import java.lang.Math;

import java.util.Scanner;

public class QuadraticEquation{

    static void quadraticEquation(double a, double b, double c){

        double firstRoot = 0, secondRoot = 0;

        double determinent = (b\*b)-(4\*a\*c);

        double squareRoot = Math.sqrt(determinent);

*if* (determinent > 0){

            firstRoot = (-b + squareRoot) / (2 \* a);

            secondRoot = (-b + squareRoot) / (2 \* a);

            System.out.println("First Root is " + firstRoot + " and " + " Second Root is " + secondRoot);

        }

*else* *if* (determinent == 0){

            System.out.println("Root is " + (-b + squareRoot) / (2 \* a));

        }

    }

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        System.out.print("Enter value of a: ");

        int a = input.nextInt();

        System.out.print("Enter value of b: ");

        int b = input.nextInt();

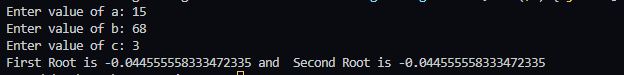
        System.out.print("Enter value of c: ");

        int c = input.nextInt();

        quadraticEquation(a, b, c);

    }

}



import java.util.Scanner;

public class FibonacciSeries {

    static int Fibonacci(int numbers){

*if* (numbers == 0){

*return* numbers;

        }

*if* (numbers == 1 || numbers == 2){

*return* 1;

        }

*return* (Fibonacci(numbers-1) + Fibonacci(numbers-2));

    }

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        System.out.println("Enter Number. ");

        int number = input.nextInt();

*for* (int i = 0; i < number; i++){

            System.out.print(Fibonacci(i) + " ");

        }

    }

}



import java.util.Scanner;

public class CommadineArgument {

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        System.out.println("Command Line argument is "+args[0]);

    }

}



import java.util.Scanner;

public class BubbleSorting {

    static void Sort(int arr[]){

        int temp;

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

*for* (int j = i+1; j < arr.length; j++) {

*if* (arr[i] > arr[j]){

                    temp = arr[i];

                    arr[i] = arr[j];

                    arr[j] = temp;

                }

            }

        }

        System.out.print("Sorted Numbers: ");

*for* (int i *:* arr) {

            System.out.print(i + " ");

        }

    }

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        int arr[];

        System.out.print("How many numbers you have? ");

        int len = input.nextInt();

        arr = *new* int[len];

        System.out.print("Enter Numbers: ");

*for* (int i = 0; i < len; i++){

            arr[i] = input.nextInt();

        }

        Sort(arr);

    }

}



import java.util.Scanner;

public class LinearSearch {

    static int Search(int number){

        int arr[] = {1,2,3,45,5,4,6,7,8,9,10};

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

*if* (arr[i] == number){

*return* i;

            }

        }

*return* -1;

    }

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        System.out.print("Enter the numner you want to search. ");

        int searchingNumber = input.nextInt();

*if* (Search(searchingNumber) == -1){

            System.out.println("Not found.");

        }

*else*{

            System.out.println("Element found at index " + Search(searchingNumber));

        }

    }

}



import java.util.Scanner;

public class BinarySearch {

    static int Search(int[] arr, int element, int beg, int end) {

        int mid = (beg + end) / 2;

*if* (beg <= end) {

*if* (element == arr[mid]) {

*return* mid;

            } *else* *if* (element < arr[mid]) {

*return* Search(arr, element, beg, mid - 1);

            } *else* {

*return* Search(arr, element, mid + 1, end);

            }

        } *else* {

*return* - 1;

        }

    }

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        int[] arr = {1,2,3,4,5,6,7,8,9,10};

        System.out.print("Enter element. ");

        int element = input.nextInt();

        int index = Search(arr, element, 0, arr.length-1);

*if* (index == -1){

            System.out.println("element not found.");

        } *else* {

            System.out.println("element found on index " + index);

        }

    }

}



public class CallByValueAndReference {

    private String name;

    private int id;

    private CallByValueAndReference obj;

    public void setName(String name){

*this*.name = name;

    }

    public void setId(int id){

*this*.id = id;

    }

    public String getName(){

*return* name;

    }

    public int getId(){

*return* id;

    }

    public void setValue(CallByValueAndReference obj){

        obj.name = "Muhammad Tayyab Bhutto";

        obj.id = 286;

*this*.obj = obj;

    }

    public CallByValueAndReference getValue(){

*return* obj;

    }

    public static void main(String[] args) {

        CallByValueAndReference obj = *new* CallByValueAndReference();

        obj.setName("Muhammad Muzammil Bhutto");

        obj.setId(786);

        System.out.println("Call By Value. " + obj.getName() +"  "+ obj.getId());

        obj.setValue(obj);

        System.out.println("Call By Reference. " + obj.name + "  "+obj.id);

    }

}



*// Why there is a need of static: it will execute first in program and will destroy when program will end.*

public class StaticBlockVarMeth {

    static int count = 0;

    static int Calculate(int val){

*return* val \* count;

    }

    public static void main(String[] args) {

        count = 10;

        System.out.println("Calling Calculate Method in Main Method. " + Calculate(10));

        System.out.println("Value of Static variable in Main after updating. " + count);

    }

    static {

        System.out.println("Hey this static Block And Calling Calculate Method in static block. " + Calculate(9));

    }

}



*// Three usages of super Keyword*

*// Variable having same name in super class and in subclass*

*// Method having same name in super class and in subclass*

*// Calling Constructor*

class SuperClass {

    String name;

    SuperClass(String name){

*this*.name = name;

    }

    void Display(){

        System.out.println("Name " + name);

    }

}

class SubClass extends SuperClass{

    String name;

    SubClass(String name){

*super*(name);    *// Super With Constructor*

        name = *super*.name;  *// Super With Variable Name*

    }

    void Display(){

*super*.Display();

    }

}

public class SuperKeyword {

    public static void main(String[] args) {

        SuperClass superClass = *new* SuperClass("Muhammad Tayyab Bhutto");

        System.out.println("before using super keyword.");

        superClass.Display();

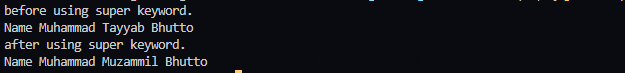
        SubClass subClass = *new* SubClass("Muhammad Muzammil Bhutto");

        System.out.println("after using super keyword.");

        subClass.Display();

    }

}



class Animal {

    private String name;

    public void setName(String name){

*this*.name = name;

    }

    public String getName(){

*return* name;

    }

}

class Dog extends Animal {

    private String name;

    public void setName(String name){

*super*.setName(name);

*this*.name = *super*.getName();

    }

    public String getName(){

*return* name;

    }

}

class Cat extends Animal {

    private String name;

    public void setName(String name){

*super*.setName(name);

*this*.name = *super*.getName();

    }

    public String getName(){

*return* name;

    }

}

public class MultiLevelInheritence {

    public static void main(String[] args) {

        Animal animal = *new* Animal();

        Dog dog = *new* Dog();

        Cat cat = *new* Cat();

        animal.setName("Animals");

        dog.setName("Sheru");

        cat.setName("Billy");

        System.out.println(animal.getName());

        System.out.println(dog.getName());

        System.out.println(cat.getName());

    }

}



*// Overloading and Overriding and two different kinds*

*// Overloading: when two or more methods having same name and different parameters is know as overloading it is compile time pollymorphism*

*// Overriding: when two or more methods having same name and same parameters is know as overriding it is run time pollymorphism*

class Application {

    private String appName;

    private String userName;

    private String password;

*// setData is an overloading method*

    public void setData(String appName){

*this*.appName = appName;

    }

    public void setData(String userName, String password){

*this*.userName = userName;

*this*.password = password;

    }

    public String getAppName(){

*return* appName;

    }

    public String getUsername(){

*return* userName;

    }

    public String getPassword(){

*return* password;

    }

}

class Profile extends Application{

    private String userName;

*// Overriding*

    public void setData(String userName){

*this*.userName = userName;

    }

    public String getUsername(){

*return* userName;

    }

}

public class OvelodingAndOverloading {

    public static void main(String[] args) {

        Application application = *new* Application();

        Profile profile = *new* Profile();

        application.setData("My App");

        application.setData("Muhammad Tayyab Bhutto", "asd123");

        profile.setData("muhammad-tayyab-bhutto");

        System.out.println("App Name: " + application.getAppName() + " \nUser Name: " + application.getUsername());

        System.out.println("Profile id: " + profile.getUsername());

    }

}



class Application {

    private String appName;

    private String userName;

    private String password;

*// Constructor overloading*

    Application(){}

    Application(String appName){

*this*.appName = appName;

    }

    Application(String userName, String password){

*this*.userName = userName;

*this*.password = password;

    }

*// setData is an overloading method*

    public void setData(String appName){

*this*.appName = appName;

    }

    public void setData(String userName, String password){

*this*.userName = userName;

*this*.password = password;

    }

    public String getAppName(){

*return* appName;

    }

    public String getUsername(){

*return* userName;

    }

    public String getPassword(){

*return* password;

    }

}

public class MethodAndConstructorOverloading {

    public static void main(String[] args) {

        Application application = *new* Application();

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

        System.out.println("Constructor Overloading");

        Application application1 = *new* Application("CodeSmashers");

        Application application2 = *new* Application("Muhammad Tayyab Bhutto" + "qwe123");

        System.out.println("App Name: " + application1.getAppName() + " \nUser Name: " + application2.getUsername());

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

        System.out.println("Method Overloading");

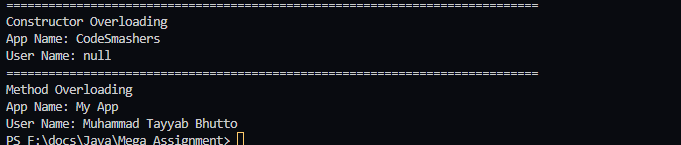
        application.setData("My App");

        application.setData("Muhammad Tayyab Bhutto", "asd123");

        System.out.println("App Name: " + application.getAppName() + " \nUser Name: " + application.getUsername());

    }

}



import java.util.Scanner;

public class TryCatch {

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        int number;

        int[] arr;

*try*{

            System.out.print("Enter Size Of Array: ");

            number = input.nextInt();

            arr = *new* int[number];

        } *catch* (NegativeArraySizeException e){

            System.out.println(e);

            System.out.print("Enter Again Size Of Array: ");

            number = input.nextInt();

            arr = *new* int[number];

        }

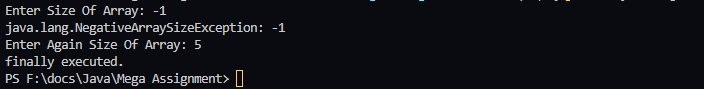
*finally*{

            System.out.println("finally executed.");

        }

    }

}



import java.util.Scanner;

class EmployeeIdException extends Exception{

    public EmployeeIdException(String exception){

*super*(exception);

    }

}

class Employee{

    private int id;

    public void setId(int id) throws EmployeeIdException{

*if* (id <= 0 || id > 9999){

*throw* *new* EmployeeIdException("Invalid Employee ID.");

        }

*else* {

*this*.id = id;

        }

    }

    public int getId(){

*return* id;

    }

}

public class UserDefinedExceptions{

    public static void main(String[] args) {

        Scanner input = *new* Scanner(System.in);

        Employee employee = *new* Employee();

        int id;

*try* {

            System.out.print("Enter ID: ");

            id = input.nextInt();

            employee.setId(id);

        } *catch* (EmployeeIdException e) {

            System.out.println(e.getMessage());

*try*{

                System.out.print("Please Enter ID Again: ");

                id = input.nextInt();

                employee.setId(id);

            }

*catch*(EmployeeIdException ex){

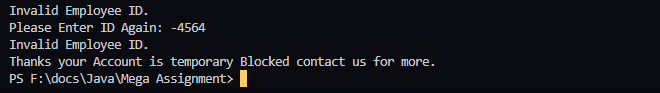
                System.out.println(ex.getMessage());

                System.out.println("Thanks your Account is temporary Blocked contact us for more.");

            }

        }

    }

}

package Animal;  
  
public abstract class Animal {  
 protected int legs;  
 protected Animal(int legs){  
 this.legs = legs;  
 }  
 public void walk(){  
 System.*out*.println("Animal walks.");  
 }  
 public abstract void eat();  
}

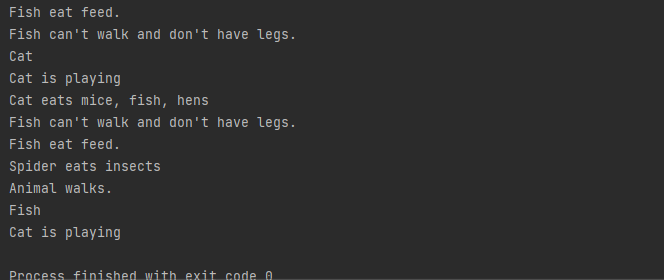
package Animal;  
  
public class Spider extends Animal{  
 protected Spider(){  
 super(8);  
 }  
  
 @Override  
 public void eat() {  
 System.*out*.println("Spider eats insects");  
 }  
}

package Animal;  
  
public interface Pet {  
 public abstract String getName();  
 public abstract void setName(String name);  
 public abstract void play();  
}

package Animal;  
  
public class Cat extends Animal implements Pet{  
 String name;  
 Cat(String name){  
 super(4);  
 this.name = name;  
 }  
 Cat(){  
 this("Cat");  
 }  
 @Override  
 public void eat() {  
 System.*out*.println("Cat eats mice, fish, hens");  
 }  
 @Override  
 public void setName(String name){  
 this.name = name;  
 }  
 @Override  
 public String getName(){  
 return name;  
 }  
 @Override  
 public void play(){  
 System.*out*.println("Cat is playing");  
 }  
  
}

package Animal;  
  
public class Fish extends Animal implements Pet{  
 String name;  
 protected Fish(){  
 super(0);  
 }  
 @Override  
 public void walk(){  
 System.*out*.println("Fish can't walk and don't have legs.");  
 }  
 @Override  
 public void eat(){  
 System.*out*.println("Fish eat feed.");  
 }  
  
 @Override  
 public void play() {  
 System.*out*.println("Fish is playing in water.");  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String getName() {  
 return name;  
 }  
  
}

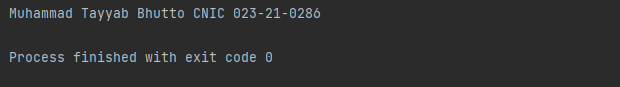
package Animal;  
  
public class Main {  
 public static void main(String[] args) {  
 Fish d = new Fish();  
 Cat c = new Cat("Fluffy");  
 Animal a = new Fish();  
 Animal e = new Spider();  
 Pet p = new Cat();  
 d.eat();  
 d.walk();  
 c.setName("Cat");  
 System.*out*.println(c.getName());  
 c.play();  
 c.eat();  
 a.walk();  
 a.eat();  
 e.eat();  
 e.walk();  
 p.setName("Fish");  
 System.*out*.println(p.getName());  
 p.play();  
 }  
}



package Employee;  
public class Employee {  
 private String firstName;  
 private String lastName;  
 private String nationalIdCardNumber;  
 public Employee () {  
 }  
 public Employee(String firstName, String lastName, String nationalIdCardNumber) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.nationalIdCardNumber = nationalIdCardNumber;  
 }  
 void setFirstName(String firstName){  
 this.firstName = firstName;  
 }  
 String getFirstName(){  
 return firstName;  
 }  
 void setLastName(String lastName){  
 this.lastName = lastName;  
 }  
 String getLastName(){  
 return lastName;  
 }  
 void setNationalIdCardNumber(String nationalIdCardNumber){  
 this.nationalIdCardNumber = nationalIdCardNumber;  
 }  
 String getNationalIdCardNumber(){  
 return nationalIdCardNumber;  
 }  
 public String toString() {  
 return firstName+ " " + lastName + " CNIC " + nationalIdCardNumber;  
 }  
 double earnings(){  
 return 0.0;  
 }  
}

//Main

import Employee.\*;  
public class Main {  
 public static void main(String[] args) {  
 Employee employee = new Employee("Muhammad" , "Tayyab Bhutto", "023-21-0286");  
 System.*out*.println(employee.toString());  
 }  
}



import java.awt.*\**;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.*\**;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.*\**;

import java.util.List;

class solveExpression {

    private String exp;

    protected int returnType = 0; */\* 0, 1, 2 \*/*

    protected String ans[] = {"Syntax Error !", "Math Error !", ""};

    private StringTokenizer element;

    private List<String> infix, prefix;

    protected solveExpression(String exp) {

*this*.exp = exp;

*if* (!error()) {

            infixToPrefix();

            calculatePrefix();

        }

    }

    protected String Answer() {

*return* ans[returnType];

    }

    private boolean error() {

        element = *new* StringTokenizer(exp, "+-x/()\*", true);

        infix = *new* ArrayList<String>();

*while* (element.hasMoreElements())

            infix.add(*new* String(element.nextElement().toString()));

*for* (int i = 0; i < infix.size(); i++) {

            String s = infix.get(i);

*if* (s.equals("+") || s.equals("-") || s.equals("(")) {

*if* (i == infix.size()-1) *return* true;

                String t = infix.get(i+1);

*if* (t.equals("x") || t.equals("/") || t.equals(")")) *return* true;

*if* (s.equals("+") || s.equals("-")) {

*if* (t.equals("+")) {

                        infix.remove(i+1);

                        i--;

                    }

*else* *if* (t.equals("-")) {

                        infix.set(i, s.equals("+") *?* "-" *:* "+");

                        infix.remove(i+1);

                        i--;

                    }

*else* {

*if* (i == 0) {infix.add(i--, "0"); *continue*;}

                        String g = infix.get(i-1);

*if* (t.equals("(")) {

*if* (g.equals("x") || g.equals("/")) {

                                infix.set(i, s.equals("+") *?* "1" *:* "-1");

                                infix.add(i+1, g);

                                i--;

                            }

*else* *if* (g.equals("(")) {

*if* (s.equals("+")) infix.remove(i--);

*else* infix.add(i--, "0");

                            }

*continue*;

                        }

*if* (g.equals("(") || g.equals("x") || g.equals("/")) {

*if* (s.equals("+")) infix.remove(i--);

*else* {

                                infix.remove(i);

                                StringBuilder num = *new* StringBuilder("-");

                                num.append(infix.get(i));

                                infix.set(i, *new* String(num));

                                i--;

                            }

                        }

                    }

                }

            }

*else* *if* (s.equals("x") || s.equals("/")) {

*if* (i == 0 || i == infix.size()-1) *return* true;

                String t = infix.get(i+1);

*if* (t.equals("x") || t.equals("/") || t.equals(")")) *return* true;

            }

*else* *if* (s.equals(")")) {

*if* (i == 0) *return* true;

*if* (i == infix.size()-1) *continue*;

                String t = infix.get(i+1);

*if* (t.equals("+") || t.equals("-") || t.equals("x") || t.equals("/") || t.equals(")")) *continue*;

*if* (t.equals("(")) {

                    infix.add(i+1, "x"); *continue*;

                }

*return* true;

            }

*else* {

*try* {

                    double t = (double) Double.parseDouble(s);

                }

*catch* (NumberFormatException e) {

*return* true;

                }

*if* (i == infix.size()-1) *continue*;

                String t = infix.get(i+1);

*if* (t.equals("(")) infix.add(i+1, "x");

            }

        }

*return* checkOpenClose();

    }

    private boolean checkOpenClose() {

        int cnt = 0;

*for* (String s *:* infix) {

*if* (s.equals("(")) cnt++;

*else* *if* (s.equals(")")) {

*if* (cnt == 0) *return* true;

                cnt--;

            }

        }

*for* (; cnt > 0; cnt--) infix.add(")");

*return* cnt == 0 *?* false *:* true;

    }

    private int rank(String s) {

*if* (s.equals("x") || s.equals("/")) *return* 2;

*if* (s.equals("+") || s.equals("-")) *return* 1;

*return* 0;

    }

    private void infixToPrefix() {

        Collections.reverse(infix);

        Deque<String> st = *new* LinkedList<String>();

        prefix = *new* ArrayList<String>();

*for* (String t *:* infix) {

*if* (t.equals("+") || t.equals("-") || t.equals("x") || t.equals("/")) {

*while* (!st.isEmpty() && rank(st.peek()) > rank(t)) {

                    prefix.add(st.pop());

                }

                st.push(t);

            }

*else* *if* (t.equals(")")) {

                st.push(t);

            }

*else* *if* (t.equals("(")) {

*while* (!st.isEmpty() && !st.peek().equals(")")) {

                    prefix.add(st.pop());

                }

                st.pop();

            }

*else* {

                prefix.add(t);

            }

        }

*while* (!st.isEmpty()) prefix.add(st.pop());

        Collections.reverse(prefix);

    }

    private void calculatePrefix() {

        Collections.reverse(prefix);

        Deque<Double> st = *new* LinkedList<Double>();

*for* (String t *:* prefix) {

*if* (t.equals("+") || t.equals("-") || t.equals("x") || t.equals("/")) {

                double a = st.pop(), b = st.pop();

*if* (t.equals("+")) a += b;

*else* *if* (t.equals("-")) a -= b;

*else* *if* (t.equals("x")) a \*= b;

*else* {

*if* (Math.abs(b) <= Double.MIN\_NORMAL) {

                        returnType = 1;

*return* ;

                    }

                    a /= b;

                }

                st.push(a);

            }

*else* {

                st.push(Double.parseDouble(t));

            }

        }

        returnType = 2;

        ans[2] = st.pop().toString();

    }

}

public class SimpleCalculator extends JFrame {

    public SimpleCalculator() {

*this*.initComponent();

*this*.pack();

*this*.setLocationRelativeTo(null);

*this*.setVisible(true);

    }

    private void initComponent() {

        labelCalculator = *new* JLabel();

        ScreenUserInterface = *new* JPanel();

        screenInput = *new* JTextField();

        screenOutput = *new* JTextField();

        ButtonUserInterface = *new* JPanel();

        copyright = *new* JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

        setAutoRequestFocus(false);

        setBackground(*new* java.awt.Color(255, 255, 255));

        setPreferredSize(*new* java.awt.Dimension(500, 600));

        setResizable(false);

        getContentPane().setLayout(*new* java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 0, 10));

        setTitle("Calculator - Using Java Swing ");

        setFocusTraversalKeysEnabled(false);

        labelCalculator.setBackground(*new* Color(0, 0, 0));

        labelCalculator.setFont(*new* Font("Verdana", 1, 30));

        labelCalculator.setForeground(*new* Color(0, 255, 51));

        labelCalculator.setHorizontalAlignment(SwingConstants.CENTER);

        labelCalculator.setText("CALCULATOR");

        labelCalculator.setOpaque(true);

        labelCalculator.setPreferredSize(*new* Dimension(480, 50));

        getContentPane().add(labelCalculator);

        ScreenUserInterface.setPreferredSize(*new* java.awt.Dimension(450, 150));

        ScreenUserInterface.setLayout(*new* java.awt.GridLayout(2, 0, 0, 20));

        screenInput.setFont(*new* java.awt.Font("Courier New", Font.BOLD, 30));

        screenInput.setHorizontalAlignment(javax.swing.JTextField.LEFT);

        screenInput.setBorder(*new* javax.swing.border.LineBorder(*new* java.awt.Color(102, 102, 102), 3, true));

        screenInput.setSelectedTextColor(Color.WHITE);

        screenInput.setSelectionColor(Color.DARK\_GRAY);

        addScreenInputEvent();

        screenOutput.setEditable(false);

        screenOutput.setFont(*new* java.awt.Font("Courier New", Font.BOLD, 30));

        screenOutput.setForeground(*new* java.awt.Color(0, 51, 204));

        screenOutput.setOpaque(true);

        screenOutput.setBackground(Color.WHITE);

        screenOutput.setHorizontalAlignment(javax.swing.JTextField.RIGHT);

        screenOutput.setBorder(*new* javax.swing.border.LineBorder(*new* java.awt.Color(102, 102, 102), 3, true));

        screenOutput.setSelectedTextColor(Color.WHITE);

        screenOutput.setSelectionColor(Color.DARK\_GRAY);

        ScreenUserInterface.add(screenInput);

        ScreenUserInterface.add(screenOutput);

        getContentPane().add(ScreenUserInterface);

        ButtonUserInterface.setPreferredSize(*new* java.awt.Dimension(420, 270));

        ButtonUserInterface.setLayout(*new* java.awt.GridLayout(0, 5, 10, 10));

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

            b[i] = *new* JButton(stringButton[i]);

            b[i].setFont(*new* java.awt.Font("Verdana", 1, 25));

            b[i].setMargin(*new* java.awt.Insets(0, 0, 0, 0));

            b[i].setForeground(Color.black);

        }

        b[10].setForeground(*new* Color(0,50,162));

        b[11].setForeground(*new* Color(0,50,162));

        b[12].setForeground(*new* Color(0,50,162));

        b[19].setForeground(*new* Color(200, 0, 0));

*for*(int i=13;i<=16;i++) b[i].setForeground(*new* Color(200, 0, 0));

        b[17].setForeground(Color.red);

        b[18].setForeground(Color.red);

        b[10].setFont(*new* java.awt.Font("Verdana", 1, 35));

        b[13].setFont(*new* java.awt.Font("Verdana", 1, 30));

        b[14].setFont(*new* java.awt.Font("Verdana", 1, 40));

        b[15].setFont(*new* java.awt.Font("Verdana", 1, 30));

        b[19].setFont(*new* java.awt.Font("Verdana", 1, 30));

        addButtonEvent();

*for* (int i = 0; i < stringButton.length; i++) ButtonUserInterface.add(b[orderButtonDisplay[i]]);

        getContentPane().add(ButtonUserInterface);

        copyright.setBackground(*new* Color(0, 0, 0));

        copyright.setFont(*new* Font("Verdana", 1, 23));

        copyright.setForeground(*new* Color(0, 255, 0));

        copyright.setHorizontalAlignment(SwingConstants.CENTER);

        copyright.setText("Muhammad Tayyab Bhutto");

        copyright.setOpaque(true);

        copyright.setPreferredSize(*new* Dimension(480, 50));

        getContentPane().add(copyright);

    }

    private void updateScreenInput(String add) {

        StringBuilder cur = *new* StringBuilder(screenInput.getText());

        int pos = screenInput.getCaretPosition();

        cur.insert(pos, add);

        screenInput.setText(*new* String(cur));

        screenInput.setCaretPosition(pos+add.length());

    }

    private void addScreenInputEvent() {

        screenInput.addKeyListener(*new* KeyAdapter() {

            @Override

            public void keyTyped(KeyEvent e) {

                e.consume();

                char c = e.getKeyChar();

                String sc = Character.toString(c);

*for* (int i = 0; i < 17; i++) {

*if* (sc.equals(stringButton[i])) {

                        updateScreenInput(sc); *break*;

                    }

                }

*if* (c == '\*') updateScreenInput("x");

*if* (c == '=' || c == KeyEvent.VK\_ENTER) {

                    String exp = screenInput.getText();

                    screenOutput.setText(*new* solveExpression(exp).Answer());

                }

            }

        });

    }

    private void addButtonEvent() {

*for* (int i = 0; i <= 16; i++) {

            String g = stringButton[i];

            b[i].addActionListener(*new* ActionListener() {

                @Override

                public void actionPerformed(ActionEvent e) {

                    updateScreenInput(g);

                }

            });

        }

        b[17].addActionListener(*new* ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                screenInput.setText("");

                screenOutput.setText("");

            }

        });

        b[18].addActionListener(*new* ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String s = screenInput.getText();

*if* (s.length() == 0) *return* ;

                int pos = screenInput.getCaretPosition();

*if* (pos == 0) *return* ;

                String t = s.substring(0, pos-1);

                screenInput.setText(t.concat(s.substring(pos, s.length())));

                screenInput.setCaretPosition(pos-1);

            }

        });

        b[19].addActionListener(*new* ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String exp = screenInput.getText();

                screenOutput.setText(*new* solveExpression(exp).Answer());

            }

        });

    }

*/\* Variables declaration\*/*

    private javax.swing.JPanel ButtonUserInterface;

    private javax.swing.JPanel ScreenUserInterface;

    private javax.swing.JLabel labelCalculator;

    private javax.swing.JLabel copyright;

    private javax.swing.JTextField screenInput;

    private javax.swing.JTextField screenOutput;

    private String stringButton[] = {"0","1","2","3","4","5","6","7","8","9",

                                    ".","(",")","+","-","x","/","AC","DEL","="};

    private JButton b[] = *new* JButton[stringButton.length];

    private int orderButtonDisplay[] = {

        7,  8,  9,  18, 17,

        4,  5,  6,  15, 16,

        1,  2,  3,  13, 14,

        0,  10, 19, 11, 12,

    };

    public static void main(String[] args) {

*new* SimpleCalculator();

    }

}

