HIGHER EDUCATIONAL ESTABLISHMENT OF UKOOPSPILKA
«POLTAVA UNIVERSITY OF ECONOMICS AND TRADE»

EDUCATIONAL AND SCIENTIFIC INSTITUTE OF INTERNATIONAL EDUCATION

FORM OF DAY EDUCATION
DEPARTMENT OF COMPUTER SCIENCES AND INFORMATION TECHNOLOGY

Allowed for protection
Head of the department _____ Olkhovska O.V.
(signature)
"_____"_____2022

EXPLANATORY NOTE
FOR THE GRADUATE THESIS

ON THE TOPIC «APPLICATION OF ARRAYS IN JAVA PROGRAMMING LANGUAGE FOR DEVELOPMENT OF SOFTWARE»

from specialty 122 «Computer science»
educational program «Computer science»
master's degree

The executor of the work is Sufian Milad Ghrmida
_____ "___"_____2022.
(signature)
Scientific supervisor, Ph.D. k.ped.s., Oksana Koshova

_____ "___"_____2022.
(signature)

POLTAVA - 2022

HIGHER EDUCATIONAL ESTABLISHMENT OF UKOOPSPILKA
"POLTAVA UNIVERSITY OF ECONOMICS AND TRADE"

I APPROVE

Head of the department_____Olkhovska O.V.
" ___» _____2022

TASKS AND CALENDAR SCHEDULE
OF EXECUTION OF DIPLOMA THESIS

Graduate of higher education in specialty 122 "Computer science"
Educational program "Computer Science"
Surname, first name, patronymic _ Sufian Milad Ghrmida_
1. **The topic «Application of Arrays in Java Programming Language for Development of Software»**
approved by the rector's order No. 139-H from 01. 09. 2022
The deadline for the student to submit a thesis «__» _____ 2022

2. Source data for the master's thesis: publications on the topic of educational simulators in distance courses in computer science.

3. Content of the explanatory note (list of issues to be developed)

LIST OD SYMBOLS, UNITS, ABBREVIATIONS, RERMS
AN INTRODUCTION
CHAPTER 1. PROBLEM STATEMENT
- 1.1. Problem Statement
- 1.2. Information Review

CHAPTER2. THEORETICAL PART
- 2.1. Array statement
- 2.2. The purposes of an Array
- 2.3. Array Lists

CHAPTER 3. PRACTICAL PART
- 3.1. NetBeans in Java
- 3.2. Java Swing
- 3.3. Examples of Application of Array in Java
- 3.4. Design And Programming of The Scientific Calculator

CONCLUSIONS
REFERENCES
APPENDIX A. Algorithm translation
Descriptive Algorithm
APPENDIX B. program code

4. List of graphic material : 3-4 sheets of block diagrams, other necessary illustrations.

5. Consultants of sections of the master's thesis

Section	Surname, initials, position of consultant	Signature, date	
		issued the task	accepted the task
1. Statement of the problem	Oksana Koshova		
2. Information review	Oksana Koshova		
3. Theoretical part	Oksana Koshova		
4. Practical implementation	Oksana Koshova		

6. Calendar schedule of master's work

The content of the work	Deadline	Actual performance
1. Introduction		
2. Study of methodological recommendations and standards and report to the manager		
3. Formulation of the problem		
4. Information review of library and Internet sources		
5. Theoretical part		
6. The practical part		
7. Completion of registration		
8. Student report at the department		
9. Editing (if necessary), reviewing		

Issue date of task " ___ " _____2022.

Sufian Milad Ghrmida, a higher education graduate

Scientific supervisor, Ph.D., k.ped.n., Oksana Koshova.

The results of the thesis defense

The thesis was evaluated at _____

(points, assessment according to the national scale, assessment according to ECTS)

Minutes of the meeting of the EC No. ____ of "____" _____ 2022.

Secretary of the EC_____ _____

(signature) (initials and surname)

I approve
Chief department _____
Ph.D.. k.ph.-m.s. O. Olkhovska
"____" _____ 2022

Agreed
Supervisor _____
associate professor, Ph.D. O. Koshova
"____" _____ 2022

Plan

Thesis of a higher education graduate with a master's degree
majors 122 Computer science
educational program 122 Computer science
Surname, first name, patronymic____ Sufian Milad Ghrmida _____
On the topic "Application of Arrays In Java Programming Language For Development Of Software"

CHAPTER 1. PROBLEM STATEMENT
1.1. Problem Statement
1.2. Information Review
CHAPTER2. THEORETICAL PART
2.1. Array statement
2.2. The purposes of an Array
2.3. Array Lists
CHAPTER 3. PRACTICAL PART
3.1. NetBeans in Java
3.2. Java Swing
3.3. Examples of Application of Array in Java
3.4. Design And Programming of The Scientific Calculator
CONCLUSIONS
REFERENCES
APPENDIX A. Algorithm translation
Descriptive Algorithm
APPENDIX B. program code

Graduate of higher education _____ Sufian Milad Ghrmida
"____" _____ 2022

ABSTRACT

Note: 87 pp., including the main part is 40 pages, 18 sources.

The purpose of the Grade work. The purpose of the grade work is development of software with application of arrays in Java programming language.

The object of the Grade work is process of professional education of students majoring in Computer Sciences.

The subject of the Grade work is software with application of arrays in java programming language.

Keywords: Java, Java Swing Platform, “Array in Java”, NetBeans, Math Class.

CONTENT

LIST OF SYMBOLS, UNITS, ABBREVIATIONS, TERMS

SYMBOLS, UNITS, ABBREVIATIONS, TERMS	Explanation of symbols, units, abbreviations, terms
Simulator	a program enabling a computer to execute program
class	A user-defined data type, which holds it's own data members and member functions, which can be accessed and used by creating an instance of that class
Math class	Java Math class provides several methods to work on math calculations like min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs() etc.
JDK	The Java Development Kit is a distribution of Java Technology by Oracle Corporation. It implements the Java Language Specification and the Java Virtual Machine Specification and provides the Standard Edition of the Java Application Programming
JFrame	JFrame class is a type of container which inherits the java. awt. Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.
Java	Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

AN INTRODUCTION

Topicality. In our world today, computers are used in many fields such as business, industry, administration, education, and in various aspects of our lives, and this is only done by using one of the programming languages.

A programming language is a set of symbols, grammar, and rules with the help of which one is able to translate algorithms to programs that will be executed by the computer. The programmer communicates with a machine using programming languages. Most of the programs have a highly structured set of rules. The primary classifications of programming languages are Machine Languages. Assembly Languages. High-level Languages.

In this work grade, I have implemented a simulator on the topic "Array" in the Java programming language.

Arrays in java are used to store multiple values in a single variable, instead of declaring separate variables for each value.

Java array is an object which contains elements of a similar data type. Additionally, the elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on the 1st index, and so on.

In Java, an array is an object of a dynamically generated class. Java array inherits the Object class and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimentional or multidimentional arrays in Java.

Moreover, Java provides the feature of anonymous arrays which is not available in C/C++.

For the first task in this grade work, I will initially study arrays in Java, clarify their types and methods of declaring them, and explain how to enter the values of different types inside them, and I will explain their advantages and disadvantages.

For the second task in this grade work, I will design and write a program in the Java language that implements a scientific calculator, and I will use the array to store the values that the user will enter and also to store the values resulting from the calculations executed through the scientific calculator.

In the era of globalization, we used Calculator so many times. We use it to do the calculation in a short time. In this perspective, I have made a java Scientific Calculator software. By using this software, we can easily calculate our mathematical problem.

The purpose of the Grade work. The purpose of the grade work is development of software with application of arrays in Java programming language.

The object of the Grade work is process of professional education of students majoring in Computer Sciences.

The subject of the Grade work is software with application of arrays in java programming language.

The aim of the simulator is for the simulator to be used as training aid to help students understand the topic of Array in java language.

The following methods for the development have been used:

- NetBeans Platform including JDK.
- JFrame container.
- java language.
- java Swing.

Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The target of Java is to write a program once and then run this program on multiple operating systems.

The first publicly available version of Java (Java 1.0) was released in 1995 Sun Microsystems was acquired by the Oracle Corporation in 2010. Oracle has now the steermanship for Java In 2006 Sun started to make Java available under the GNU General Public License (GPL). Oracle continues this project called Open JDK. Over time new enhanced versions of Java have been released .The current version of Java is Java 17; Java is defined by a specification and consists of a programming language, a compiler, core libraries, and a runtime (Java virtual machine). The Java runtime allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries. The Java language was designed with the following properties:

Platform independent: Java programs use the Java virtual machine as an abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program (which is standard-compliant and follows certain rules) can run unmodified on all supported platforms, e.g., Windows or Linux. Object-orientated programming language: Except for the primitive data types, all elements in Java are objects. Strongly-typed programming language: Java is strongly-typed, e.g., the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g., must be done in most cases by the programmer. Interpreted and compiled language: Java source code is transferred into the bytecode format which does not depend on the target platform. These bytecode instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so-called Hotspot-Compiler which translates performance-critical bytecode instructions into native code instructions. Automatic memory management: Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector automatically deletes objects to which no active pointer exists.

This Grade work will study one of the most important topics of the Java language, which is: In Java, an Array is an object instantiated from a dynamically

generated class. Internally, an array in Java is a set of variables referenced by using a single variable name combined with an index number. Each item of an array is an element. All the elements in an array must be of the same type. An int array can contain int values, for example, and a String array can contain strings, Java array implements Serializable and Cloneable interfaces. Java array also has an object class as its parent class. You can store built-in and derived type values in Java arrays. The list of methods used is the use of Java software. The Grade work uses the NetBeans IDE visual development environment and the object-oriented Java programming language. The explanatory note to the Grade work consists of three sections, the Array statement, the purpose of the Array, Array list. The structure is built so that it allows a logical presentation of the material and the disclosure of the topic of work.

The volume of the explanatory note: 87 pages, incl. The main part is 40 pages, 18 sources.

CHAPTER 1. PROBLEM STATEMENT

1.1. Problem Statement

Create a desktop application written in Java using the NetBeans platform, which is a scientific calculator that helps students to study and do arithmetic and mathematical work. At the same time, this application is a practical example of Where the problem-solving principle was applied and entering values by the user and is stored These entries are in the Array, and the user interface has been designed uncomplicatedly so that it is used effectively, and the ready-made functions of the Math class, and JFrame were used, in the design of the user interface, and this application may be developed in the future.

1.2. Information Review

Review of simulators with similar topics and tasks

From the electronic archive of the North East University Bangladesh, I select work that has a similar subject, task, and programmatic implementation to the future project.

According to the following parameters, one Project Report was found:

Course Title: Project Work I, Course Code: CSE-200, Project Report created by Md. Abdul Mutalib .

The Project Feature: In his software, I have designed 5 five features.

Which are

- Standard Mode
- Scientific Mode
- Math Quiz
- Currency Converter
- Play Game

Tools/Software used: To build his software, he has used Apache NetBeans IDE 12.2 software. On the other hand, I have used Java language along with Java GUI.

Details of his work with the necessary screenshot:

As he mentioned before, he has designed five features for his project. Now he has written the details of his project. This is the Standard Mode of his Scientific Calculator. Wherein we can calculate normal mathematical equations. Like, addition, subtraction, multiplication, division, and square root. On the run time of the calculator, we can clear all the previous calculations by pressing the c button. Likewise, it also possible to delete a single digit from the calculator by pressing the left arrow (←) sign button.

The following image is the first feature of his software

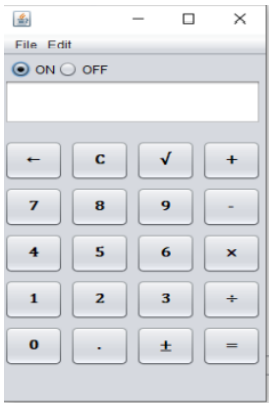

Figure 1.1. the Standard Mode of his Scientific Calculator.

The following image is the second feature of his software. By using Scientific Mode, we can get the value of Sin, Cos, Tan, and others. This feature is basically for doing Scientific mathematical operations.

Figure 1.2. The Standard Mode of his Scientific Calculator.

This is the 3rd feature of his software. Here a user can take a math quiz.

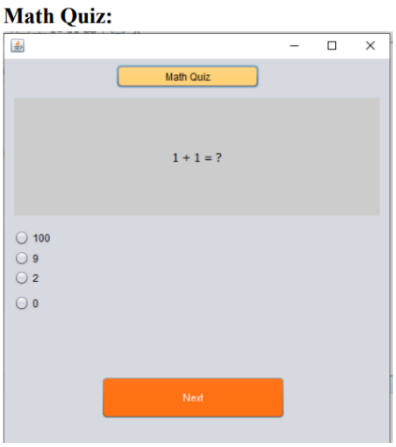

Figure 1.3. The starting page of a math quiz

At the end of quiz, user will get the score of the quiz. User can restart the quiz.

Figure 1.4. The score of the quiz

This is the 4 th feature of his project (Currency Converter). Here user can convert a specific amount of money to one currency to another currency. In this screenshot he has shown a demo. Like The amount is: 256 USD to BD Taka will be 21818.88.

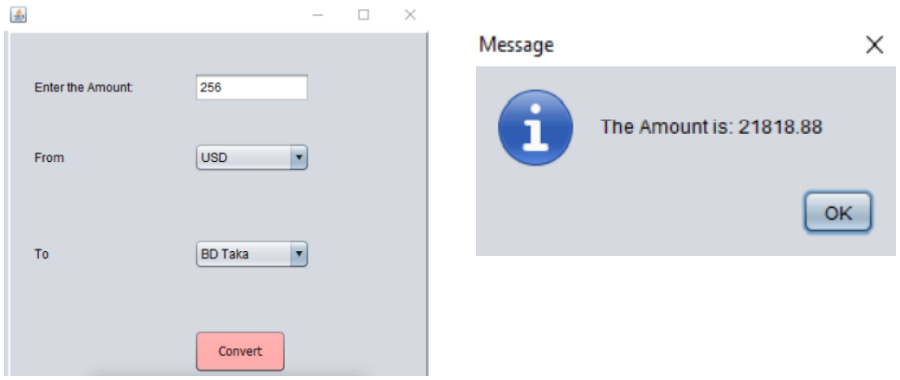

Figure 1.5. The Currency Converter

At last, he has added a minor game feature to my project. A user can play the game whenever he wants. There will be nine boxes to fill up. Here he has added a screenshot of this game.

Figure 1.6. The play of game

Challenges that he faced to do this project:

Basically, in this project he has faced some issues with completing this project. Most of the time, he faced implementing logic in coding. There are so many classes in the Java language that he is not a well-known student to implement in the project. These kinds of issues he has faced in this project.

Results:

In his project, all the feature works perfectly except Tik Tac Toe game. There are some bugs that he couldn't fix the last time.

CHAPTER 2. THEORETICAL PART

2.1. Array statement

For the distance course "Programming II" you need to develop a simulator. The main points of the work:

- describe the Java language;
- give the basic concepts of the topic Array;
- develop an array for the simulator and make a block diagram;
- describe the programming language and technologies used in developing the program.

2.2. The purposes of an array

What are the Arrays

The Arrays hold sequences of values of the same type. the following sections show how to work with arrays in Java.

Declaring Arrays

an array is a data structure that stores a collection of values of the same type. We can access each individual value through an integer index. For example, if a is an array of integers, then a[i] is the i^{th} integer in the array.

Declare an array variable by specifying the array type—which is the element type followed by []—and the array variable name. For example, here is the declaration of an array a of integers:

```
int[] a;
int[] a= new int[100];
```

However, this statement only declares the variable a. It does not yet initialize a with an actual array. Use the new operator to create the array. This statement declares and initializes an array of 100 integers. The array length need not be a constant: new int[n] creates an array of length n. Once we create an array, we cannot change its length (although we can, of course, change an individual array element). If we

frequently need to expand the length of arrays while your program is running, we should use array lists . 🗎 Note

we can define an array variable either as

int[] a;

or as

int a[];

Most Java programmers prefer the former style because it neatly separates the type int[] (integer array) from the variable name.

Java has a shortcut for creating an array object and supplying initial values:

int[] smallPrimes = { 2, 3, 5, 7, 11, 13 };

Notice that we do not use new with this syntax, and we don’t specify the length. A comma after the last value is allowed, which can be convenient for an array to which we keep adding values over time:

// add more names here and put a comma after each name

String[] authors = { "James Gosling", "Bill Joy", "Guy Steele", };

we can declare an anonymous array:

new int[] { 17, 19, 23, 29, 31, 37 }

This expression allocates a new array and fills it with the values inside the braces. It counts the number of initial values and sets the array size accordingly. we can use this syntax to reinitialize an array without creating a new variable. For example:

smallPrimes = new int[] { 17, 19, 23, 29, 31, 37 };

is shorthand for :

int[] anonymous = { 17, 19, 23, 29, 31, 37 };

smallPrimes = anonymous;

🗎 Note

It is legal to have arrays of length 0. Such an array can be useful if we write a method that computes an array result and the result happens to be empty. Construct an array of length 0 as

new elementType[0]

or new

elementType[] {}

Note that an array of length 0 is not the same as null.

Accessing Array Elements

When we create an array of numbers, all elements are initialized with zero.

Arrays of boolean are initialized with false. Arrays of objects are initialized with the special value null, which indicates that they do not (yet) hold any objects.

This can be surprising for beginners. For example,

String[] names = new String[10];

creates an array of ten strings, all of which are null. If we want the array to hold empty strings, we must supply them:

String[] names = new String[10];

for (int i = 0; i < 10; i++) names[i] = "";

💣 Caution

If we construct an array with 100 elements and then try to access the element a[100] (or any other index outside the range from 0 to 99),

an “array index out of bounds” exception will occur.

To find the number of elements of an array, use array.length. For example:

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

System.out.println(a[i]);

The “for each” Loop

Java has a powerful looping construct that allows we to loop through each element in an array (or any other collection of elements) without having to fuss with index values. The enhanced for loop

for (variable : collection) statement

sets the given variable to each element of the collection and then executes the statement (which, of course, may be a block).

Array Copying

we can copy one array variable into another, but then both variables refer to the same array:

int[] luckyNumbers = smallPrimes;

luckyNumbers[5] = 12; // now smallPrimes[5] is also 12

Figure 1.1 shows the result. If we actually want to copy all values of one array into a new array, use the copyOf method in the Arrays class:

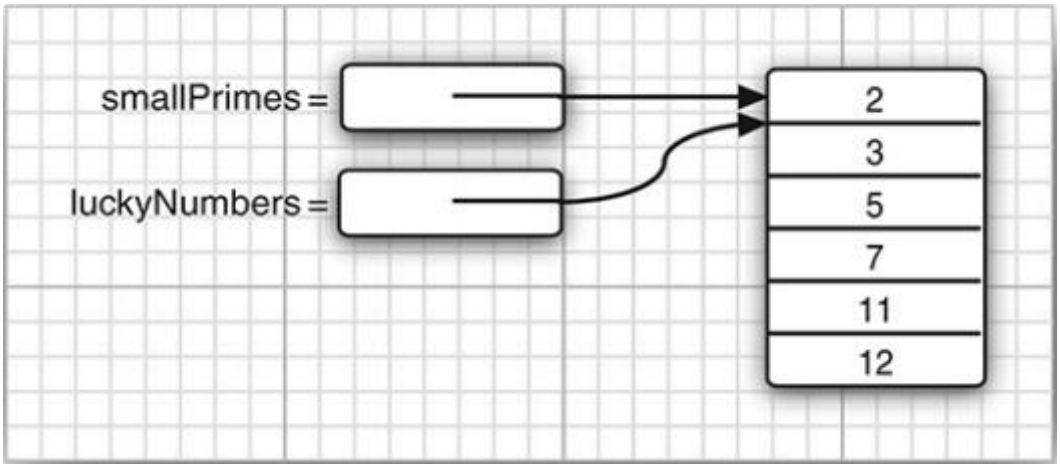

Figure 1.1. copying an array variable

int[] copiedLuckyNumbers = Arrays.copyOf(luckyNumbers, luckyNumbers.length);

The second parameter is the length of the new array. A common use of this method is to increase the size of an array:

luckyNumbers = Arrays.copyOf(luckyNumbers, 2 * luckyNumbers.length);

The additional elements are filled with 0 if the array contains numbers, false if the array contains boolean values. Conversely, if the length is less than the length of the original array, only the initial values are copied.

Command-Line Parameters

java.util.Arrays
• static String toString(xxx[] a) 5 • returns a string with the elements of a, enclosed in brackets and delimited by commas. In this and the following methods, the component type xxx of the array can be int, long, short, char, byte, boolean, float, or double. • static xxx[] copyOf(xxx[] a, int end) • static xxx[] copyOfRange(xxx[] a, int start, int end) returns an array of the same type as a, of length either end or end – start, filled with the values of a. If end is larger than a.length, the result is padded with 0 or false values. static void sort(xxx[] a) sorts the array, using a tuned QuickSort algorithm. static int binarySearch(xxx[] a, xxx v) static int binarySearch(xxx[] a, int start, int end, xxx v) uses the binary search algorithm to search for the value v in the sorted array a. If v is found, its index is returned. Otherwise, a negative value r is returned; –r – 1 is the spot at which v should be inserted to keep a sorted. static void fill(xxx[] a, xxx v) sets all elements of the array to v. static boolean equals(xxx[] a, xxx[] b) returns true if the arrays have the same length and if the elements at corresponding indexes match.

Multidimensional Arrays (matrix)

Multidimensional arrays use more than one index to access array elements. They are used for tables and other more complex arrangements. Declaring a two-dimensional array in Java is simple enough. For example:

double[][] balances;

we cannot use the array until you initialize it.

In this case, you can do the initialization as follows:

balances = new double[NYEARS][NRATES];

In other cases, if we know the array elements, we can use a shorthand notation for initializing a multidimensional array without a call to new. For example:

int[][] magicSquare = { {16, 3, 2, 13}, {5, 10, 11, 8}, {9, 6, 7, 12}, {4, 15, 14, 1} };

Once the array is initialized, we can access individual elements by supplying two pairs of brackets—for example, balances[i][j]. The example program stores a one-dimensional array interest of interest rates and a two-dimensional array balances of account balances, one for each year and interest rate. We initialize the first row of the array with the initial balance:

for (int j = 0; j < balances[0].length; j++)

balances[0][j] = 10000;

Then we compute the other rows, as follows:

for (int i = 1; i < balances.length; i++)

{ for (int j = 0; j < balances[i].length; j++)

{ double oldBalance = balances[i - 1][j]; double interest = . . .; balances[i][j] = oldBalance + interest; } }

shows the full program.

Ragged Arrays

java has no multidimensional arrays at all, only one-dimensional arrays. Multidimensional arrays are faked as “arrays of arrays.” For example, the balances array in the preceding example is actually an array that contains ten elements, each of which is an array of six floating-point numbers (Figure 2.2).

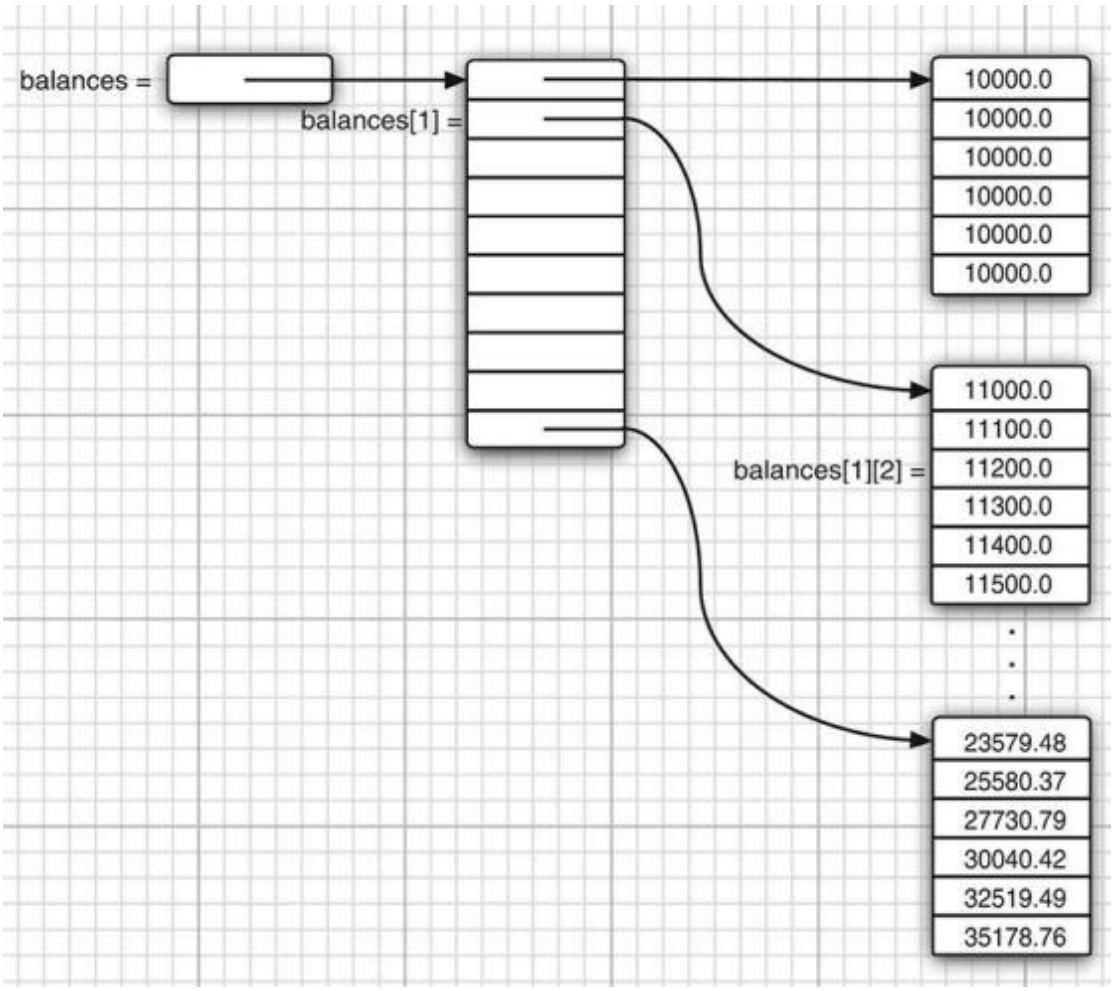

Figure 1.2 A two-dimensional array

The expression balances[i] refers to the ith subarray—that is, the ith row of the table.

It is itself an array, and balances[i][j] refers to the jth element of that array. Since rows of arrays are individually accessible, you can actually swap them!

```
double[] temp = balances[i];
balances[i] = balances[i + 1];
balances[i + 1] = temp;
```

2.3. Array lists

Review of Generic Array Lists

ArrayList class implements List interface and it is based on an Array data structure. It is widely used because of the functionality and flexibility it offers. Most

of the developers choose ArrayList over Array as it's a very good alternative of traditional java arrays. ArrayList is a resizable-array implementation of the interface. It implements all optional list operations, and permits all elements, including. The List interface describes an ordered collection in which the position of elements matters. There are two protocols for visiting the elements: through an iterator and by random access with methods get and set. The latter is not appropriate for linked lists, but of course get and set make a lot of sense for arrays. The collections library supplies the familiar ArrayList class that also implements the List interface. An ArrayList encapsulates a dynamically reallocated array of objects.

Why Array List is better than Array?

The limitation with array is that it has a fixed length so if it is full, you cannot add any more elements to it, likewise if there are number of elements gets removed from it the memory consumption would be the same as it doesn't shrink. On the other ArrayList can dynamically grow and shrink after addition and removal of elements. Apart from these benefits ArrayList class enables us to use predefined methods of it which makes our task easy. Let's see the diagrams to understand the addition and removal of elements from ArrayList and then we will see the programs.

Note

If we are a veteran Java programmer, we may have used the Vector class whenever you need a dynamic array.

Why use an ArrayList instead of a Vector?

For one simple reason: All methods of the Vector class are synchronized. It is safe to access a Vector object from two threads. But if we access a vector from only a single thread - by far the more common case - our code wastes quite a bit of time with synchronization. In contrast, the ArrayList methods are not synchronized. We recommend that you use an ArrayList instead of a Vector whenever you don't need synchronization.we can set the size of an array at runtime.

```
int actualSize = . . .;
var staff = new Employee[actualSize];
```

Of course, this code does not completely solve the problem of dynamically modifying arrays at runtime. Once you set the array size, you cannot change it easily. Instead, in Java you can deal with this common situation by using another Java class, called ArrayList. The ArrayList class is similar to an array, but it automatically adjusts its capacity as you add and remove elements, without any additional code. ArrayList is a generic class with a type parameter. To specify the type of the element objects that the array list holds, you append a class name enclosed in angle brackets, such as ArrayList<Employee>.

The following sections show you how to work with array lists.

Declaring Array Lists:

Here is how to declare and construct an array list that holds Employee objects:

ArrayList staff = new ArrayList();

As of Java 10, it is a good idea to use the var keyword to avoid duplicating the class name:

var staff = new ArrayList();

it we don't use the var keyword, you can omit the type parameter on the righthand side:

ArrayList staff = new ArrayList<>();

This is called the "diamond" syntax because the empty brackets <> resemble a diamond.

Use the diamond syntax together with the new operator. The compiler checks what happens to the new value. If it is assigned to a variable, passed into a method, or returned from a method, then the compiler checks the generic type of the variable, parameter, or method. It then places that type into the <>. In our example, the new ArrayList<>() is assigned to a variable of type ArrayList. Therefore, the generic type is Employee.

💣 Caution

If we declare an ArrayList with var, do not use the diamond syntax. The declaration

var elements = new ArrayList<>();

yields an ArrayList

🗎 Note

Before Java 5, there were no generic classes. Instead, there was a single ArrayList class, a one-size-fits-all collection holding elements of type Object.we can still use ArrayList without a suffix. It is considered a “raw” type, with the type parameter erased.

In even older versions of Java, programmers used the Vector class for dynamic arrays. However, the ArrayList class is more efficient, and there is no longer any good reason to use the Vector class.

Use the add method to add new elements to an array list. For example, here is how you populate an array list with Employee objects:

staff.add(new Employee("Harry Hacker", . . .));

staff.add(new Employee("Tony Tester", . . .));

The array list manages an internal array of object references. Eventually, that array will run out of space. This is where array lists work their magic: If you call add and the internal array is full, the array list automatically creates a bigger array and copies all the objects from the smaller to the bigger array.If we already know, or have a good guess, how many elements you want to store, call the ensureCapacity method before filling the array list:

staff.ensureCapacity(100);

That call allocates an internal array of 100 objects.

Then, the first 100 calls to add will not involve any costly reallocation.

You can also pass an initial capacity to the ArrayList constructor:

ArrayList staff = new ArrayList<>(100);

💣 Caution

Allocating an array list as

new ArrayList<>(100) // capacity is 100

is not the same as allocating a new array as

new Employee[100] // size is 100

There is an important distinction between the capacity of an array list and the size of an array. If you allocate an array with 100 entries, then the array has 100 slots, ready for use.An array list with a capacity of 100 elements has the potential of holding 100 elements (and, in fact, more than 100, at the cost of additional reallocations)—but at the beginning, even after its initial construction, an array list holds no elements at all.The size method returns the actual number of elements in the array list. For example,

staff.size()

returns the current number of elements in the staff array list.

This is the equivalent of

a.length

for an array a.

Once you are reasonably sure that the array list is at its permanent size, you can call the trimToSize method. This method adjusts the size of the memory block to use exactly as much storage space as is required to hold the current number of elements.

The garbage collector will reclaim any excess memory. Once you trim the size of an array list, adding new elements will move the block again, which takes time. we should only use trimToSize when you are sure you won’t add any more elements to the array list.

java.util.ArrayList
• ArrayList() constructs an empty array list. • ArrayList(int initialCapacity) constructs an empty array list with the specified capacity. • boolean add(E obj) appends obj at the end of the array list. Always returns true. • int size() returns the number of elements currently stored in the array list. (Of course,

this is never larger than the array list's capacity).

- void ensureCapacity(int capacity)
  ensures that the array list has the capacity to store the given number of elements without reallocating its internal storage array.
- void trimToSize()
  reduces the storage capacity of the array list to its current size.

Accessing Array List Elements

Unfortunately, nothing comes for free. The automatic growth convenience of array lists requires a more complicated syntax for accessing the elements. The reason is that the ArrayList class is not a part of the Java programming language; it is just a utility class programmed by someone and supplied in the standard library. Instead of the pleasant [] syntax to access or change the element of an array, you use the get and set methods. For example, to set the ith element, use

staff.set(i, harry);

This is equivalent to

a[i] = harry;

for an array a.(As with arrays, the index values are zero-based.)

💣 Caution

Do not call list.set(i, x) until the size of the array list is larger than i. For example, the following code is wrong:

var list = new ArrayList(100);

// capacity 100,

size 0 list.set(0, x);

// no element 0 yet

Use the add method instead of set to fill up an array, and use set only to replace a previously added element.

To get an array list element, use

Employee e = staff.get(i);

This is equivalent to

Employee e = a[i];

Note

When there were no generic classes, the get method of the raw ArrayList class had no choice but to return an Object. Consequently, callers of get had to cast the returned value to the desired type:

Employee e = (Employee) staff.get(i);

The raw ArrayList is also a bit dangerous. Its add and set methods accept objects of any type.

A call

staff.set(i, "Harry Hacker");

compiles without so much as a warning, and you run into grief only when you retrieve the object and try to cast it.

If you use an ArrayList instead, the compiler will detect this error.

we can sometimes get the best of both worlds—flexible growth and convenient element access—with the following trick. First, make an array list and add all the elements:

var list = new ArrayList();

while (. . .)

{ x = . . .; list.add(x); }

When we are done, use the toArray method to copy the elements into an array:

var a = new X[list.size()];

list.toArray(a);

Sometimes, you need to add elements in the middle of an array list. Use the add method with an index parameter:

int n = staff.size() / 2; staff.add(n, e);

The elements at locations n and above are shifted up to make room for the new entry.

If the new size of the array list after the insertion exceeds the capacity, the array list reallocates its storage array. Similarly, you can remove an element from the middle of an array list:

Employee e = staff.remove(n);

The elements located above it are copied down, and the size of the array is reduced by one. Inserting and removing elements is not terribly efficient.

It is probably not worth worrying about for small array lists. But if you store many elements and frequently insert and remove in the middle of a collection, consider using a linked list instead.

```
for (Employee e : staff)
do something with e
This loop has the same effect as
for (int i = 0; i < staff.size(); i++)
{ Employee e = staff.get(i);
do something with e }
```

Below is a code using ArrayList :

```
ArrayList<String> list=new ArrayList<String>(); //Creating arraylist
 list.add("Mango");//Adding object in arraylist
 list.add("Apple");
  list.add("Banana");
  list.add("Grapes");
  //Printing the arraylist object
   System.out.println(list);
```

java.util.ArrayList
• E set(int index, E obj) puts the value obj in the array list at the specified index, returning the previous contents. • E get(int index) gets the value stored at a specified index.

- void add(int index, E obj)
  shifts up elements to insert obj at the specified index.
- E remove(int index)
  removes the element at the given index and shifts down all elements above it. The removed element is returned.

Compatibility between Typed and Raw Array Lists

In this section, we will see how to interoperate with legacy code that does not use type parameters.

public class EmployeeDB {

public void update(ArrayList list) { . . . }

public ArrayList find(String query) { . . . } }

we can pass a typed array list to the update method without any casts.

ArrayList staff = . . .;

employeeDB.update(staff);

The staff object is simply passed to the update method.

💣 Caution

Even though we get no error or warning from the compiler, this call is not completely safe. The update method might add elements into the array list that are not of type Employee. When these elements are retrieved, an exception occurs. This sounds scary, but if you think about it, the behavior is simply as it was before generics were added to Java. The integrity of the virtual machine is never jeopardized. In this situation, you do not lose security, but you also do not benefit from the compile-time checks.

Conversely, when we assign a raw ArrayList to a typed one, we get a warning.

ArrayList result = employeeDB.find(query); // yields warning

🗎 **Note**

To see the text of the warning, compile with the option - Xlint:unchecked.

Using a cast does not make the warning go away. ArrayList result = (ArrayList) employeeDB.find(query); // yields another warning

Instead, we get a different warning, telling we that the cast is misleading. This is the consequence of a somewhat unfortunate limitation of generic types in Java. For compatibility, the compiler translates all typed array lists into raw ArrayList objects after checking that the type rules were not violated. In a running program, all array lists are the same—there are no type parameters in the virtual machine. Thus, the casts (ArrayList) and (ArrayList) carry out identical runtime checks. There isn't much we can do about that situation. When we interact with legacy code, study the compiler warnings and satisfy yourself that the warnings are not serious.

CHAPTER 3. PRACTICAL PART

3.1. NetBeans in Java

NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5,[3] and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactoring, version control (supporting CVS, Subversion, Git, Mercurial and Clearcase).

Modularity: All the functions of the IDE are provided by modules. Each module provides a well-defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules.

NetBeans Profiler

The NetBeans Profiler is a tool for the monitoring of Java applications: It helps developers find memory leaks and optimize speed. Formerly downloaded separately, it is integrated into the core IDE since version 6.0. The Profiler is based on a Sun Laboratories research project that was named JFluid. That research uncovered specific techniques that can be used to lower the overhead of profiling a Java application. One of those techniques is dynamic bytecode instrumentation, which is particularly useful for profiling large Java applications. Using dynamic bytecode instrumentation and additional algorithms, the NetBeans Profiler is able to obtain

runtime information on applications that are too large or complex for other profilers. NetBeans also support Profiling Points that let you profile precise points of execution and measure execution time.

GUI design tool

NetBeans GUI Builder

Formerly known as project Matisse, the GUI design-tool enables developers to prototype and design Swing GUIs by dragging and positioning GUI components.

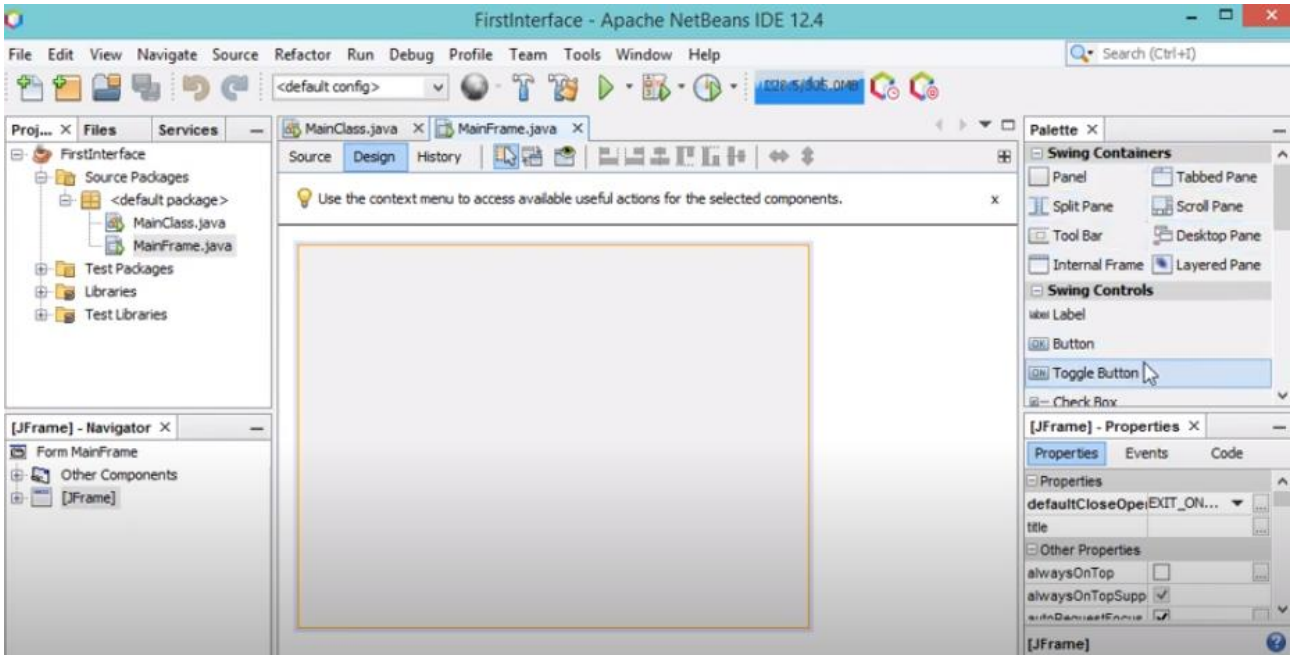

Figure 2.1. NetBeans GUI Builder

3.2. Java Swing

Java Swing is a software platform

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying

platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent.

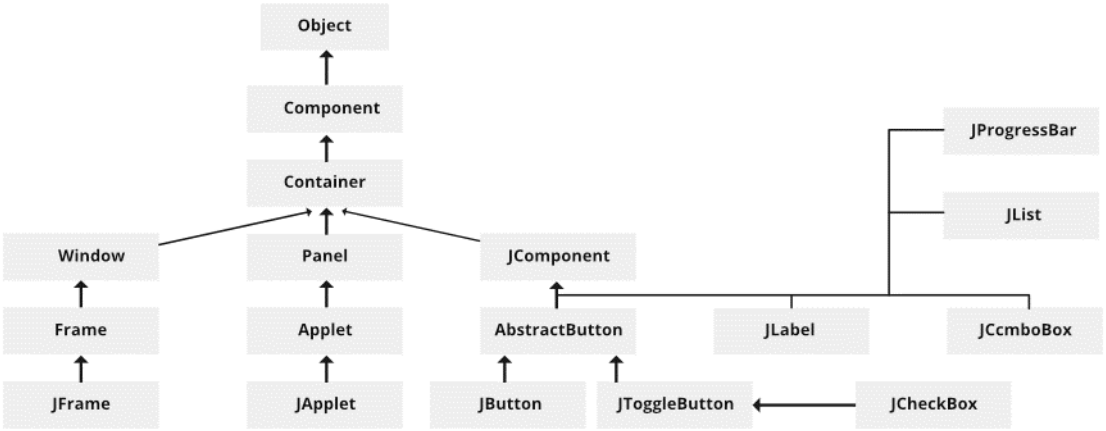

Figure 2.2. Swing Classes Hierarchy

3.3. Examples of Application of Array in Java

The array elements are numbered from 0 to 99 (and not 1 to 100). Once the array is created, you can fill the elements in an array, for example, by using a loop:
Example 1.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
int[] a = new int[100];
for (int i = 0; i < 100; i++)
{   a[i] = i;       // fills the array with numbers 0 to 99
System.out.println(a[i]);
```

```
}} // end of main()
    }// end of MainClass
```

The collection expression must be an array or an object of a class that implements the Iterable interface, such as ArrayList. For example,

Example 2.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
int[] a = { 17, 19, 23, 29, 31, 37 };
for (int element : a)
System.out.println(element);
} // end of main()
    }// end of MainClass
```

prints each element of the array a on a separate line. we should read this loop as “for each element in a”. The designers of the Java language considered using keywords, such as foreach and in. But this loop was a late addition to the Java language, and in the end nobody wanted to break the old code that already contained methods or variables with these names (such as System.in). Of course, we could achieve the same effect with a traditional for loop:

Example 3.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
int[] a = { 1, 2, 3, 4, 5, 6 };
for (int i = 0; i < a.length; i++)
System.out.println(a[i]);
} // end of main()
}// end of MainClass
```

However, the “for each” loop is more concise and less error-prone, as we don’t have to worry about those pesky start and end index values.

🗎 Note

The loop variable of the “for each” loop traverses the elements of the array, not the index values.

The “for each” loop is a pleasant improvement over the traditional loop if you need to process all elements in a collection. However, there are still plenty of opportunities to use the traditional for loop. For example, you might not want to traverse the entire collection, or you may need the index value inside the loop.

✔ Tip

There is an even easier way to print all values of an array, using the toString method of the Arrays class. The call Arrays.toString(a) returns a string containing the array elements, enclosed in brackets and separated by commas, such as "[2, 3, 5, 7, 11, 13]". To print the array, simply call import java.util.Arrays;

Example 4.

```
package JavaProgrammingExample;
import java.util.Arrays;
public class MainClass {
public static void main(String[] args) {
int[] a = { 1, 2, 3, 4, 5, 6 };
System.out.println(Arrays.toString(a));
} // end of main()
}// end of MainClass
```

we have already seen one example of a Java array repeated quite a few times. Every Java program has a main method with a String[] args parameter. This parameter indicates that the main method receives an array of strings— namely, the arguments specified on the command line.

For example, consider this program:

Example 5.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
if (args.length == 0 || args[0].equals("-h"))
System.out.print("Hello,");
else if (args[0].equals("-g"))
System.out.print("Goodbye,"); // print the other command-line arguments
for (int i = 1; i < args.length; i++)
System.out.print(" " + args[i]); System.out.println("!");
} // end of main()
    }// end of MainClass
```

If the program is called as java Message -g cruel world

then the args array has the following contents:

args[0]: "-g"

args[1]: "cruel"

args[2]: "world"

The program prints the message

Goodbye, cruel world!

Array Sorting

To sort an array of numbers, you can use one of the sort methods in the Arrays class:

Example 6.

```
package JavaProgrammingExample;
import java.util.Arrays;
public class MainClass {
public static void main(String[] args) {
int[] a = { 6, 2, 1, 4, 3, 0};
Arrays.sort(a);
} // end of main()
```

```
}// end of MainClass
```

This method uses a tuned version of the QuickSort algorithm that is claimed to be very efficient on most data sets. The Arrays class provides several other convenience methods for arrays that are included in the API notes at the end of this section.

Example 7.

```
package JavaProgrammingExample;
import java.util.Arrays;
public class MainClass {
public static void main(String[] args) {
int[] a = { 6, 2, 1, 4, 3, 0};
   System.out.println(Arrays.toString(a));
} // end of main()
}// end of MainClass
```

Example 8.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
/*  This program shows how to store tabular data in a 2D array.
@version 1.40 2004-02-10
@author Cay Horstmann */
final double STARTRATE = 10;
final int NRATES = 6;
final int NYEARS = 10;
//set interest rates to 10 . . . 15%
double[] interestRate = new double[NRATES];
for (int j = 0; j < interestRate.length; j++)
interestRate[j] = (STARTRATE + j) / 100.0;
double[][] balances = new double[NYEARS][NRATES];
```

```
// set initial balances to 10000
for (int j = 0; j < balances[0].length; j++)
balances[0][j] = 10000;
// compute interest for future years
for (int i = 1; i < balances.length; i++)
{ for (int j = 0; j < balances[i].length; j++)
{ // get last year's balances from previous row
double oldBalance = balances[i - 1][j];
// compute interest
double interest = oldBalance * interestRate[j];
// compute this year's balances
balances[i][j] = oldBalance + interest;
}}
// print one row of interest rates
for (int j = 0; j < interestRate.length; j++)
System.out.printf("%9.0f%%", 100 * interestRate[j]);
System.out.println();
// print balance table
for (double[] row : balances)
{ // print table row
for (double b : row)
System.out.printf("%10.2f", b);
System.out.println();}
} // end of main()
        }// end of MainClass
```

Note

A “for each” loop does not automatically loop through all elements in a two-dimensional array.

Instead, it loops through the rows, which are themselves one-dimensional arrays.

To visit all elements of a two dimensional array a, nest two loops, like this:

```
for (double[] row : a)
for (double value : row)
{do something with value …..};
```

✔ Tip

To print out a quick-and-dirty list of the elements of a two-dimensional array, call

```
int[][] a = { {16, 3, 2, 13}, {5, 10, 11, 8}, {9, 6, 7, 12}, {4, 15, 14, 1} };
System.out.println(Arrays.deepToString(a));
```

The output is formatted like this:

```
[[16, 3, 2, 13], [5, 10, 11, 8], [9, 6, 7, 12], [4, 15, 14, 1]]
```

It is also easy to make “ragged” arrays—that is, arrays in which different rows have different lengths. Here is the standard example.

Let us make an array in which the element at row i and column j equals the number of possible outcomes of a “choose j numbers from i numbers” lottery.

Example 9.

```
package JavaProgrammingExample;
public class MainClass {
public static void main(String[] args) {
int twoD[][] = new int[4][];
twoD[0] = new int[1];
twoD[1] = new int[2];
twoD[2] = new int[3];
twoD[3] = new int[4];
int i, j, k = 0;
for(i=0; i<4; i++){
for(j=0; j<i+1; j++) {
twoD[i][j] = k;
```

```
k++; } }
for(i=0; i<4; i++) {
for(j=0; j<i+1; j++){
System.out.print(twoD[i][j] + " ");}
System.out.println();}
} // end of main()
}// end of MainClass
```

When we run this code, we will get the following result:

```
0
1 2
3 4 5
6 7 8 9
```

As j can never be larger than i, the matrix is triangular.

The ith row has i + 1 elements. (We allow choosing 0 elements; there is one way to make such a choice). To build this ragged array, first allocate the array holding the rows:

```
int[][] odds = new int[NMAX + 1][];
```

Next, allocate the rows:

```
for (int n = 0; n <= NMAX; n++)
odds[n] = new int[n + 1];
```

Now that the array is allocated, we can access the elements in the normal way, provided we do not overstep the bounds:

```
for (int n = 0; n < odds.length; n++)
for (int k = 0; k < odds[n].length; k++)
{ // compute lotteryOdds . . . odds[n][k] = lotteryOdds; }
```

Below is the code written in full:

Example 10.

```
package JavaProgrammingExample;
public class MainClass {
```

```
public static void main(String[] args) {
/* This program demonstrates a triangular array.
@version 1.20 2004-02-10
@author Cay Horstmann */
final int NMAX = 10;
// allocate triangular array
int[][] odds = new int[NMAX + 1][];
for (int n = 0; n <= NMAX; n++)
odds[n] = new int[n + 1];
// fill triangular array
for (int n = 0; n < odds.length; n++)
for (int k = 0; k < odds[n].length; k++)
{ /*compute binomial coefficient n*(n-1)*(n-2)*...*(n k+1)/(1*2*3*...*k)*/
int lotteryOdds = 1;
for (int i = 1; i <= k; i++)
lotteryOdds = lotteryOdds * (n - i + 1) / i;
odds[n][k] = lotteryOdds;}
// print triangular array
for (int[] row : odds){
for (int odd : row)
System.out.printf("%4d", odd);
System.out.println();}
} // end of main()
        }// end of MainClass
```

When we run this code, we will get the following result:

```
   1
   1   1
   1   2   1
   1   3   3   1
```

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

1 10 45 120 210 252 210 120 45 10 1

3.4. Design And Programming of The Scientific Calculator

Details of my work with the necessary screenshot.

As I mentioned before, I have designed two features for my project.

Project Feature: In my software, I have designed 2 two features. Which are

- Standard Mode
- Scientific Mode

Now I have written the details of my project.

the first feature Standard Mode of my Scientific Calculator. Wherein we can calculate normal mathematical equations. Like, addition, subtraction, multiplication, division, and square root.

On the run time of calculator, we can clear all the previous calculations by pressing CLEAR button.

the second feature of my Scientific Calculator. By using Scientific Mode, we can get the value of Sin, Cos, Tan, and others. This feature is basically for doing Scientific mathematical operations.

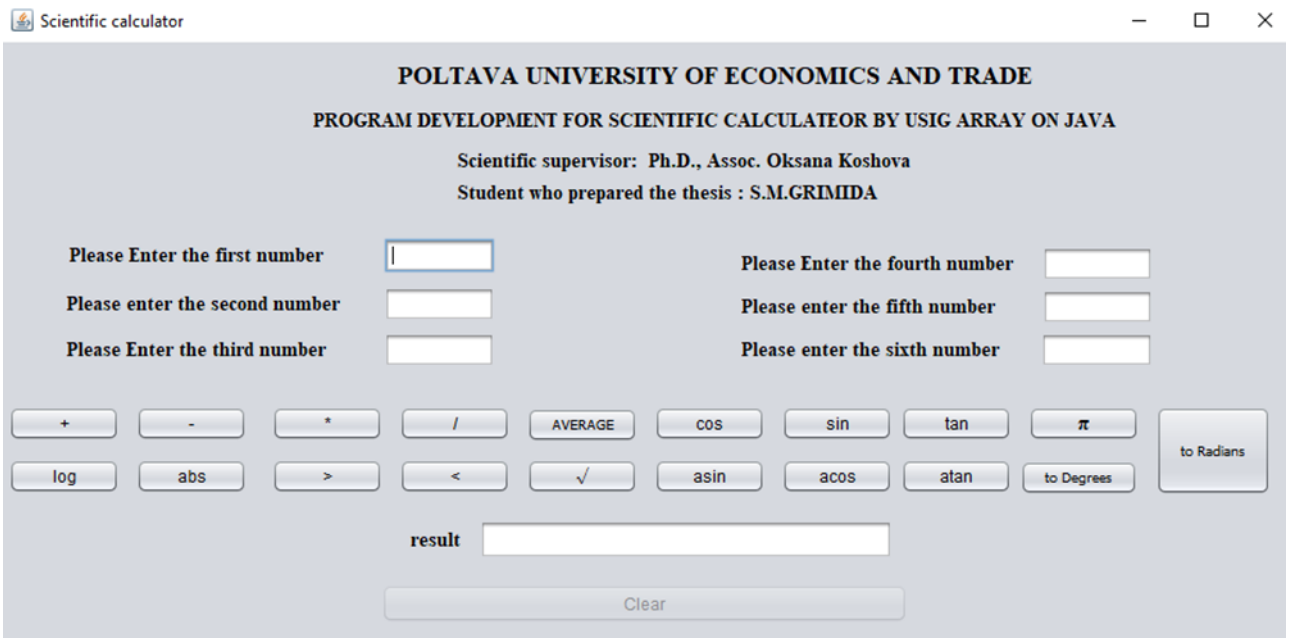

Figure 3.1. The graphical interface of the scientific calculator application

CONCLUSIONS

In that Grade work project, there was an independent consideration of different sources of information and mastering the project. Consolidation of theoretical knowledge that was provided in the lecture material I have studied at university.

For the solving of our first task in my grade work (with using of different sources of information):

- I have studied arrays in Java;
- I have clarified their types and methods of them declaring;
- I have explained how to enter the values of different types inside them
- I have explained their advantages and disadvantages.

For the solving of our second task in my grade work, I have designed and written a program that implements a scientific calculator using the Java language and has an array to store the values entered by the user and store the values resulting from the mathematical operations executed through the scientific calculator.

I have completed all the tasks that I set at the beginning of Grade work. And this is an excellent experience for me to do this project as one. This project can be expanded with more advanced features. If some automatic equation-solving function is added to the scientific calculator design, it will be more suitable for this project.

REFERENCES

1. CAY S. HORSTMANN, Core java volume I - fundamentals ELEVENTH EDITION, Copyright © 2019 Pearson Education Inc. ISBN-13: 978-0-13-516630-7.

2. A Brain-Friendly Guide to OOA&D, Head First Object-Oriented Analysis and Design, Copyright © 2007, Published by O'Reilly Media, Inc., Printing History: November 2006: First Edition. ISBN-978-0-596-00867-3.

3. A Wiley Brand, Java ALL-IN-ONE, 4th Edition, Published by: John Wiley & Sons, Inc., Copyright © 2014 by John Wiley & Sons, Inc., Hoboken, New Jersey ISBN 978-1-118-40803-2 (pbk); ISBN 978-1-118-41765-2 (ebk); ISBN 978-1-118-46206-5 (ebk); ISBN 978-1-118-61288-0 (ebk).

4. Iuliana Cosmina, Java for Absolute Beginners, Learn to Program the Fundamentals the Java 9+ Way, Copyright © 2018 by Iuliana Cosmina, ISBN-13 (pbk): 978-1-4842-3777-9, ISBN-13 (electronic): 978-1-4842-3778-6.

5. Maurice Naftalin and Philip Wadler, Java Generics and Collections, Copyright © 2007 O'Reilly Media . All rights reserved., Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, Printing History: October 2006: First Edition, ISBN: 978-0-596-52775-4.

6. EARLY OBJECTS, JAVA HOW TO PROGRAM, Copyright © 2015, 2012 and 2009 Pearson Education, Inc., ISBN-13: 978-0-13-380780-6.

7. Charlie Hunt, Binu John., Java Performance, Copyright © 2012 Oracle America, Inc., U.S. Corporate and Government Sales, first printing, September 2011, Includes bibliographical references and index, ISBN-13: 978-0-13-714252-1.

8. Scott Sanderson, Java For Beginners, (Written By A Software Engineer).

9. Cay S. Horstmann, Java SE 8 for the Really Impatient, Copyright © 2014 Pearson Education, Inc., ISBN-13: 978-0-321-92776-7, Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana, First printing, January 2014.

10. Herbert Schildt, Java the Complete Reference Eleventh Edition, Comprehensive coverage of the Java Language, Copyright © 2019 by McGraw-Hill Education (Publisher), ISBN: 978-1-26-044024-9.

11. Bruce Eckel, Thinking in Java, President, MindView, Inc., Fourth Edition, ISBN 0-13-187248-6 (pbk. : alk. paper), Copyright © 2006, First printing, January 2006.

12. Robert C. Martin Series, Clean Code A Handbook of Agile Software Craftsmanship, ISBN-13: 978-0-13-235088-4, Copyright © 2009 Pearson Education, Inc. First printing July, 2008.

13. Brian Goetz with Tim Peierls, Joshua Bloch, Joseph Bowbeer ,David Holmes and Doug Lea, Java Concurrency in Practice, Copyright © 2006 Pearson Education, Inc. , ISBN 0-321-34960-1, 9th Printing March 2010 , Copyright © 2006 Pearson Education, Inc.

14. Joshua Bloch, Effective Java, Third Edition, Copyright © 2018 Pearson Education Inc., Portions copyright © 2001-2008 Oracle and/or its affiliates, ISBN-13: 978-0-13-468599-1.

15. Herbert Schildt, Java A Beginner's Guide Eighth Edition, Copyright © 2019 by McGrawHill Education (Publisher), ISBN: 978¬1¬26¬044022¬5.

16. Scott Oaks, Java Performance: The Definitive Guide, Copyright © 2014 Scott Oaks. All rights reserved., Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., Editor: Meghan Blanchette, April 2014: First Edition, 2014-04-09: First release, ISBN: 978-1-449-35845-7.

17. CRAIG WALLS, Spring in Action, Fifth Edition, COVERS SPRING 5.0, ©2019 by Manning Publications Co. All rights reserved., ISBN: 9781617294945.

18. LASSE KOSKELA, Test Driven, PRACTICAL TDD AND ACCEPTANCE TDD FOR JAVA DEVELOPERS, ©2008 by Manning Publications Co., ISBN 1-932394-85-0.

APPENDIX A. Algorithm translation

Descriptive Algorithm.

Step 1. Start the Desktop application by launching a program NetBeans.

Step 2. The user will enter the numbers that he wants to apply the arithmetic operations to, in text filed.

Step 3. The user chooses the calculation he wants.

Step 4. The result will be displayed in text field titled result.

Step 5. The user can clear the entries he entered in the text field by pressing the clear button.

Step 6. End program.

APPENDIX B. Program code

```
public class NewJFrame1 extends javax.swing.JFrame {

    public NewJFrame1() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        number1 = new javax.swing.JTextField();
        number2 = new javax.swing.JTextField();
        sumButton = new javax.swing.JButton();
        subtractButton = new javax.swing.JButton();
        multiplyButton = new javax.swing.JButton();
        dividingButton = new javax.swing.JButton();
        averageButton = new javax.swing.JButton();
        resultButton = new javax.swing.JLabel();
```

```
number3 = new javax.swing.JTextField();
claerButton = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
number4 = new javax.swing.JTextField();
jLabel8 = new javax.swing.JLabel();
number5 = new javax.swing.JTextField();
jLabel9 = new javax.swing.JLabel();
number6 = new javax.swing.JTextField();
jLabel10 = new javax.swing.JLabel();
number7 = new javax.swing.JTextField();
cos = new javax.swing.JButton();
sin = new javax.swing.JButton();
tan = new javax.swing.JButton();
Iog = new javax.swing.JButton();
abs = new javax.swing.JButton();
greater_than = new javax.swing.JButton();
less_than = new javax.swing.JButton();
sqrt = new javax.swing.JButton();
asin = new javax.swing.JButton();
acos = new javax.swing.JButton();
atan = new javax.swing.JButton();
to_Degrees = new javax.swing.JButton();
to_Radians = new javax.swing.JButton();
jLabel6 = new javax.swing.JLabel();
PI = new javax.swing.JButton();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Scientific calculator");

        jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        jLabel1.setText("Please Enter the first number");
        jLabel1.setPreferredSize(new java.awt.Dimension(185, 25));

        jLabel2.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        jLabel2.setText("Please enter the second number");
        jLabel2.setPreferredSize(new java.awt.Dimension(185, 25));

        number1.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number1.setPreferredSize(new java.awt.Dimension(80, 25));
        number1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                number1ActionPerformed(evt);
            }
        });

        number2.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number2.setPreferredSize(new java.awt.Dimension(80, 25));
        number2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                number2ActionPerformed(evt);
```

```
    }
});

sumButton.setText("+");
sumButton.setMaximumSize(new java.awt.Dimension(40, 25));
sumButton.setPreferredSize(new java.awt.Dimension(80, 25));
sumButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        sumButtonActionPerformed(evt);
    }
});

subtractButton.setText("-");
subtractButton.setMaximumSize(new java.awt.Dimension(40, 25));
subtractButton.setPreferredSize(new java.awt.Dimension(80, 25));
subtractButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        subtractButtonActionPerformed(evt);
    }
});

multiplyButton.setText("*");
multiplyButton.setMaximumSize(new java.awt.Dimension(40, 25));
multiplyButton.setPreferredSize(new java.awt.Dimension(80, 25));
multiplyButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        multiplyButtonActionPerformed(evt);
    }
});
```

```
        dividingButton.setText("/");
        dividingButton.setMaximumSize(new java.awt.Dimension(40, 25));
        dividingButton.setPreferredSize(new java.awt.Dimension(80, 25));
        dividingButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                dividingButtonActionPerformed(evt);
            }
        });

        averageButton.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
        averageButton.setText("AVERAGE");
        averageButton.setMaximumSize(new java.awt.Dimension(40, 25));
        averageButton.setPreferredSize(new java.awt.Dimension(80, 25));
        averageButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                averageButtonActionPerformed(evt);
            }
        });

        resultButton.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N

resultButton.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        resultButton.setText("result");

        number3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                number3ActionPerformed(evt);
```

```
        }
    });

    claerButton.setText("Clear");
    claerButton.setEnabled(false);
    claerButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            claerButtonActionPerformed(evt);
        }
    });

    jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 18)); // NOI18N
    jLabel3.setText("POLTAVA UNIVERSITY OF ECONOMICS AND TRADE");

    jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
    jLabel4.setText("PROGRAM DEVELOPMENT FOR SCIENTIFIC CALCULATEOR BY USIG ARRAY ON JAVA");

    jLabel5.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
    jLabel5.setText("Student who prepared the thesis : S.M.GRIMIDA");

    jLabel7.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
    jLabel7.setText("Please Enter the third number");
    jLabel7.setPreferredSize(new java.awt.Dimension(185, 25));
```

```
        number4.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number4.setPreferredSize(new java.awt.Dimension(80, 25));
        number4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                number4ActionPerformed(evt);
            }
        });

        jLabel8.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        jLabel8.setText("Please Enter the fourth number");
        jLabel8.setPreferredSize(new java.awt.Dimension(185, 25));

        number5.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number5.setPreferredSize(new java.awt.Dimension(80, 25));

        jLabel9.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        jLabel9.setText("Please enter the fifth number");
        jLabel9.setPreferredSize(new java.awt.Dimension(185, 25));

        number6.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number6.setPreferredSize(new java.awt.Dimension(80, 25));
```

```
        jLabel10.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        jLabel10.setText("Please enter the sixth number");
        jLabel10.setPreferredSize(new java.awt.Dimension(185, 25));

        number7.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
        number7.setPreferredSize(new java.awt.Dimension(80, 25));

        cos.setText("cos");
        cos.setPreferredSize(new java.awt.Dimension(80, 25));
        cos.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cosActionPerformed(evt);
            }
        });

        sin.setText("sin");
        sin.setPreferredSize(new java.awt.Dimension(80, 25));
        sin.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sinActionPerformed(evt);
            }
        });

        tan.setText("tan");
        tan.setPreferredSize(new java.awt.Dimension(80, 25));
        tan.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        tanActionPerformed(evt);
    }
});

Iog.setText("Iog");
Iog.setPreferredSize(new java.awt.Dimension(80, 25));
Iog.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        IogActionPerformed(evt);
    }
});

abs.setText("abs");
abs.setPreferredSize(new java.awt.Dimension(80, 25));
abs.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        absActionPerformed(evt);
    }
});

greater_than.setText(">");
greater_than.setPreferredSize(new java.awt.Dimension(80, 25));
greater_than.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        greater_thanActionPerformed(evt);
    }
});

less_than.setText("<");
```

```
less_than.setPreferredSize(new java.awt.Dimension(80, 25));
less_than.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        less_thanActionPerformed(evt);
    }
});

sqrt.setText("âˆš");
sqrt.setPreferredSize(new java.awt.Dimension(80, 25));
sqrt.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        sqrtActionPerformed(evt);
    }
});

asin.setText("asin");
asin.setPreferredSize(new java.awt.Dimension(80, 25));
asin.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        asinActionPerformed(evt);
    }
});

acos.setText("acos");
acos.setPreferredSize(new java.awt.Dimension(80, 25));
acos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        acosActionPerformed(evt);
    }
```

```
});

atan.setText("atan");
atan.setPreferredSize(new java.awt.Dimension(80, 25));
atan.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        atanActionPerformed(evt);
    }
});

to_Degrees.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
to_Degrees.setText("to Degrees");
to_Degrees.setPreferredSize(new java.awt.Dimension(85, 25));
to_Degrees.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        to_DegreesActionPerformed(evt);
    }
});

to_Radians.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
to_Radians.setText("to Radians");
to_Radians.setPreferredSize(new java.awt.Dimension(83, 25));
to_Radians.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        to_RadiansActionPerformed(evt);
    }
});
```

```
            jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
            jLabel6.setText("Scientific supervisor:  Ph.D., Assoc. Oksana Koshova ");

            PI.setFont(new java.awt.Font("Times New Roman", 1, 14)); // NOI18N
            PI.setText("Ї€");
            PI.setPreferredSize(new java.awt.Dimension(80, 25));
            PI.addActionListener(new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    PIActionPerformed(evt);
                }
            });

            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(48, 48, 48)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE,                    467,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(78, 78, 78))
            .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE,                    606,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE,                    358,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE,                    367,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(135, 135, 135))
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                .addComponent(jLabel7,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                    .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE,                    211,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                    .addGap(2, 2, 2)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE,                    209,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                    .addComponent(number4,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(number2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(number1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(178, 178, 178)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                    .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(jLabel10,
javax.swing.GroupLayout.PREFERRED_SIZE,                    204,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(number7,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE,                    211,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel9,
javax.swing.GroupLayout.PREFERRED_SIZE,                    209,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addComponent(number6,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(number5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))))))
```

```
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(resultButton,
javax.swing.GroupLayout.PREFERRED_SIZE,                    52,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(number3,
javax.swing.GroupLayout.PREFERRED_SIZE,                    298,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, false)
                            .addComponent(sumButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(Iog,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
```

```
                .addComponent(subtractButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(abs,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                .addComponent(multiplyButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(greater_than,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                .addComponent(dividingButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(less_than,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                .addComponent(sqrt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(asin,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))
                                .addGroup(layout.createSequentialGroup()
                                .addComponent(averageButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(cos,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(10, 10, 10)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, false)
                        .addComponent(sin,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(acos,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(atan,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(to_Degrees,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(tan,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(PI,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                            .addComponent(to_Radians,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(275, 275, 275)
                            .addComponent(claerButton,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    380,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addContainerGap(19, Short.MAX_VALUE))
                );
                layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(6, 6, 6)
                        .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    36,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel5)
                .addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                            .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(number5,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(jLabel9,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(number6,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(jLabel10,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(number7,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
```

```
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE,                    29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(number1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE,                    29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(number2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(jLabel7,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(number4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGap(28, 28, 28)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                        .addComponent(sumButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(subtractButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(multiplyButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(dividingButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(averageButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(cos,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(sin,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(tan,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(PI,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(Iog,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(abs,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(greater_than,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(less_than,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(sqrt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(asin,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(acos,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(atan,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(to_Degrees,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(to_Radians,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(number3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(resultButton,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    22,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(claerButton)
                .addGap(19, 19, 19))
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private void number1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_number1ActionPerformed
            // TODO add your handling code here:
```

```
    }//GEN-LAST:event_number1ActionPerformed

    private void sumButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_sumButtonActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        a[1] = Double.parseDouble(number2.getText());
        a[2] = Double.parseDouble(number4.getText());
        a[3] = Double.parseDouble(number5.getText());
        a[4] = Double.parseDouble(number6.getText());
        a[5] = Double.parseDouble(number7.getText());

        Double sum = a[0] + a[1] + a[2] + a[3] + a[4] + a[5];
        number3.setText("" + sum);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_sumButtonActionPerformed

    private void subtractButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_subtractButtonActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        a[1] = Double.parseDouble(number2.getText());
        a[2] = Double.parseDouble(number4.getText());
        a[3] = Double.parseDouble(number5.getText());
        a[4] = Double.parseDouble(number6.getText());
        a[5] = Double.parseDouble(number7.getText());

        Double subtract = a[0] - a[1] - a[2] - a[3] - a[4] - a[5];
```

```
            number3.setText("" + subtract);
            claerButton.setEnabled(true);

        }//GEN-LAST:event_subtractButtonActionPerformed

        private void multiplyButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_multiplyButtonActionPerformed
            double[] a = new double[6];
            a[0] = Double.parseDouble(number1.getText());
            a[1] = Double.parseDouble(number2.getText());
            a[2] = Double.parseDouble(number4.getText());
            a[3] = Double.parseDouble(number5.getText());
            a[4] = Double.parseDouble(number6.getText());
            a[5] = Double.parseDouble(number7.getText());

            Double multiply = a[0] * a[1] * a[2] * a[3] * a[4] * a[5];
            number3.setText("" + multiply);
            claerButton.setEnabled(true);

        }//GEN-LAST:event_multiplyButtonActionPerformed

        private void dividingButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_dividingButtonActionPerformed
            double[] a = new double[6];
            a[0] = Double.parseDouble(number1.getText());
            a[1] = Double.parseDouble(number2.getText());
            a[2] = Double.parseDouble(number4.getText());
            a[3] = Double.parseDouble(number5.getText());
            a[4] = Double.parseDouble(number6.getText());
```

```
        a[5] = Double.parseDouble(number7.getText());

        Double dividing = a[0] / a[1] / a[2] / a[3] / a[4] / a[5];
        number3.setText("" + dividing);
        claerButton.setEnabled(true);
    }//GEN-LAST:event_dividingButtonActionPerformed

    private void averageButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_averageButtonActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        a[1] = Double.parseDouble(number2.getText());
        a[2] = Double.parseDouble(number4.getText());
        a[3] = Double.parseDouble(number5.getText());
        a[4] = Double.parseDouble(number6.getText());
        a[5] = Double.parseDouble(number7.getText());

        Double average = (a[0] + a[1] + a[2] + a[3] + a[4] + a[5]) / 2;
        number3.setText("" + average);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_averageButtonActionPerformed

    private void claerButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_claerButtonActionPerformed
        number1.setText("");
        number2.setText("");
        number3.setText("");
        number4.setText("");
```

```
        number5.setText("");
        number6.setText("");
        number7.setText("");
        claerButton.setEnabled(false);

    }//GEN-LAST:event_claerButtonActionPerformed

    private void number2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_number2ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_number2ActionPerformed

    private void number4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_number4ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_number4ActionPerformed

    private void number3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_number3ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_number3ActionPerformed

    private void cosActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cosActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());

        Double cos = Math.cos(a[0]);
        number3.setText("" + cos);
```

```
        claerButton.setEnabled(true);

    }//GEN-LAST:event_cosActionPerformed

    private void sinActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_sinActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());

        Double sin = Math.sin(a[0]);
        number3.setText("" + sin);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_sinActionPerformed

    private void tanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_tanActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());

        Double tan = Math.tan(a[0]);
        number3.setText("" + tan);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_tanActionPerformed

    private void PIActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_PIActionPerformed
```

```
        double[] a = new double[6];

        Double PI = Math.PI;
        number3.setText("" + PI);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_PIActionPerformed

    private void IogActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_IogActionPerformed

        double[] a = new double[6];

        a[0] = Double.parseDouble(number1.getText());

        Double Iog = Math.log10(a[0]);
        number3.setText("" + Iog);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_IogActionPerformed

    private void absActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_absActionPerformed

        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        Double abs = Math.abs(a[0]);

        number3.setText("" + abs);
```

```
        claerButton.setEnabled(true);

    }//GEN-LAST:event_absActionPerformed

    private void greater_thanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_greater_thanActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        a[1] = Double.parseDouble(number2.getText());
        a[2] = Double.parseDouble(number4.getText());
        a[3] = Double.parseDouble(number5.getText());
        a[4] = Double.parseDouble(number6.getText());
        a[5] = Double.parseDouble(number7.getText());

        Double greater_than1 = Math.max(a[0], a[1]);
        Double greater_than2 = Math.max(greater_than1, a[2]);
        Double greater_than3 = Math.max(greater_than2, a[3]);
        Double greater_than4 = Math.max(greater_than3, a[4]);
        Double greater_than5 = Math.max(greater_than4, a[5]);

        number3.setText("" + greater_than5);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_greater_thanActionPerformed

    private void sqrtActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_sqrtActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
```

```
        Double sqr = Math.sqrt(a[0]);
        number3.setText("" + sqr);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_sqrtActionPerformed

    private void asinActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_asinActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());

        Double asin = Math.asin(a[0]);
        number3.setText("" + asin);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_asinActionPerformed

    private void acosActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_acosActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        Double acos = Math.acos(a[0]);
        number3.setText("" + acos);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_acosActionPerformed
```

```
    private void atanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_atanActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        Double acos = Math.atan(a[0]);
        number3.setText("" + acos);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_atanActionPerformed

    private void to_DegreesActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_to_DegreesActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        Double to_Degrees = Math.toDegrees(a[0]);
        number3.setText("" + to_Degrees);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_to_DegreesActionPerformed

    private void to_RadiansActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_to_RadiansActionPerformed

        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        Double to_Radians = Math.toRadians(a[0]);
        number3.setText("" + to_Radians);
        claerButton.setEnabled(true);
```

```
    }//GEN-LAST:event_to_RadiansActionPerformed

    private void less_thanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_less_thanActionPerformed
        double[] a = new double[6];
        a[0] = Double.parseDouble(number1.getText());
        a[1] = Double.parseDouble(number2.getText());
        a[2] = Double.parseDouble(number4.getText());
        a[3] = Double.parseDouble(number5.getText());
        a[4] = Double.parseDouble(number6.getText());
        a[5] = Double.parseDouble(number7.getText());

        Double less_than1 = Math.min(a[0], a[1]);
        Double less_than2 = Math.min(less_than1, a[2]);
        Double less_than3 = Math.min(less_than2, a[3]);
        Double less_than4 = Math.min(less_than3, a[4]);
        Double less_than5 = Math.min(less_than4, a[5]);

        number3.setText("" + less_than5);
        claerButton.setEnabled(true);

    }//GEN-LAST:event_less_thanActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
```

```
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(NewJFrame1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(NewJFrame1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(NewJFrame1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(NewJFrame1.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
            }
            //</editor-fold>

            /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new NewJFrame1().setVisible(true);
                }
            });
        }

        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JButton Iog;
        private javax.swing.JButton PI;
        private javax.swing.JButton abs;
        private javax.swing.JButton acos;
        private javax.swing.JButton asin;
        private javax.swing.JButton atan;
        private javax.swing.JButton averageButton;
        private javax.swing.JButton claerButton;
        private javax.swing.JButton cos;
        private javax.swing.JButton dividingButton;
        private javax.swing.JButton greater_than;
        private javax.swing.JLabel jLabel1;
        private javax.swing.JLabel jLabel10;
        private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JButton less_than;
    private javax.swing.JButton multiplyButton;
    private javax.swing.JTextField number1;
    private javax.swing.JTextField number2;
    private javax.swing.JTextField number3;
    private javax.swing.JTextField number4;
    private javax.swing.JTextField number5;
    private javax.swing.JTextField number6;
    private javax.swing.JTextField number7;
    private javax.swing.JLabel resultButton;
    private javax.swing.JButton sin;
    private javax.swing.JButton sqrt;
    private javax.swing.JButton subtractButton;
    private javax.swing.JButton sumButton;
    private javax.swing.JButton tan;
    private javax.swing.JButton to_Degrees;
    private javax.swing.JButton to_Radians;
    // End of variables declaration//GEN-END:variables
}
```