可能接近效果图，不需要实现按钮功能。注意有几个按钮的背景色有所不同。



**解题思路**：将窗口分为两部分，上半部分为文本域，下半部分为网格，分别利用函数实现，再进行拼装。

**源代码**：

//package testjavafx;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.geometry.VPos;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.GridPane;  
import javafx.stage.Stage;  
  
public class Test4 extends Application {  
 @Override  
 public void start(Stage primaryStage) {  
 BorderPane pane = new BorderPane();  
 pane.setTop(getTextField());  
 pane.setBottom(getGrid());  
  
 Scene scene = new Scene(pane);  
 primaryStage.setScene(scene);  
 primaryStage.setTitle("计算器");  
 primaryStage.setResizable(false);  
 primaryStage.show();  
 }  
  
 private TextField getTextField() {  
 TextField textField = new TextField("0");  
 textField.setPadding(new Insets(30, 20, 5, 10));  
 textField.setEditable(false);  
 textField.setAlignment(Pos.*BOTTOM\_RIGHT*);  
 textField.setStyle("-fx-font: 22 Consolas;");  
 return textField;  
 }  
  
 private GridPane getGrid() {  
 GridPane pane = new GridPane();  
 pane.setAlignment(Pos.*CENTER*);  
 pane.setHgap(5);  
 pane.setVgap(5);  
 pane.setPadding(new Insets(5, 0, 0, 10));  
  
 Button b00 = new Button("MC");  
 b00.setPrefSize(40, 20);  
 pane.add(b00, 0, 0);  
  
 Button b01 = new Button("MR");  
 b01.setPrefSize(40, 20);  
 pane.add(b01, 1, 0);  
  
 Button b02 = new Button("MS");  
 b02.setPrefSize(40, 20);  
 pane.add(b02, 2, 0);  
  
 Button b03 = new Button("M+");  
 b03.setPrefSize(40, 20);  
 pane.add(b03, 3, 0);  
  
 Button b04 = new Button("M-");  
 b04.setPrefSize(40, 20);  
 pane.add(b04, 4, 0);  
  
 Button b10 = new Button("←");  
 b10.setPrefSize(40, 20);  
 pane.add(b10, 0, 1);  
  
 Button b11 = new Button("CE");  
 b11.setPrefSize(40, 20);  
 pane.add(b11, 1, 1);  
  
 Button b12 = new Button("C");  
 b12.setPrefSize(40, 20);  
 pane.add(b12, 2, 1);  
  
 Button b13 = new Button("±");  
 b13.setPrefSize(40, 20);  
 pane.add(b13, 3, 1);  
  
 Button b14 = new Button("√");  
 b14.setPrefSize(40, 20);  
 pane.add(b14, 4, 1);  
  
 Button b20 = new Button("7");  
 b20.setPrefSize(40, 20);  
 b20.setStyle("-fx-base: lightgray");  
 pane.add(b20, 0, 2);  
  
 Button b21 = new Button("8");  
 b21.setStyle("-fx-base: lightgray");  
 b21.setPrefSize(40, 20);  
 pane.add(b21, 1, 2);  
  
 Button b22 = new Button("9");  
 b22.setPrefSize(40, 20);  
 b22.setStyle("-fx-base: lightgray");  
 pane.add(b22, 2, 2);  
  
 Button b23 = new Button("/");  
 b23.setPrefSize(40, 20);  
 pane.add(b23, 3, 2);  
  
 Button b24 = new Button("%");  
 b24.setPrefSize(40, 20);  
 pane.add(b24, 4, 2);  
  
 Button b30 = new Button("4");  
 b30.setPrefSize(40, 20);  
 b30.setStyle("-fx-base: lightgray");  
 pane.add(b30, 0, 3);  
  
 Button b31 = new Button("5");  
 b31.setPrefSize(40, 20);  
 b31.setStyle("-fx-base: lightgray");  
 pane.add(b31, 1, 3);  
  
 Button b32 = new Button("6");  
 b32.setPrefSize(40, 20);  
 b32.setStyle("-fx-base: lightgray");  
 pane.add(b32, 2, 3);  
  
 Button b33 = new Button("\*");  
 b33.setPrefSize(40, 20);  
 pane.add(b33, 3, 3);  
  
 Button b34 = new Button("1/x");  
 b34.setPrefSize(40, 20);  
 pane.add(b34, 4, 3);  
  
 Button b40 = new Button("1");  
 b40.setPrefSize(40, 20);  
 b40.setStyle("-fx-base: lightgray");  
 pane.add(b40, 0, 4);  
  
 Button b41 = new Button("2");  
 b41.setPrefSize(40, 20);  
 b41.setStyle("-fx-base: lightgray");  
 pane.add(b41, 1, 4);  
  
 Button b42 = new Button("3");  
 b42.setPrefSize(40, 20);  
 b42.setStyle("-fx-base: lightgray");  
 pane.add(b42, 2, 4);  
  
 Button b43 = new Button("-");  
 b43.setPrefSize(40, 20);  
 pane.add(b43, 3, 4);  
  
 Button b44 = new Button("=");  
 b44.setPrefSize(40, 50);  
 GridPane.*setValignment*(b44, VPos.*TOP*);  
 pane.add(b44, 4, 4, 4, 5);  
  
 Button b50 = new Button("0");  
 b50.setPrefSize(85, 20);  
 GridPane.*setValignment*(b50, VPos.*TOP*);  
 b50.setStyle("-fx-base: lightgray");  
 pane.add(b50, 0, 5, 2, 5);  
  
 Button b52 = new Button(".");  
 b52.setPrefSize(40, 20);  
 b52.setStyle("-fx-base: lightgray");  
 pane.add(b52, 2, 5);  
  
 Button b53 = new Button("+");  
 b53.setPrefSize(40, 20);  
 pane.add(b53, 3, 5);  
  
 return pane;  
 }  
}

**测试样例**：

效果如下：

