Here’s the ordered list of all the algorithms you've requested:

---

1. Algorithm for the NegativeException Code:

1. **Initialize Variables:**
    - Declare variables: n, i, sum, and avg.
2. **Input Integer:**
    - Read the integer n from command-line arguments.
3. **Check Validity:**
    - If n <= 0, throw an IllegalArgumentException.
4. **Calculate Sum:**
    - For valid n, use a loop to sum numbers from 1 to n.
5. **Calculate Average:**
    - Divide the sum by n to get the average.
6. **Display Result:**
    - Print the average.
7. **Handle Exception:**
    - If an invalid number is entered, print "InvalidNumberException".

---

2. Algorithm for the ArrayListEx Code:

1. **Initialize ArrayList:**
    - Create an ArrayList<Integer> named list.
    - Initialize a Scanner for user input.
2. **Display Menu:**
    - Print a menu with options: Insert, Delete, Display, Exit.
3. **Start Loop:**
    - Use an infinite loop to continuously ask for user input.
4. **Process User Choice:**
    - **Insert:** Add user input to the list.
    - **Delete:** Remove the element at a given index.
    - **Display:** Print all elements in the list.
    - **Exit:** Close the scanner and exit.

5. **Handle Invalid Input:**
    - Display an error message for invalid menu choices.

---

3. Algorithm for the LinkedListEx Code:

1. **Initialize LinkedList:**
    - Create a LinkedList<Integer> and a Scanner.
2. **Display Menu:**
    - Print options: Create, Delete, Display, Exit.
3. **Loop for User Input:**
    - Continuously prompt for a choice from the menu.
4. **Process User Choice:**
    - **Create:** Add user input to the list.
    - **Delete:** Remove an element by its index.
    - **Display:** Print all elements using an iterator.
    - **Exit:** Close the scanner and terminate the program.
5. **Handle Invalid Choices:**
    - Display an error message for invalid inputs.

---

4. Algorithm for the BubbleSort Code:

1. **Initialize ArrayList:**
    - Create an ArrayList<Integer> to store elements.
    - Use a Scanner to get user input.
2. **Input Array Size:**
    - Ask for the size of the array (n).
3. **Input Elements:**
    - Add n elements to the list.
4. **Perform Bubble Sort:**
    - Use nested loops to compare adjacent elements and swap them if they are in the wrong order.
5. **Display Sorted Array:**
    - Print the sorted list.

6. **Close Scanner:**
    - Close the scanner after sorting and displaying.

---

5. Algorithm for the BinarySearch Code:

1. **Initialize ArrayList:**
    - Create a list for integer elements and a scanner for input.
2. **Input Array Size and Elements:**
    - Read the size n and the sorted elements from the user.
3. **Perform Bubble Sort:**
    - Sort the list using the Bubble Sort algorithm to ensure it's sorted.
4. **Display Sorted Array:**
    - Print the sorted array.
5. **Input Search Key:**
    - Ask for the element to search.
6. **Binary Search:**
    - Initialize low, high, and mid.
    - Use a loop to perform binary search by adjusting low and high based on comparisons.
    - If the element is found, print its position.
7. **Handle Element Not Found:**
    - If the element is not found, display an appropriate message.
8. **Close Scanner:**
    - Close the scanner when finished.

---

6. Algorithm for the Stringtokendemo Code:

1. **Initialize Scanner:**
    - Create a scanner to get input from the user.
2. **Input String of Integers:**
    - Ask the user to input integers separated by spaces.
3. **Tokenize the String:**
    - Use StringTokenizer to break the input into tokens (integers).

4. **Process Each Token:**
    - For each token, convert it to an integer and add it to the sum.
    - Print each integer.
5. **Display Sum:**
    - After processing all tokens, display the sum of the integers.
6. **Close Scanner:**
    - Close the scanner when finished.

1. StringTokenDemo Algorithm

plaintext Copy code

```plaintext
BEGIN StringTokenDemo
    DECLARE n, sum as INTEGER
    SET sum = 0
    PROMPT user for input "Enter integers separated by space"
    READ input string s

    CREATE StringTokenizer st with s using " " as delimiter
    WHILE st has more tokens DO
        SET temp to next token from st
        SET n to parse temp as INTEGER
        PRINT n
        sum = sum + n
    END WHILE

    PRINT "Sum of INTEGERS is " + sum
END
```

plaintext

Copy code

```
BEGIN OddThread
    DECLARE cube as INTEGER

    FUNCTION OddThread(cube)
        IF cube < 0 THEN
            cube = -cube
        END IF
        PRINT "Cube of " + cube + " is: " + (cube * cube * cube)
    END FUNCTION
END

BEGIN EvenThread
    DECLARE sq as INTEGER

    FUNCTION EvenThread(sq)
        IF sq < 0 THEN
            sq = -sq
        END IF
        PRINT "Square of " + sq + " is: " + (sq * sq)
    END FUNCTION
END

BEGIN RandThread
    DECLARE n as INTEGER
    FUNCTION RandThread(n)
        CREATE Random r
        FOR i FROM 0 TO n-1 DO
            SET nn = r.nextInt(100)
            IF nn % 2 == 0 THEN
                CREATE EvenThread t2 with nn
                START t2
            ELSE
                CREATE OddThread t3 with nn
                START t3
            END IF
        END FOR
    END FUNCTION
END

BEGIN main
    PROMPT user for input "Enter limit"
    READ l as INTEGER

    CREATE RandThread t1 with l
    START t1

    WAIT for t1 to finish
END
```

plaintext

Copy code

```
BEGIN Synchronization
    FUNCTION display(x)
        PRINT "Square of " + x + " is " + (x * x)
    END FUNCTION

    FUNCTION run()
        SET name to current thread name
        IF name == "t1" THEN
            CALL display(2)
        ELSE IF name == "t2" THEN
            CALL display(3)
        ELSE
            CALL display(4)
        END IF
    END FUNCTION

    FUNCTION main()
        CREATE Synchronization s
        CREATE Thread t1, t2, t3 using s
        SET names for t1, t2, t3 as "t1", "t2", "t3"
        START t1, t2, t3
    END FUNCTION
END
```

plaintext

Copy code

```
BEGIN CopyCharFile
    CREATE BufferedReader br
    PROMPT user for input "Enter the source file name"
    READ in as STRING

    CREATE FileReader fis with in
    PROMPT user for input "Enter the destination file name"
    READ out as STRING

    CREATE FileWriter fos with out
    DECLARE buffer as CHAR array of size 1024
    DECLARE bufferSize as INTEGER

    WHILE (bufferSize = fis.read(buffer)) != -1 DO
        fos.write(buffer, 0, bufferSize)
    END WHILE

    CLOSE fis
    CLOSE fos
    PRINT "File copied successfully."
END
```

5. FileStatsCounter Algorithm

plaintext Copy code

```plaintext
BEGIN FileStatsCounter
    DECLARE characterCount, wordCount, lineCount as INTEGER
    SET characterCount = 0, wordCount = 0, lineCount = 0

    CREATE Scanner scanner
    PROMPT user for input "Enter the file name"
    READ fileName as STRING

    CREATE FileReader fileReader with fileName
    CREATE BufferedReader bufferedReader with fileReader

    WHILE (line = bufferedReader.readLine()) != null DO
        INCREMENT lineCount
        SET words to split line by whitespace
        ADD words.length to wordCount
        ADD line.length() to characterCount
    END WHILE

    CLOSE bufferedReader
    PRINT "Total number of characters: " + characterCount
    PRINT "Total number of words: " + wordCount
    PRINT "Total number of lines: " + lineCount
END
```

plaintext Copy code

```
BEGIN FileSum
    CREATE BufferedReader br
    PROMPT user for input "Enter input file name"
    READ infile as STRING
    CREATE FileReader fis with infile
    CREATE BufferedReader fileReader with fis

    PROMPT user for input "Enter output file name"
    READ outfile as STRING
    CREATE FileWriter fos with outfile
    CREATE BufferedWriter bw with fos

    DECLARE line as STRING
    IF (line = fileReader.readLine()) != null THEN
        SET numbers to split line by whitespace
        DECLARE sum as INTEGER
        SET sum = 0
        FOR EACH num in numbers DO
            DECLARE currentNumber as INTEGER
            SET currentNumber = parse num as INTEGER
            PRINT currentNumber + " "
            sum = sum + currentNumber
            bw.write("Integer: " + num + "\n")
        END FOR
        bw.write("Sum: " + sum)
        CLOSE bw
        PRINT "\nSum and Integers copied to " + outfile
    ELSE
        PRINT "Input file is empty."
    END IF
END
```

7.Calculator Algorithm

```
BEGIN Calculator
  INITIALIZE GUI components: JTextField t1, JButtons b1 to b16
  SET GUI layout and bounds

  FUNCTION doAction(op)
    IF operation is null THEN
      SET operation = op
```

```
    SET res = parse t1.text as INTEGER
    CLEAR t1.text
  ELSE
    SWITCH operation
      CASE "+":
        res = res + parse t1.text as INTEGER
      CASE "-":
        res = res - parse t1.text as INTEGER
      CASE "*":
        res = res * parse t1.text as INTEGER
      CASE "/":
        IF t1.text == "0" THEN
          SET res = 0
          SET t1.text = "Divide by Zero"
          SET operation = null
          RETURN
        END IF
        res = res / parse t1.text as INTEGER
      CASE "%":
        res = res % parse t1.text as INTEGER
    END SWITCH

    IF op == "=" THEN
      SET t1.text = res.toString()
      SET res = 0
      SET operation = null
    ELSE
      SET operation = op
      CLEAR t1.text
    END IF
  END IF
```

```
    END FUNCTION

    FUNCTION actionPerformed(e)
        SWITCH e.source
            CASE b1: t1.text += "1"
            CASE b2: t1.text += "2"
            CASE b3: t1.text += "3"
            CASE b4: doAction("+")
            CASE b5: t1.text += "4"
            CASE b6: t1.text += "5"
            CASE b7: t1.text += "6"
            CASE b8: doAction("-")
            CASE b9: t1.text += "7"
            CASE b10: t1.text += "8"
            CASE b11: t1.text += "9"
            CASE b12: doAction("*")
            CASE b13: doAction("/")
            CASE b14: doAction("%")
            CASE b15: doAction("=")
            CASE b16: CLEAR t1.text; SET res = 0; SET operation = null
        END SWITCH
    END FUNCTION

    FUNCTION main()
        INITIALIZE JFrame
        SET JFrame visibility to true
    END FUNCTION
END
```

7. Traffic Light Algorithm

```
BEGIN TrafficLight
  INITIALIZE GUI components: JRadioButton rl, r2, r3
  CREATE ButtonGroup and add radio buttons
  SET default selection
  FUNCTION actionPerformed(e)
    IF rl.isSelected() THEN
      SET red_c = Color.red
      SET green_c = getBackground()
      SET orange_c = getBackground()
    ELSE IF r2.isSelected() THEN
      SET red_c = getBackground()
      SET green_c = Color.green
      SET orange_c = getBackground()
    ELSE IF r3.isSelected() THEN
      SET red_c = getBackground()
      SET green_c = getBackground()
      SET orange_c = Color.orange
    END IF
    CALL repaint()
  END FUNCTION
  FUNCTION paintComponent(g)
    CALL super.paintComponent(g)
    DRAW circles
    FILL circles with respective colors
  END FUNCTION
  FUNCTION main()
    INITIALIZE JFrame
    SET JFrame visibility to true
  END FUNCTION
END
```