BSCCS2005: JAVA: Programs discussed in live sessions
Weeks {7-12}

1. **Benefits of using streams over collections when the program uses complex data structures.**

   This program has a map:
   `HashMap<City, ArrayList<Citizen>> chart`
   Corresponding to each city a list of citizens are stored inside the map. The program must find those cities where more than 50 percent of the population is not vaccinated and then it should set the `containment_zone` variable of those cities as `true`

```
import java.util.*;
import java.util.stream.*;

class City{
    private String name;
    private boolean containment_zone;
    private double population;
    private String state;
    City(String n, double p, String s){
        name = n;
        population = p;
        state = s;
    }
    public double getPopulation() {
        return population;
    }
    /*
    ============================================================================
    public City makeContainment() {
        containment_zone = true;
        return this;
      }
    we can use map(e->e.getKey().makeContainment()).forEach(e->e.display())
    and ignore the last for loop
    ============================================================================
    */

    public void makeContainment() {
        containment_zone = true;
    }
    public void display() {
        System.out.println("City: "+name+", State: "+state+",
                Containment status: "+ containment_zone);
    }
}
```

```
class Citizen{
    private String id;
    private int age;
    private boolean vaccinated;
    Citizen(String i, int a, boolean v){
        id = i;
        age = a;
        vaccinated = v;
    }
    public boolean isVaccinated() {
        return vaccinated;
    }
    public int getAge() {
        return age;
    }
    public void display() {
        System.out.println("age: "+age+", vaccinated: "+vaccinated+", id:"+id);
    }
}
public class StreamvsCollection{
    public static void test(HashMap<City, ArrayList<Citizen>> ch) {

    // TRY TO DO THIS USING COLLECTIONS INSTEAD
        ch.entrySet().stream()
            .filter( e -> e.getValue().stream()
                        .filter(citizen -> !citizen.isVaccinated())
                        .count() >= 0.5*e.getKey().getPopulation()
            ).forEach(e -> e.getKey().makeContainment());

        for(Map.Entry<City, ArrayList<Citizen>> i : ch.entrySet()) {
            i.getKey().display();
        }
    }
    public static void main(String args[]) {
        HashMap<City, ArrayList<Citizen>> chart = new LinkedHashMap<>();
        ArrayList<Citizen> arr1 = new ArrayList<>();
        ArrayList<Citizen> arr2 = new ArrayList<>();
        ArrayList<Citizen> arr3 = new ArrayList<>();
        arr1.add(new Citizen("C1",56,true));
        arr1.add(new Citizen("C2",34,false));
        arr1.add(new Citizen("C3",22,true));
        arr2.add(new Citizen("C4",87,false));
        arr2.add(new Citizen("C5",44,false));
        arr3.add(new Citizen("C6",77,false));
```

```
        chart.put(new City("Saltlake",arr1.length,"WB"), arr1);
        chart.put(new City("Pune",arr2.length,"Maharashtra"), arr2);
        chart.put(new City("Chennai",arr3.length,"TamilNadu"), arr3);
        test(chart);

    }
}
```

Exercise:
Try to modify the code so that after setting the containment zone variable for the necessary cities, it must also count the number of such cities which are containment zones and the average age of the population is more than 50. Try to put this count in an optional variable and then display the count using `ifPresentOrElse` method because there maybe no such city which satisfies the required conditions.

Note:
As you go on making more complex features on nested collections you will realise that using loops/iterators to operate upon the collections becomes more cumbersome with a lot of extra space requirement. That's why streams are useful in such scenarios.

2. **Thread Lifecycle States:**

```
import java.util.*;
public class ThreadState implements Runnable   {
    public synchronized void run() {
        try {
            Thread.sleep(800);
        }catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName()+
                           " is inside run method but not sleeping");
    }
    public static void fun() throws InterruptedException  {
        ThreadState obj = new ThreadState();
        Thread t1= new Thread(obj,"Dijkstra");
        Thread t2= new Thread(obj,"Turing");
        Thread.currentThread().setPriority(10);
        System.out.println("Dijkstra before calling start(): "+t1.getState());
        System.out.println("Turing before calling start(): "+t2.getState());
        t1.start();
        t2.start();
        System.out.println("Dijkstra after calling start(): "+t1.getState());
        System.out.println("Turing after calling start(): "+t2.getState());

        Thread.sleep(200);
        System.out.println("Dijkstra before joining: "+t1.getState());
        System.out.println("Turing before joining: "+t2.getState());

        t1.join();
        t2.join();

        System.out.println("Dijkstra after joining: "+t1.getState());
        System.out.println("Turing after joining: "+t2.getState());
    }
    public static void main(String args[]) throws InterruptedException{
        for(int i=1;i<=10;i++){
            fun();
            Thread.sleep(100);
            System.out.println("\n=========================================
                               ========================\n");
        }
    }
}
```

3. **Reentrant Lock:**
Detailed discussion regarding this topic is done in weekly live session, revision session as well as in Discourse, please check the tag `reentrantlock` *on discourse while revising this topic*

```
import java.util.*;
import java.util.concurrent.locks.*;
import java.util.concurrent.*;

class Mathfun{
    long fact_result;
    ReentrantLock r = new ReentrantLock();
    public long factorial(long x) {
        try{
            r.lock();
            System.out.println("Lock count : "+ r.getHoldCount());
            if(x == 1) return 1;
            else  return x*factorial(x-1);
        }catch(Exception e) { return -1;}
        finally{
            r.unlock();
        }
    }
    public void displayFactorial(long x_val) {
        r.lock();
        fact_result = this.factorial(x_val);
        System.out.println("Lock count : "+ r.getHoldCount());
        System.out.println(Thread.currentThread().getName()+" "+fact_result);
        r.unlock();
    }
}
class Thutility extends Thread{
    Mathfun obj;
    Thutility(Mathfun o, String thread_name){
        super(thread_name);
        obj = o;
    }
    public void run() {
        if(Thread.currentThread().getName().equals("t1")) {
            obj.displayFactorial(8);
        }
        else {
            obj.displayFactorial(2);
        }
```

```
    }
}
public class ReEntrantLock{
    public static void main(String args[]) {
        Mathfun o = new Mathfun();
        Thutility t1 = new Thutility(o,"t1");
        Thutility t2 = new Thutility(o,"t2");
        t1.start();
        t2.start();
    }
}
```

There is a subtle but major difference between the working of reentrant locks and semaphores in JAVA. A semaphore can be acquired and released by different threads i.e. it is not necessary that a thread who has acquired a semaphore must only release it, even another thread can release the same semaphore. So placing the acquire and release operations are left at the discretion whereas reentrant locks must be unlocked by the same thread which locked it initially.
Refer the corresponding live session for details

4. **Lambda and Functional Interface:**

   We discussed multiple approaches of using lambdas and the benefits of each of them in this code. Please refer the live session while revising because different parts of this code are not quite interrelated. As the motive was to discuss different approaches and thus give an introduction to this topic therefore specific segment of this code relates to specific approaches discussed in the session. **Please follow the session while checking out this code.**

```
import java.util.*;
interface Eligible1<T>{
    boolean checkEligibility(T obj);
}

interface Eligible2<T>{
    boolean checkEligibility(T obj);
}
class Voter1{
    int age;
    Voter1(int x){
        age =x;
    }
}

class ContainmentZone{
    String colour;
    ContainmentZone(String s){
        colour = s;
    }
}

class MutualFund{
    double returns;
    MutualFund(double r ){
        returns = r;
    }
}

class Account1{
    int balance;
    int overdraftlimit;
    Account1(int bal,int odl){
        balance = bal;
        overdraftlimit = odl;
```

```
    }
}

public class Lambda_feature {
    public static <T> void sort(T[] arr, Comparator<T> fn) {
        // write the complex code for swapping objects to implement the sort
        // use fn.compare(obj1,obj2) whenever required
    }

    public static void do_for_each(Eligible2 fn, ArrayList l) {
        int i=0;
        while(!l.isEmpty()) {
            // do some complex work using this same named method
            fn.checkEligibility(l.get(i));
            i++;
        }
    }

    public static void main(String args[]) {
        Voter1 v1 = new Voter1(10);
        ContainmentZone c1 = new ContainmentZone("red");
        MutualFund m1 = new MutualFund(5.6);

        Eligible2<Voter1> l1 =  x -> {
                                    if(x.age >= 18)
                                        return true;
                                    else
                                        return false;
                                };

        Eligible2<ContainmentZone> l2 = x -> {
                                            if(x.colour.equals("red"))
                                                return true;
                                            else
                                                return false;
                                            };
        System.out.println(l1.checkEligibility(v1));
        System.out.println(l2.checkEligibility(c1));

        Comparable<Account1> com1 =  x -> {
                                             if(x.balance > 5000)
                                                 return 1;
                                            else
                                                return -1;
```

```
                                    };
        Account1 a1 = new Account1(200000,12000);
        System.out.println(com1.compareTo(a1));

        ArrayList<Voter1> Voter11list = new ArrayList<>();
        ArrayList<ContainmentZone> zonelist = new ArrayList<>();
        do_for_each(l1,Voter11list);
        do_for_each(l2,zonelist);

        Voter1[] arrlst = new Voter1[4];
        Comparator<Voter1> ref4 = (Voter1 obj1, Voter1 obj2) -> {
                                        if(obj1.age > obj2.age)
                                            return 1;
                                        else
                                            return -1;
                                    };
        sort(arrlst,ref4);

        Voter1 vv1 = new Voter1(23);        // Eligible2 bcoz > 18
        Account1 aa1 = new Account1(12000,11000); // not Eligible2 bcoz > 10000
        Eligible1<Voter1> fn1 = (obj) -> {return obj.age > 18;};
        Eligible1<Account1> fn2 = (obj) -> {return obj.overdraftlimit < 10000;};

        System.out.println(fn1.checkEligibility(vv1));
        System.out.println(fn2.checkEligibility(aa1));
    }
}
```

5. **Graphical interfaces and event-driven programming using Swing:**

A Java program with GUI to convert height given in "inch" to "cm" and "ft".

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FClass implements ActionListener{
    JFrame frm;
    JTextField inText, outText;
    JButton cmBtn, ftBtn;
    JLabel lblIn, lblOut;
    JPanel inputPanel, outputPanel, btnPanel;
    FClass(){
        frm = new JFrame("Height Converter");
        frm.setSize(340, 140);
        lblIn = new JLabel("Height (inch)");
        inText = new JTextField(10);
        lblOut = new JLabel("Output");
        outText = new JTextField(10);
        cmBtn = new JButton("Convert to CM");
        ftBtn = new JButton("Convert to FT");
        cmBtn.setActionCommand("cm");
        ftBtn.setActionCommand("ft");
        cmBtn.addActionListener(this);
        ftBtn.addActionListener(this);

        inputPanel = new JPanel();
        outputPanel = new JPanel();
        btnPanel = new JPanel();
        inputPanel.add(lblIn);
        inputPanel.add(inText);
        frm.add(inputPanel, "North");
        outputPanel.add(lblOut);
        outputPanel.add(outText);
        frm.add(outputPanel, "Center");
        btnPanel.add(cmBtn);
        btnPanel.add(ftBtn);
        frm.add(btnPanel, "South");

        frm.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        double in = Double.parseDouble(inText.getText());
```

```
        //String s = e.getActionCommand();
        JButton btn = (JButton)e.getSource();
        double out = 0.0;
        if(btn.equals(ftBtn))
         out = in * 0.0833333;
        else if(btn.equals(cmBtn))
         out = in * 2.54;
        outText.setText(out + "");
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

6. **Graphical interfaces and event-driven programming using Swing:**

A Java program with GUI to implement `WindowListener`.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class FClass extends JFrame implements WindowListener{
    JLabel label;
    Container contentPane;
    String str="";
    public FClass() {
        label=new JLabel();
        addWindowListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(700,200);
    }
    public void windowOpened(WindowEvent e) {
        str+="Window Opened ";
        label.setText(str);
    }
    public void windowClosing(WindowEvent e) {
        str+="Window Closing ";
        label.setText(str);
    }
    public void windowClosed(WindowEvent e) {
    }
    public void windowIconified(WindowEvent e) {
        str+="Window Iconified ";
        label.setText(str);
    }
    public void windowDeiconified(WindowEvent e) {
        str+="Window Deiconified ";
        label.setText(str);
    }
    public void windowActivated(WindowEvent e) {
        str+="Window Activated";
        label.setText(str);
    }
    public void windowDeactivated(WindowEvent e) {
        str+="Window Deactivated ";
        label.setText(str);
```

```
    }
    public static void main(String[] args) {
        new FClass();
    }
}
```

7. **Graphical interfaces and event-driven programming using Swing:**

A Java program with GUI to implement a panel changing color based on the check-box selected.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FClass extends JFrame implements ActionListener{
    JCheckBox b1,b2,b3;
    JPanel panel1,panel2;
    public FClass(){
        panel1=new JPanel();
        panel2=new JPanel();
        b1=new JCheckBox("Yellow");
        b2=new JCheckBox("Black");
        b3=new JCheckBox("Cyan");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        panel1.add(b1);
        panel1.add(b2);
        panel1.add(b3);
        add(panel1,"South");
        add(panel2,"Center");
        setVisible(true);
        setSize(400,400);
    }
    public void actionPerformed(ActionEvent e) {
     if(e.getSource().equals(b1))
         panel2.setBackground(Color.yellow);
     if(e.getSource().equals(b2))
         panel2.setBackground(Color.black);
     if(e.getSource().equals(b3))
         panel2.setBackground(Color.cyan);
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

8. **Graphical interfaces and event-driven programming using Swing:**

A Java program with GUI to implement `KeyListener` interface.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FClass extends JFrame implements KeyListener{
    JLabel label;
    Container contentPane;
    public FClass() {
        label=new JLabel();
        addKeyListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(200,200);
    }
    public void keyTyped(KeyEvent e) {
        label.setText("KeyTyped");
    }
    public void keyPressed(KeyEvent e) {
        label.setText("keyPressed");
    }
    public void keyReleased(KeyEvent e) {
        label.setText("keyReleased");
    }
    public static void main(String[] args) {
        new FClass();
    }
}
```

9. **Graphical interfaces and event-driven programming using Swing:**

A Java Swing program to understand layouts.
Please uncomment the code as required.

```
import javax.swing.*;
import java.awt.*;
public class FClass extends JFrame{
    JButton b1,b2,b3,b4,b5;
    LayoutManager manager;
    public FClass(){
     //manager=new BorderLayout();
     //manager=new GridLayout(3, 3);
     manager=new FlowLayout();
        setLayout(manager);
        b1=new JButton("Button 1");
        b2=new JButton("Button 2");
        b3=new JButton("Button 3");
        b4=new JButton("Button 4");
        b5=new JButton("Button 5");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        //add(b1,"North");
        //add(b2,"Center");
        //add(b3,"West");
        //add(b4,"East");
        //add(b5,"South");
        setVisible(true);
        setSize(400,400);
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

10. **Stream:**

    Using stream to count number of words in a given text that have length more than 5.

```
import java.util.*;
class FClass{
    public static void main(String[] args){
        String text = """
                As a sample, Week 1 content videos have been
                made available for the first four
                Foundational Level courses for you
                to try and learn from. We recommend you to
                check out these lectures and try the sample
                assignment we have put out for each
                course. Here are the links to Week 1 Content
                & Assignment Pages
                """;

        List<String> words = List.of(text.split(" "));

        long count = words.stream()
            .filter(w -> w.length() > 5)
            .count();
        System.out.println(count);
    }
}
```

    Alternate code using `Stream.of`.

```
import java.util.*;
import java.util.stream.*;
class FClass{
    public static void main(String[] args){
        String text = """
            As a sample, Week 1 content videos have been made available
            for the first four
            Foundational Level courses for you to try and learn
            from. We recommend you to
            check out these lectures and try the sample
            assignment we have put out for each
            course. Here are the links to Week 1 Content
            & Assignment Pages
"""";
```

```
        //String[] arr = text.split(" ");
        Stream<String> words = Stream.of(text.split(" "));

        long count = words.filter(w -> w.length() > 5)
            .count();
        System.out.println(count);
    }
}
```

11. **Stream:**

    Use of `forEach` method.

```
import java.util.*;
import java.util.stream.*;
class FClass{
    public static void main(String[] args){
        /*
        Stream.generate(() -> "hello")
                        .limit(5)
                        .forEach(System.out::println);

        Stream.generate(() -> "hello")
                            .forEach(System.out::println);

        Stream.generate(Math::random)
                            .map(i -> (int)(i * 100))
                            .limit(10)
                            .forEach(System.out::println);
        */
        Stream.iterate(0, n -> n+1)
                            .limit(10)
                            .forEach(System.out::println);

        Stream.iterate(0, n -> n < 10, n -> n+1)
                            .forEach(System.out::println);
    }
}
```

12. **Stream:**

    Use of `map` function.

```
import java.util.*;
import java.util.stream.*;
class FClass{
    public static void main(String[] args){
        String text = """
            As a sample, Week 1 content videos have been
            made available for the first four
            Foundational Level courses for you to try
            and learn from. We recommend you to
            check out these lectures and try the sample
            assignment we have put out for each
            course. Here are the links to Week 1 Content
            & Assignment Pages
            """;

        List<String> words = List.of(text.split(" "));

        /*
        words.stream()
            .filter(w -> w.length() > 5)
            .map(s -> s.substring(0,1))
            .forEach(System.out::println);

        Stream.iterate(0, n -> n < 10, n -> n+1)
                            .map(n -> n * n)
                            .forEach(System.out::println);
    }
}
```

13. **Stream:**

Use of `flatMap` method.

```
import java.util.*;
import java.util.stream.*;
class FClass{
    public static List<String> getCharList(String s){
        String[] sArr = s.split("");
        ArrayList<String> sList = new ArrayList<String>();
        for(String i : sArr)
            sList.add(i);
        return sList;
    }
    public static void main(String[] args){
        List<String> list = new ArrayList();
        list.add("one");
        list.add("two");
        list.add("three");

        List<List<String>> list1 = list.stream()
        //{"one", "two", "three"}
            .map(FClass::getCharList)
            //{{"o", "n", "e"}, {"t", "w", "o"}, {"t", "h", "r", "e", "e"}}
            .collect(Collectors.toList());

        System.out.println(list1);

        List<String> list2 = list.stream()
            .map(FClass::getCharList)
            .flatMap(Collection::stream)
            //{"o", "n", "e", "t", "w", "o", "t", "h", "r", "e", "e"}
            .collect(Collectors.toList());

        System.out.println(list2);
    }
}
```

14. **Stream:**

Generating streams using `iterate` and `generate` methods.

```
import java.util.*;
import java.util.stream.*;
class FClass{
    public static void main(String[] args){
        /*
        int n = 20;
        Stream.iterate(1, i -> i + 1)
                    .limit(n)
                    .skip(n - 5)
                    //.takeWhile(i -> i <= 10)
                    //.filter(i -> i < 11)
                    .forEach(System.out::println);

        Stream.iterate(1, i -> i + 1)
                    .takeWhile(i -> i < 5)
                    .forEach(System.out::println);

        Stream.iterate(1, i -> i + 1)
                    .dropWhile(i -> i < 5)
                    .limit(10)
                    .forEach(System.out::println);

        Stream.generate(Math::random)
                    .limit(10)
                    .filter(i -> i < 0.5)
                    .forEach(System.out::println);
        */

        Optional<Double> maxrand  = Stream.generate(Math::random)
                    .limit(10)
                    .filter(i -> i < 0.5)
                    .max(Double::compareTo);
        System.out.println(maxrand);
    }
}
```

15. **Thread:**

Create thread by extending `Thread` class.

```
class PrlCount extends Thread{
    public PrlCount(String tname) {
        super(tname);
}
    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println(Thread.currentThread().getName() + " : " + i);
            try {
                sleep(10);
            }
            catch(InterruptedException e) {}
        }
    }
}
class FClass{
    public static void main(String[] args) {
        Thread th1 = new PrlCount("worker-1");
        Thread th2 = new PrlCount("worker-2");
        Thread.currentThread().setName("Main-thread");
        th1.start();
        th2.run();
        th2.start();
    }
}
```

16. **Thread – Race Condition**

A Java program to understand race condition and critical section.

```
class Bank{
    private double[] accounts = new double[100];
    Bank(double initBalance){
        for(int i = 0; i < 100; i++)
            accounts[i] = initBalance;
    }
    public boolean transfer(int src, int trg, double amount) {
        if(accounts[src] < amount)
            return false;
        accounts[src] -= amount;    //1000 - 500 = 500
        System.out.println(src + " to " + trg + " : " + amount);
        accounts[trg] += amount;    //1000 + 500 = 1500
        return true;
    }
    public double audit() {
        double balance = 0.0;
        for(double b : accounts) {  //1000 + 1500
            balance += b;
        }
        return balance;
    }
}

class Transaction extends Thread{
    Bank b;
    public Transaction(Bank bank) {
        b = bank;
        int src = (int)(Math.random() * 100);    //7
        int trg = (int)(Math.random() * 100);    //8

        double amount = 500;
        b.transfer(src, trg, amount);
    }
}

class CheckBalance extends Thread{
    Bank b;
    public CheckBalance(Bank bank) {
        b = bank;
    }
```

```
    public void run() {
        System.out.println("Total balance : " + b.audit());
    }
}
class FClass{
    public static void main(String[] args) {
        Bank b = new Bank(1000.00);
        for(int i = 0; i < 100; i++) {
            Thread th2 = new Transaction(b);
            Thread th1 = new CheckBalance(b);
            th2.start();
            th1.start();
        }
    }
}
```

17. **GUI**

A Java program to create a sample GUI that can perform arithmetic operations.

```
import javax.swing.*;
import java.awt.event.*;
public class AWTAddition extends JFrame implements ActionListener{
    JLabel input1,input2,output;
    JRadioButton add,sub,mul,div;
    ButtonGroup bg=new ButtonGroup();
    JTextField tf1,tf2;
    public AWTAddition() {
        input1=new JLabel("Enter A:");
        input2=new JLabel("Enter B:");
        output=new JLabel("");
        tf1=new JTextField(25);
        tf2=new JTextField(25);
        add=new JRadioButton("Add");
        sub=new JRadioButton("Sub");
        mul=new JRadioButton("Mul");
        div=new JRadioButton("Div");
        input1.setBounds(150,150,50,25);
        tf1.setBounds(210,150,50,25);
        input2.setBounds(150,180,50,25);
        tf2.setBounds(210,180,50,25);
        add.setBounds(150,210,50,25);
        sub.setBounds(230,210,50,25);
        mul.setBounds(280,210,50,25);
        div.setBounds(330,210,50,25);
        output.setBounds(150,240,100,25);
        add(input1);
        add(input2);
        add(tf1);
        add(tf2);
        bg.add(add);
        bg.add(sub);
        bg.add(mul);
        bg.add(div);
        add(add);
        add(sub);
        add(mul);
        add(div);
        add(output);
        setVisible(true);
```

```
        setSize(600,600);
        add.addActionListener(this);
        sub.addActionListener(this);
        mul.addActionListener(this);
        div.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        int a=Integer.parseInt(tf1.getText());
        int b=Integer.parseInt(tf2.getText());
        if(add.isSelected())
        output.setText("Addition is "+(a+b));
        if(sub.isSelected())
        output.setText("Subtraction is "+(a-b));
        if(mul.isSelected())
            output.setText("Multiplication is "+(a*b));
        if(div.isSelected())
            output.setText("Division is "+(a/b));
    }
    public static void main(String[] args) {
        new AWTAddition();
    }
}
```

18. **GUI**

A Java program to create a sample GUI that can find factorial of a given number.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class FactorialGUI extends JFrame implements ActionListener{
    JLabel inputlabel,outputlabel;
    JTextField tf;
    JButton button;
    JPanel inputpanel, outputpanel;
    public FactorialGUI() {
        //Input Panel
        inputlabel=new JLabel("Enter number:");
        tf=new JTextField(25);
        button=new JButton("Find factorial");
        inputpanel=new JPanel();
        inputpanel.add(inputlabel);
        inputpanel.add(tf);
        inputpanel.add(button);
        //Output panel
        outputlabel=new JLabel("");
        outputpanel=new JPanel();
        outputpanel.add(outputlabel);
        //add panels to the frame
        add(inputpanel,"North");
        add(outputpanel,"South");
        setVisible(true);
        setSize(500,500);
        button.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        int num=Integer.parseInt(tf.getText());
        int fact=1;
        for(int i=1;i<=num;i++) {
            fact=fact*i;
        }
        outputlabel.setText("factorial is "+fact);
    }
    public static void main(String[] args) {
        new FactorialGUI();
    }
}
```