# National University of Computer and Emerging Sciences


## Laboratory Manual

*for*

## Data Structures Lab

| | |
|---|---|
| Course Instructor | Sir Waqas Ali |
| Lab Instructor(s) | Zain Ali Nasir |
| Section | BSE 5C |
| Date | Wednesday, August 28, 2024 |
| Semester | Fall 2024 |

**Department of Software Engineering**
FAST-NU, Lahore, Pakistan

## In this lab we cover:

- Input through scanner and JOptionPane
- Wrapper classes
- Control structures
- Loops
- Arrays

### 1. Scanner

**Declaration :**

```
import java.util.Scanner; // program uses class Scanner
```

A Scanner enables a program to read data (e.g.,numbers and strings) for use in a program. The data can come from many sources, such as the user at the keyboard or a file on disk. Before using a Scanner, you must create it and specify the source of the data.The = below indicates that Scanner variable input should be initialized (i.e., pre-paired for use in the program) in its declaration with the result of the expression to the right of the equals sign—new Scanner(System.in). This expression uses the new keyword to create a Scanner object that reads characters typed by the user at the keyboard. The standard input object, System.in, enables applications to read bytes of information typed by the user.The Scanner translates these bytes into types (like ints) that can be used in a program.

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

### 2. JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user.

JOptionPane have many methods but some frequently use methods are:

| Method Name | Description |
|---|---|
| showConfirmDialog | Asks a confirming question, like yes/no/cancel. |
| showInputDialog | Prompt for some input. |
| showMessageDialog | Tell the user about something that has happened. |
| showOptionDialog | The Grand Unification of the above three. |

```
1  // Fig. 3.18: NameDialog.java
2  // Basic input with a dialog box.
3  import javax.swing.JOptionPane;
4
5  public class NameDialog
6  {
7     public static void main( String[] args )
8     {
9        // prompt user to enter name
10       String name =
11          JOptionPane.showInputDialog( "What is your name?" );
12
13       // create the message
14       String message =
15          String.format( "Welcome, %s, to Java Programming!", name );
16
17       // display the message to welcome the user by name
18       JOptionPane.showMessageDialog( null, message );
19    } // end main
20 } // end class NameDialog
```

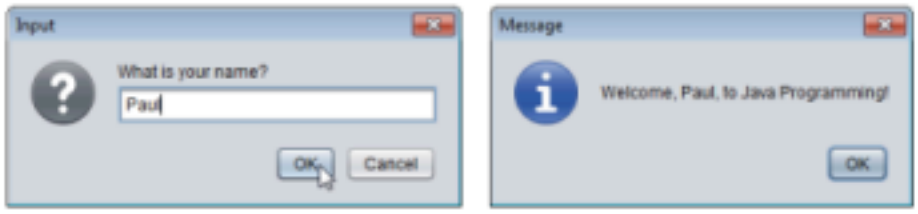



**Fig. 3.18** | Obtaining user input from a dialog.

## 3. Wrapper Classes

Java provides eight type-wrapper classes—**Boolean, Character, Double, Float, Byte, Short, Integer and Long**—that enable primitive-type values to be treated as objects.

```
ArrayList<Integer> myNumbers = new ArrayList<Integer>();
// Declaration of integer Array
```

```java
// Fig. 16.15: StaticCharMethods.java
// Character static methods for testing characters and converting case.
import java.util.Scanner;

public class StaticCharMethods
{
   public static void main( String[] args )
   {
      Scanner scanner = new Scanner( System.in ); // create scanner
      System.out.println( "Enter a character and press Enter" );
      String input = scanner.next();
      char c = input.charAt( 0 ); // get input character

      // display character info
      System.out.printf( "is defined: %b\n", Character.isDefined( c ) );
      System.out.printf( "is digit: %b\n", Character.isDigit( c ) );
      System.out.printf( "is first character in a Java identifier: %b\n",
         Character.isJavaIdentifierStart( c ) );
      System.out.printf( "is part of a Java identifier: %b\n",
         Character.isJavaIdentifierPart( c ) );
      System.out.printf( "is letter: %b\n", Character.isLetter( c ) );
      System.out.printf(
         "is letter or digit: %b\n", Character.isLetterOrDigit( c ) );
      System.out.printf(
         "is lower case: %b\n", Character.isLowerCase( c ) );
      System.out.printf(
         "is upper case: %b\n", Character.isUpperCase( c ) );
      System.out.printf(
         "to upper case: %s\n", Character.toUpperCase( c ) );
      System.out.printf(
         "to lower case: %s\n", Character.toLowerCase( c ) );
   } // end main
} // end class StaticCharMethods
```

```
Enter a character and press Enter
A
is defined: true
is digit: false
is first character in a Java identifier: true
is part of a Java identifier: true
is letter: true
is letter or digit: true
is lower case: false
is upper case: true
to upper case: A
to lower case: a
```

## 4. Control Structures

Normally, statements in a program are executed one after the other in the order in which they're written. This process is called sequential execution. Various Java statements, which we'll soon discuss, enable you to specify that the next statement to execute is **not** necessarily the next one in sequence. This is called **transfer of control.**

Here is the table of relational operator for Java:

| Standard algebraic equality or relational operator | Java equality or relational operator | Sample Java condition | Meaning of Java condition |
|---|---|---|---|
| *Equality operators* | | | |
| = | == | x == y | x is equal to y |
| ≠ | != | x != y | x is not equal to y |
| *Relational operators* | | | |
| > | > | x > y | x is greater than y |
| < | < | x < y | x is less than y |
| ≥ | >= | x >= y | x is greater than or equal to y |
| ≤ | <= | x <= y | x is less than or equal to y |

**Fig. 2.14** | Equality and relational operators.

## 5. Loops

In Java , the general format of the *for* statement is **for ( initialization; loopContinuationCondition; increment ) Statement**

**Example**

```
for ( int j = 2; j <= 80; j ++ )
```

and general format of *while* statement is

```
initialization;
while ( loopContinuationCondition )
{
statement
increment;
}
```

**Example:**

```
counter = 1
while(counter < 10)
{
 System.out.prinln(counter );
 counter++
 }
```

## 6. Arrays

An array is a group of variables (called elements or components) containing values that all have the same type. Arrays are objects, so they're considered reference types. How declared the **Arrays** in **Java:**

### Type 1 Array Definition

```
int[] c = new int[ 12 ];
```

### Type 2 Array Definition

```
int[] c; // declare the array variable
c = new int[ 12 ]; // create the array;
```

```
1  // Fig. 7.3: InitArray.java
2  // Initializing the elements of an array with an array initializer.
3
4  public class InitArray
5  {
6     public static void main( String[] args )
7     {
8        // initializer list specifies the value for each element
9        int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11       System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings
12
13       // output each array element's value
14       for ( int counter = 0; counter < array.length; counter++ )
15          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
16    } // end main
17 } // end class InitArray
```

```
Index   Value
    0      32
    1      27
    2      64
    3      18
    4      95
    5      14
    6      90
    7      70
    8      60
    9      37
```

**Fig. 7.3** | Initializing the elements of an array with an array initializer.

# Lab Manual :

## Instruction:

- Code should be proper commented.
- Proper error handling like invalid inputs and showing appropriate messages.
- Zero tolerance on plagiarism.
- Use of any code generation tool is prohibited, if anyone found using code generation tools may face serious consequences for instance **F** grade in Lab or zero in all labs.
- Complete Lab within allocated time. No extra time will be awarded.

**Q1: Fibonacci Series using JOptionPane**

Write a program using JOptionPane to show the fibonacci series till the number of terms entered by the user.

**Input:**

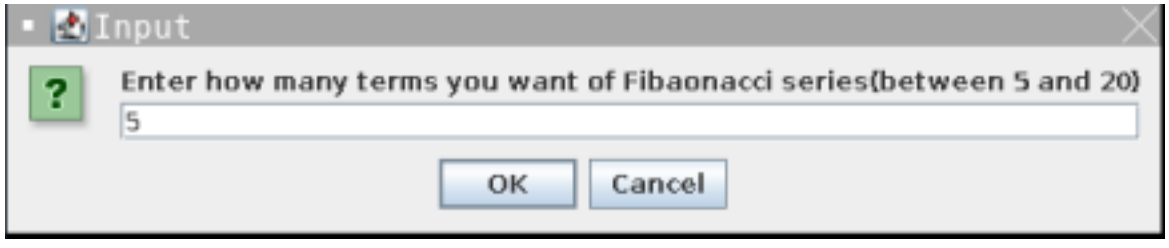
How many terms you want:

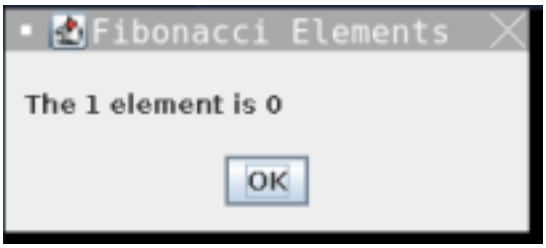
User enters : 5

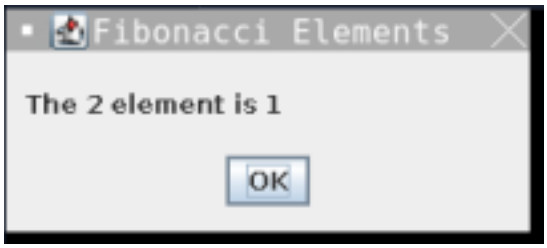
Note: This number should be between 5 to 20

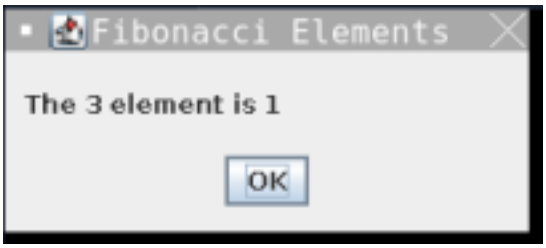
Now on every click user get next elements of series: In this case:

**Sample Output:**






**Q2: Student Grading**

Build a console base application for grading of students. In this application input students' marks at runtime from the user.Use wrapper classes for storing the students' marks.

.

**You application should return :**

Maximum marks

Average marks

Student < Average Marks

**Input:**

User enter : 10.2 12 33.5

**Output:**

Maximum Marks: 33

Average Marks : 18.5666666..

Student Less than Average Marks : 2

## Sample output:

```
Enter the student Marks

Enter the number of students:
3
Enter the marks of student 1:
10.2
Enter the marks of student 2:
12
Enter the marks of student 3:
33.5
Average of Class

Average marks: 18.566666666666666
Maximum Marks

Marks: 33

 Number of students less than average

Marks of student 1: 10.2
Marks of student 2: 12.0
Students less than average: 2

 Complete Marks Array

Marks of student 1: 10.2
Marks of student 2: 12.0
Marks of student 3: 33.5
Marks of student Integer1: 10
Marks of student Integer2: 12
Marks of student Integer3: 33
```