শিক্ষা নিয়ে গড়বো দেশ তথ্য-প্রযুক্তির বাংলাদেশ

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

Bangabandhu Sheikh Mujibur Rahman
Digital University

LAB REPORT-06

COURSE NO.- PROG 112
COURSE TITLE- OBJECT ORIENTED PROGRAMMING SESSIONAL

SUBMITTED BY

Mehrin Farzana
ID: 2101013
Department of IRE
Session :2021-2022
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

SUBMITTED TO

Md.Toukir Ahmed
Lecturer
Department of IRE
Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

---

Date of Submission: 25 September, 2023

Problem No.: 01

Problem Name: A Project using Java.

Code:

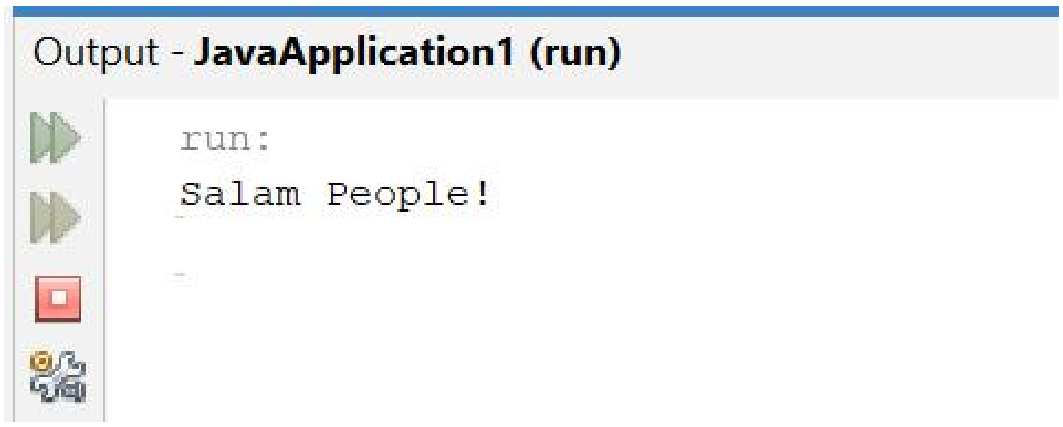
Main File:

```
public class JavaApplication1 {


   public static void main(String[] args) {
      System.out.println(Salam People!);
      JavaApplication1Frame myFrame = new JavaApplication1Frame();
      myFrame.init();
   }

}
```

Attached File:

```
import javax.swing.JFrame;
import javax.swing.WindowConstants;

public class JavaApplication1Frame extends JFrame {

   public void init(){
      setTitle("Salam!");
      setSize(450, 300);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      setVisible(true);
   }

 }
```

Output:


Output - **JavaApplication1 (run)**

```
run:
Salam People!
```

Fig 1.1: Output on console.

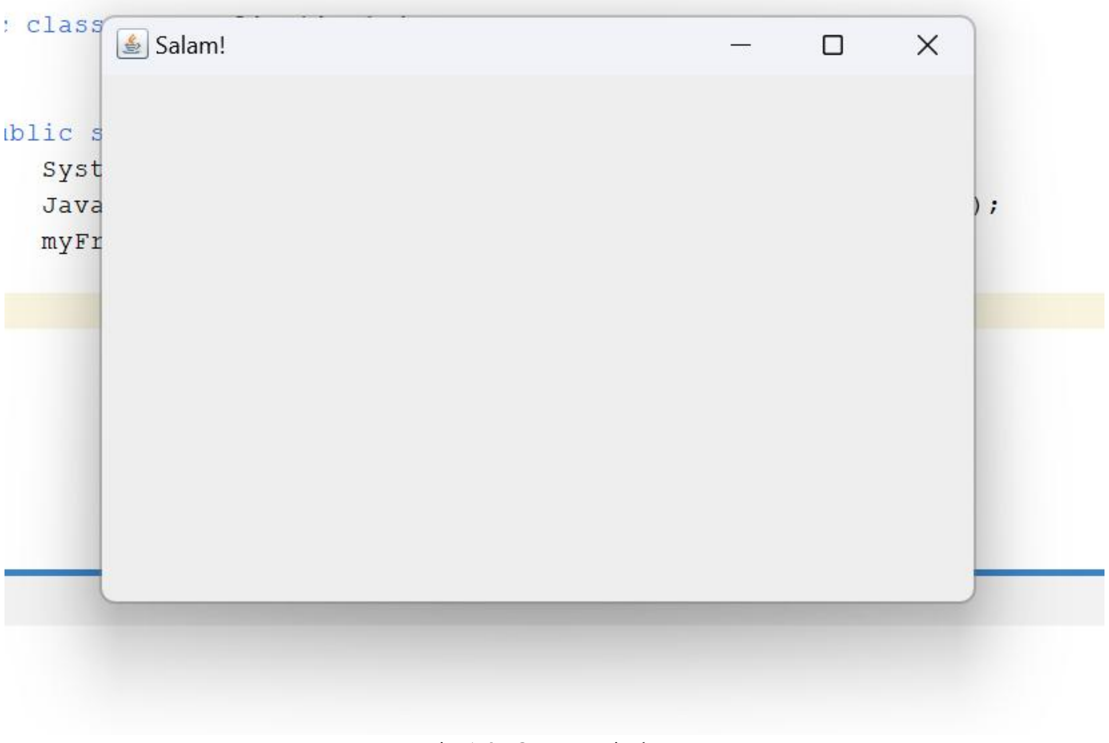


Fig 1.2: Output window.

Explanation:

In this code, Java is used to create Graphical User Interface or GUI for short. This code generates a simple window with a title and will close on clicking the close button.

Problem No.: 02

Problem Name: A Project using Java.

Code:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */

/**
 *
 * @author BAB AL SAFA
 */
public class WelcomeFrame extends javax.swing.JFrame {

    /**
     * Creates new form WelcomeFrame
     */
    public WelcomeFrame() {
        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">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        tfTitle = new javax.swing.JTextField();
        btnOK = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Title");

        btnOK.setText("OK");
        btnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOKActionPerformed(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()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(tfTitle))
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addGap(155, 155, 155)
                .addComponent(btnOK)
                .addContainerGap(173, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(18, 18, 18)
                .addComponent(tfTitle, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D, 209, Short.MAX_VALUE)
                .addComponent(btnOK)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

    }

    /**
     * @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(WelcomeFrame.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(WelcomeFrame.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(WelcomeFrame.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

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

    // Variables declaration - do not modify
    private javax.swing.JButton btnOK;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField tfTitle;
    // End of variables declaration
}
```

Modified portion of the code:

```
private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String title  = tfTitle.getText();
        setTitle(title);
    }
```

Output:

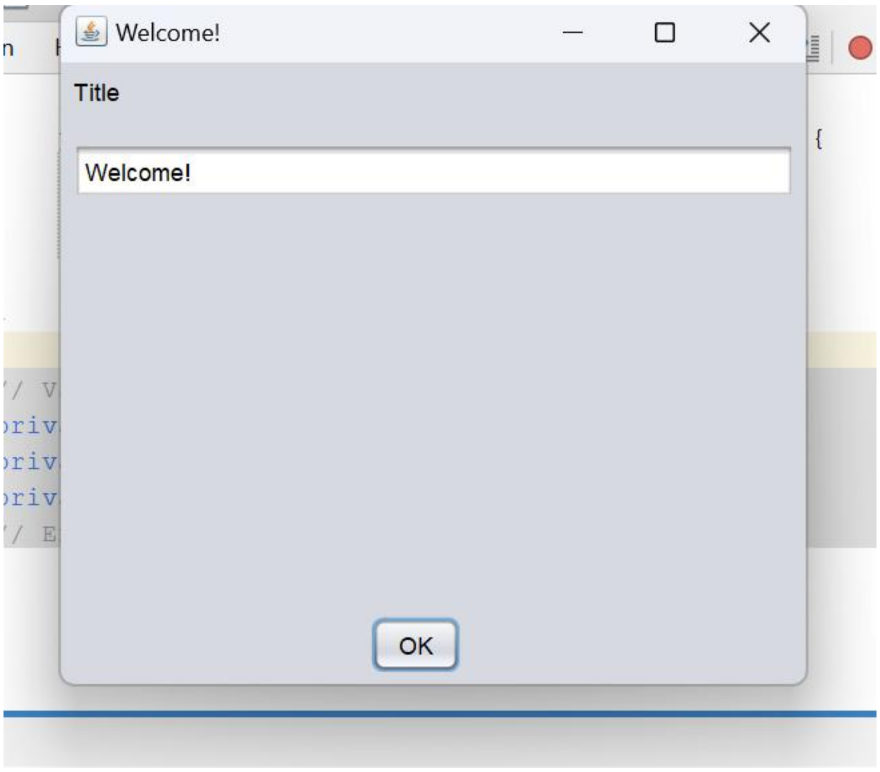


Fig 2.1: Output Window.

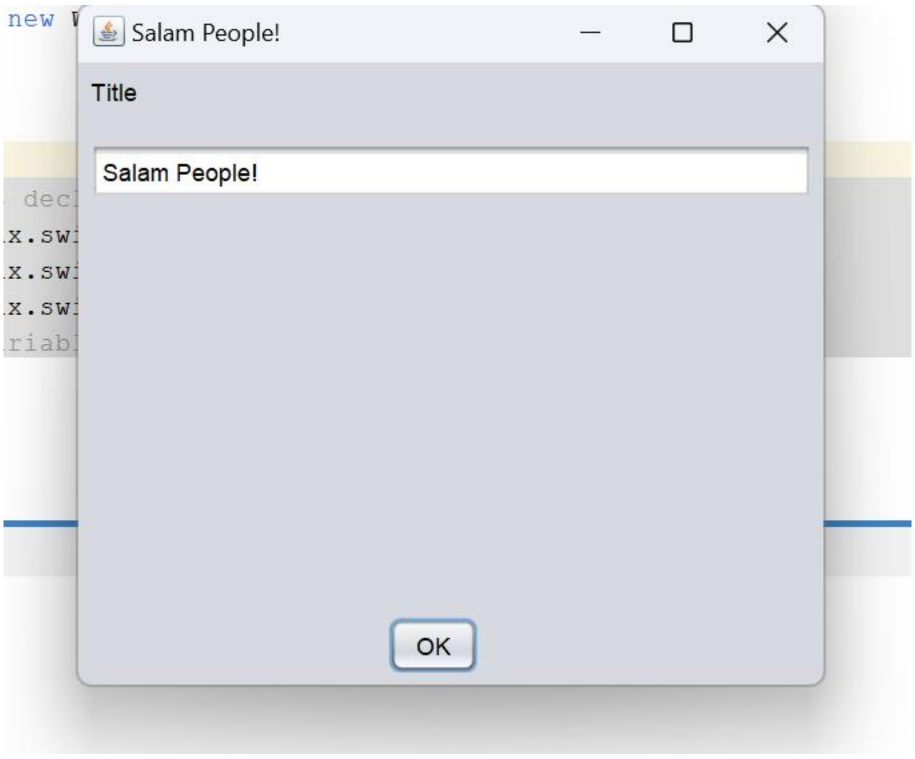


Fig 2.2: Output Window.

Explanation:

This code was auto generated for this project done on Apache NetBeans while creating GUI. The only part that was modified was the btnOKActionPerformed() method for the added button to perform the action of reading text from the text field and setting the title of the window to it.

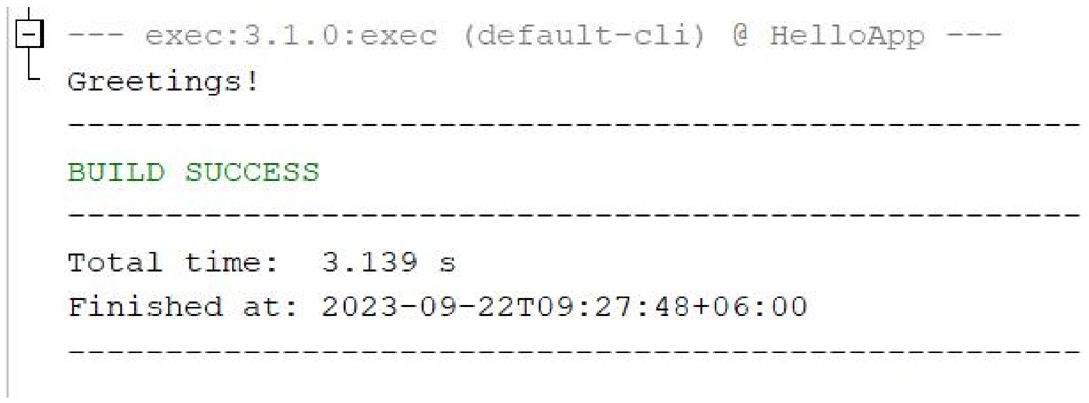
Problem No.: 03

Problem Name: Java project with Maven in Apache NetBeans.

Code:

```
package bdu.helloapp;

public class myApp {
   public static void main(String[] args) {
      System.out.println("Greetings!");
   }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
Greetings!
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.139 s
Finished at: 2023-09-22T09:27:48+06:00
------------------------------------------------------------------------
```

Fig 3.1: Output on console.

Explanation:

The code is a simple Java program that prints “Greetings!” to the console.

The program consists of a single class named myApp. The main method is the entry point of the program, which is executed when the program is run. The System.out.println statement inside the main method prints the string “Greetings!” to the console.

If compiled and run this program, it is seen “Greetings!” printed to the console.

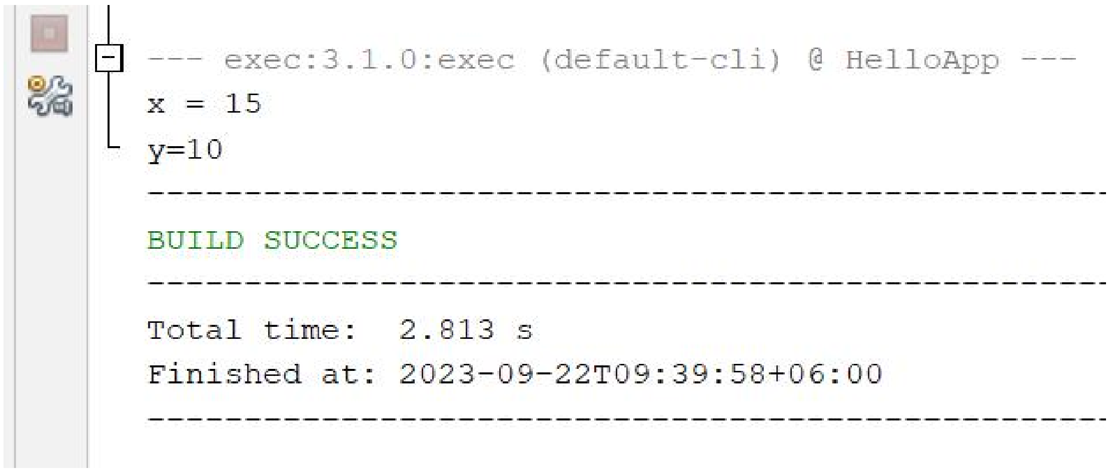
Problem No.: 04

Problem Name: Declaring a variables in java using NetBeans.

Code:

```
package bdu.helloapp;

public class myApp {
    public static void main(String[] args) {
        int x = 5;
        int y;
        y = 10;
        x = x+y;
        System.out.println("x = "+x+"\ny="+y);
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
x = 15
y=10
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.813 s
Finished at: 2023-09-22T09:39:58+06:00
------------------------------------------------------------------------
```

Fig 4.1: Output on console.

Explanation:

Ths code demostrates decraling variables in Java.

Here variable x was declared and assigned value at the same line whilst variable y was declared frist and then on the next line was assigned value. Then again x was assigned an expression.

Problem No.: 05

Problem Name: Identifiers in Java.

Code:

```
package bdu.helloapp;

public class myApp {
   public static void main(String[] args) {
       int a2;         //OK
       //int 2a;         //not OK
       int _a;         //OK
       int a_;         //OK
       int $a;         //OK
       //int a b;        //not OK
       //int &a;         //not OK
       //int *a;         //not OK
       //int &a;         //not OK
       //int 2;          //not OK
       //int void;       //not OK
       //int boolean;    //not OK
       int Boolean;    //OK
       int mainSystem; //OK
       int mSys;       //OK


   }
}
```

Explanation:

The above code demonstrates legal and illegal identifiers in Java.

Rules:

1. Can have letters, numbers, underscores (_) and dollar signs ($).
2. Cannot contain spaces
3. Cannot contain number at the beginning

Convention:

1. Follow Camel style for multi-word identifier (myName, systemJunk etc.)
2. Do not use Dollar sign ($)
3. Avoid writing short forms (myNm, sysJnk etc.). Rather use descriptive names.

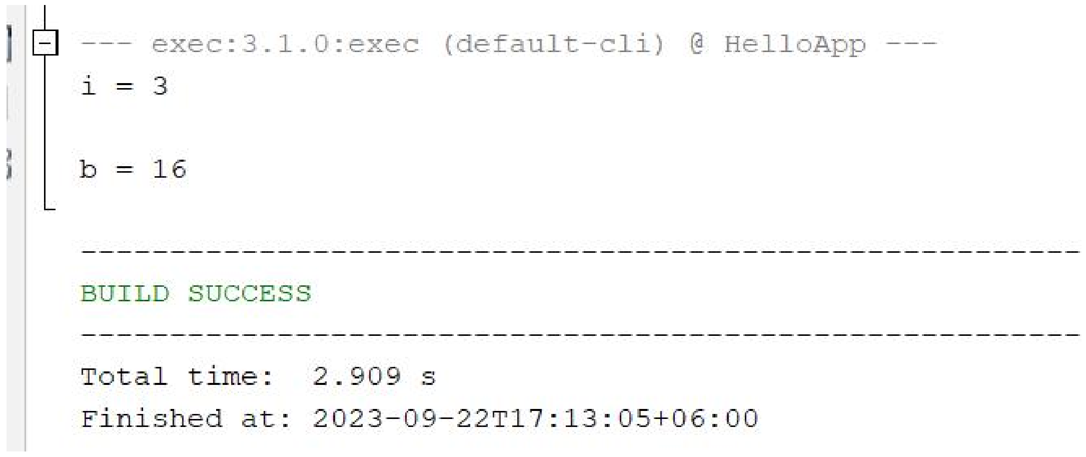
Problem No.: 06

Problem Name: Type casting in Java.

Code:

```
package bdu.helloapp;

public class myApp {
   public static void main(String[] args) {
      //Explixit type casting

      double d = 3.1416;
      int i = (int)d;                  //Truncation
      System.out.println("i = "+i+"\n");

      long l = 10000;
      byte b = (byte)l;                //Data loss
       System.out.println("b = "+b+"\n");
   }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
i = 3

b = 16

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.909 s
Finished at: 2023-09-22T17:13:05+06:00
```

Fig 6.1: Output on console.

Explanation:

The above code demonstrates explicit type casting. It shows truncating while a float or double is assigned to int or long, the floating point and fractional numbers desapear. It’s one type of data loss. Again while a larger value is cast to a smaller variable, data is lost that is seen in the case of assigning a long unto byte.

Implicit type casting happens automatically by the compiler and data is not lost.

In the case of explicit type casting, data might be lost and data must be assigned toa smaller size variable from a larger one.

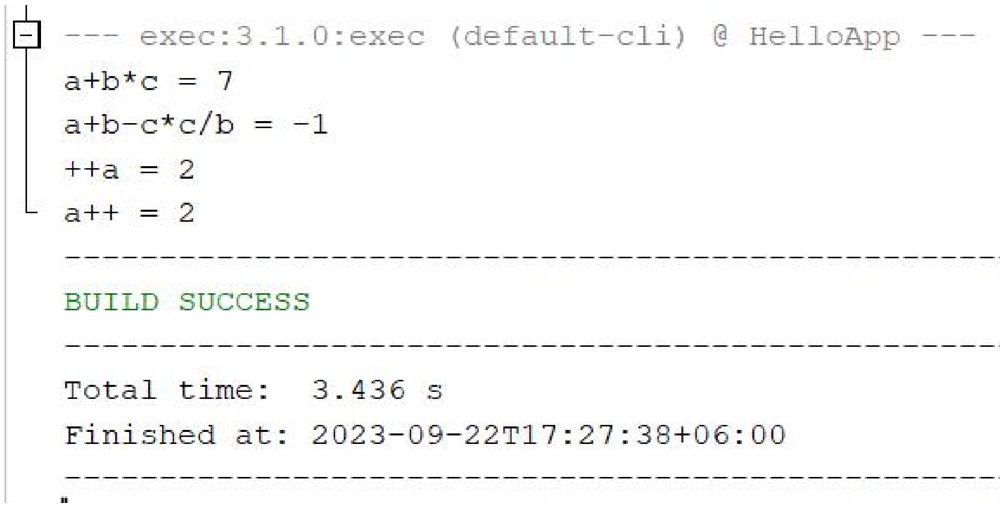
Problem No.: 07

Problem Name: Arithmetic operations in Java.

Code:

```
package bdu.helloapp;

public class myApp {
   public static void main(String[] args) {
      int a = 1;
      int b =2;
      int c = 3;
      int x = a+b*c;

      System.out.println("a+b*c = " + x);

      x = a+b-c*c/b;

      System.out.println("a+b-c*c/b = " + x);

      x = ++a;
      System.out.println("++a = " + x);

      x = a++;
      System.out.println("a++ = " + x);

   }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
a+b*c = 7
a+b-c*c/b = -1
++a = 2
a++ = 2
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
Total time:  3.436 s
Finished at: 2023-09-22T17:27:38+06:00
------------------------------------------------------
```

Fig 7.1: Output on console.

Explanation:

The above code demonstrates arithmetic operations in java which follows precedence and associativity rule.

In the case of x = a+b*c , the operators in terms of precedence, highest to lowest, *, +, = . and the associativity of + and * is from left to right whilst for = , it is right to left. Thus giving us the result x = (a+(b*c)).

In the case of x=a+b-c*c/b, it is x=((a+b)-((c*c)/b))

In the case of x=++a, it is x=(++a)

In the case of x=a++, it is (x=a) and then a++ is executed.

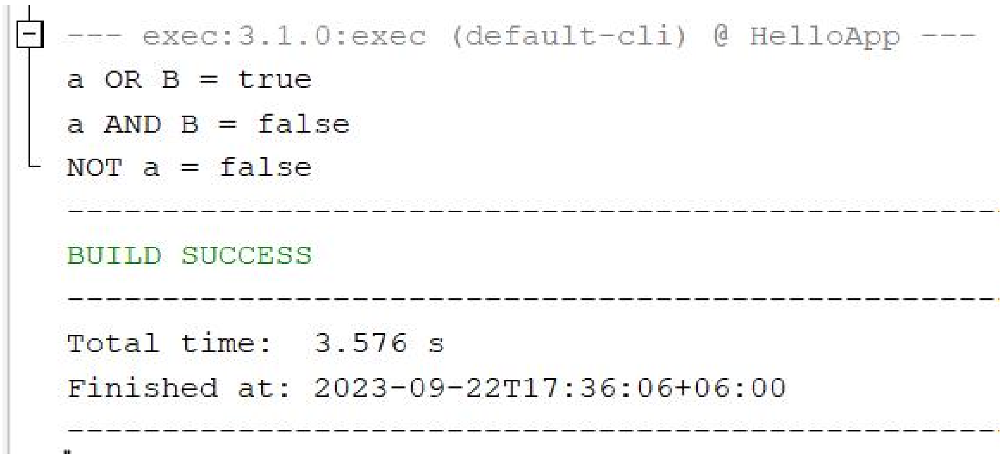
Problem No.: 08

Problem Name: Logical operators in Java.

Code:

```
package bdu.helloapp;

public class myApp {
   public static void main(String[] args) {
      boolean a=true;
      boolean b=false;
      boolean x=a||b;
      System.out.println("a OR B = "+ x);
      x=a&&b;
      System.out.println("a AND B = "+ x);
      x=!a;
      System.out.println("NOT a = "+ x);

   }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
a OR B = true
a AND B = false
NOT a = false
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
Total time:  3.576 s
Finished at: 2023-09-22T17:36:06+06:00
------------------------------------------------------
```

Fig 8.1: Output on console.

Explanation:

The above code demonstrates logical operations in Java.

OR operation outputs true iff only one operand is true.

AND operation outputs true iff all operands in true.

NOT operation reverses the truth value.

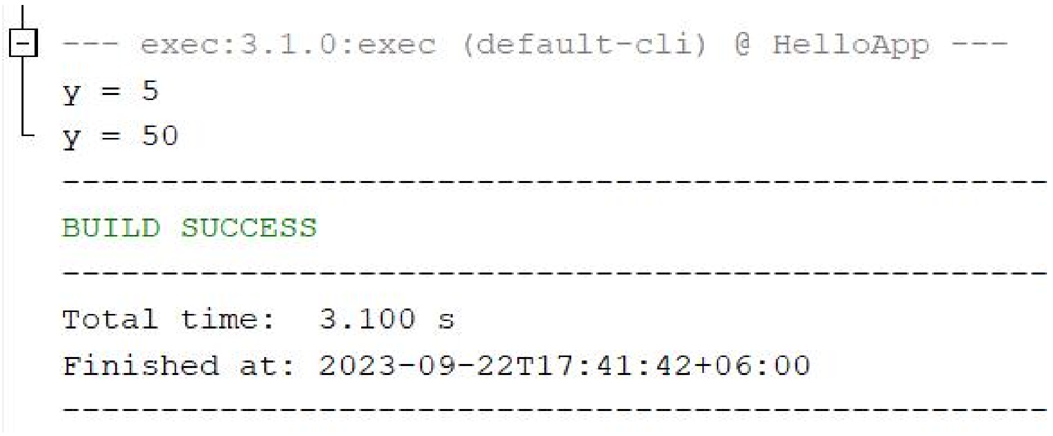
Problem No.: 09

Problem Name: Ternary operator in Java.

Code:

```
package bdu.helloapp;

public class myApp {
    public static void main(String[] args) {
        boolean a=true;
        boolean b=false;
        boolean x=a||b;
        int y=x?5:50;
        System.out.println("y = "+ y);

        x=a&&b;
        y=x?5:50;
        System.out.println("y = "+ y);


    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
y = 5
y = 50
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.100 s
Finished at: 2023-09-22T17:41:42+06:00
------------------------------------------------------------------------
```

Fig 9.1: Output on console.

Explanation:

The above code demonstrates ternary operator in Java.

The ?: operator executes the left operand of : of the operand to the left of ? is true, otherwise the right operand.

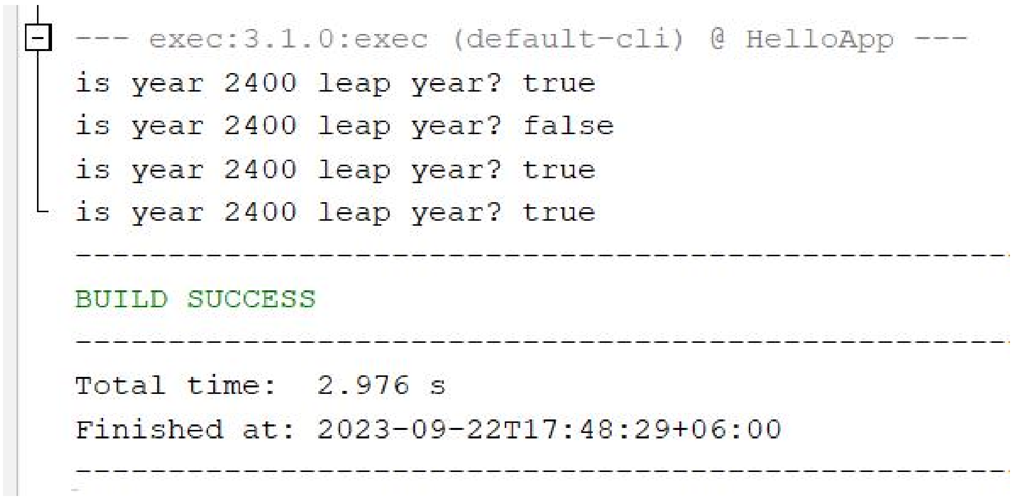
Problem No.: 10

Problem Name: Leap year checking code in Java.

Code:

```
package bdu.helloapp;

public class myApp {

    static boolean checkLeapYear(int y){
        if(y%4==0){
            if(y%100==0 && y%400!=0){
                return false;
            }
            return true;
        }
        return false;
    }

    public static void main(String[] args) {

        System.out.println("is year 2400 leap year? " +
checkLeapYear(2400));
        System.out.println("is year 2400 leap year? " +
checkLeapYear(2023));
        System.out.println("is year 2400 leap year? " +
checkLeapYear(2024));
        System.out.println("is year 2400 leap year? " +
checkLeapYear(2000));

    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ HelloApp ---
is year 2400 leap year? true
is year 2400 leap year? false
is year 2400 leap year? true
is year 2400 leap year? true
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
Total time:  2.976 s
Finished at: 2023-09-22T17:48:29+06:00
------------------------------------------------------
```

Fig 10.1: Output on console.

Explanation:

The above code checks if a given year leap year or not.

A year would be a leap year if it is divisible by 4 and not by 100 but by 400.