

Code

```
/* This program calculates summations and factorials by passing a number
into methods outside of the main method to do each calculation. It uses
four methods: iterativeFactorial(int num), recursiveFactorial(int num),
iterativeSummation(int num), and recursiveSummation(int num). */
```

```
public class Recursion{

    public static void main(String[] args){

        int num1 = 23;

        int num2 = 25;


        int iterFactAnswer = iterativeFactorial(num1);
        System.out.println("By iteration, " + num1 + "! = " + iterFactAnswer);


        int recurFactAnswer = recursiveFactorial(num1);
        System.out.println("By recursion, " + num1 + "! = " + recurFactAnswer);


        int iterSumAnswer = iterativeSummation(num2);
        System.out.println("By iteration, the summation from 1 to " + num2 + "
= " + iterSumAnswer);


        int recurSumAnswer = recursiveSummation(num2);
        System.out.println("By recursion, the summation from 1 to " + num2 + "
= " + recurSumAnswer);

    } // close main


/* Method: calculate factorial by iteration */
    static int iterativeFactorial(int num){
        int j;

        int prod = 1;

        for(j = 1; j <= num; j++){
            prod = prod * j;
        }
    }
}
```

```
    }  
    return prod;  
}
```

```
/* Method: calculate factorial by recursion */  
static int recursiveFactorial(int num){  
    int j;  
    int prod;  
    if(num == 1){  
        return 1;  
    }  
    else{  
        return num * recursiveFactorial(num -1);  
    }  
}
```

```
/* Method: calculate summation by iteration */  
static int iterativeSummation(int num){  
    int j;  
    int sum;  
    sum = 0;  
    for(j = 1; j <= num; j++){  
        sum = sum +j;  
    }  
    return sum;  
}
```

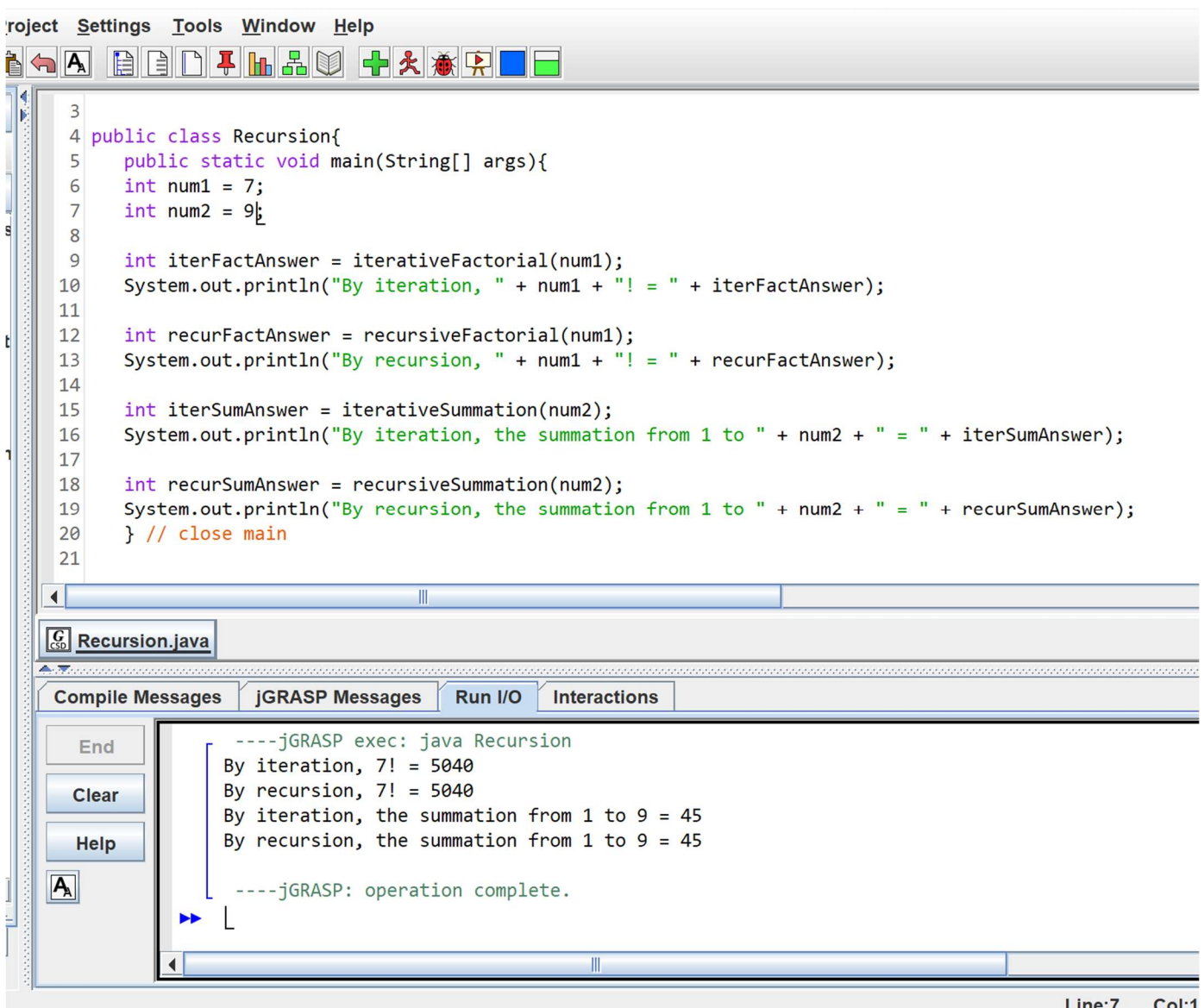
```
/* Method: calculate summation by recursion */  
static int recursiveSummation(int num){  
    int j;  
    int sum = 1;  
    if(num == 1){
```

```

        return 1;
    }
    else{
        return num + recursiveSummation(num - 1);
    }
}
}

```

Running



The screenshot shows the jGRASP IDE interface. The top menu bar includes Project, Settings, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main editor window displays the code for Recursion.java, which includes a public class Recursion with a main method. The code calculates the factorial of 7 using both iterative and recursive methods, and the summation of numbers from 1 to 9 using both iterative and recursive methods. The Run I/O window at the bottom shows the output of the program, which matches the expected results.

```

3
4 public class Recursion{
5     public static void main(String[] args){
6         int num1 = 7;
7         int num2 = 9;
8
9         int iterFactAnswer = iterativeFactorial(num1);
10        System.out.println("By iteration, " + num1 + "! = " + iterFactAnswer);
11
12        int recurFactAnswer = recursiveFactorial(num1);
13        System.out.println("By recursion, " + num1 + "! = " + recurFactAnswer);
14
15        int iterSumAnswer = iterativeSummation(num2);
16        System.out.println("By iteration, the summation from 1 to " + num2 + " = " + iterSumAnswer);
17
18        int recurSumAnswer = recursiveSummation(num2);
19        System.out.println("By recursion, the summation from 1 to " + num2 + " = " + recurSumAnswer);
20    } // close main
21

```

Recursion.java

Compile Messages | jGRASP Messages | Run I/O | Interactions

End
Clear
Help

```

----jGRASP exec: java Recursion
By iteration, 7! = 5040
By recursion, 7! = 5040
By iteration, the summation from 1 to 9 = 45
By recursion, the summation from 1 to 9 = 45

----jGRASP: operation complete.

```

Line:7 Col:1

Method: Factorial calculation by iteration

The screenshot shows a Java IDE with a menu bar (Project, Settings, Tools, Window, Help) and a toolbar. The main editor displays the following Java code:

```
17
18 int recurSumAnswer = recursiveSummation(num2);
19 System.out.println("By recursion, the summation from 1 to " + num2 + " = " + recurSumAnswer);
20 } // close main
21
22 /* Method: calculate factorial by iteration */
23 static int iterativeFactorial(int num){
24     int j;
25     int prod = 1;
26     for(j = 1; j <= num; j++){
27         prod = prod * j;
28     }
29     return prod;
30 }
31
32 /* Method: calculate factorial by recursion */
33 static int recursiveFactorial(int num){
34     int j;
35     int prod;
```


Below the code editor is a tab labeled "Recursion.java". Underneath the tab are four sub-tabs: "Compile Messages", "JGRASP Messages", "Run I/O", and "Interactions". The "Run I/O" tab is active, showing the following output:

```
----jGRASP exec: java Recursion
By iteration, 21! = -1195114496
By recursion, 21! = -1195114496
By iteration, the summation from 1 to 19 = 190
By recursion, the summation from 1 to 19 = 190
----jGRASP: operation complete.
```

At the bottom right of the IDE, a status bar displays: "Line:7 Col:17 Code:59 Top:17 OVS".

Method: Factorial calculation by recursion

object Settings Tools Window Help



```
30     }
31
32     /* Method: calculate factorial by recursion */
33     static int recursiveFactorial(int num){
34         int j;
35         int prod;
36         if(num == 1){
37             return 1;
38         }
39         else{
40             return num * recursiveFactorial(num -1);
41         }
42     }
43
44     /* Method: calculate summation by iteration */
45     static int iterativeSummation(int num){
46         int j;
47         int sum;
48         sum = 0;
```

Recursion.java

Compile Messages jGRASP Messages Run I/O Interactions

End Clear Help

```

L ----jGRASP: operation complete.

----jGRASP exec: java Recursion
By iteration, 13! = 1932053504
By recursion, 13! = 1932053504
By iteration, the summation from 1 to 17 = 153
By recursion, the summation from 1 to 17 = 153
```

Method: Summation calculation by iteration:

The screenshot shows a Java IDE with a file named `Recursion.java`. The code defines two static methods: `iterativeSummation` and `recursiveSummation`. The `iterativeSummation` method calculates the sum of integers from 1 to `num` using a loop. The `recursiveSummation` method calculates the sum of integers from 1 to `num` using recursion. The IDE's output window shows the results of running the program, displaying the factorial of 11 and the summation of integers from 1 to 15 using both methods.

```
41     }
42 }
43
44 /* Method: calculate summation by iteration */
45 static int iterativeSummation(int num){
46     int j;
47     int sum;
48     sum = 0;
49     for(j = 1; j <= num; j++){
50         sum = sum + j;
51     }
52     return sum;
53 }
54
55 /* Method: calculate summation by recursion */
56 static int recursiveSummation(int num){
57     int j;
58     int sum = 1;
59     if(num == 1){
```

Recursion.java

Compile Messages | JGRASP Messages | Run I/O | Interactions

End
Clear
Help

```
----jGRASP exec: java Recursion
By iteration, 11! = 39916800
By recursion, 11! = 39916800
By iteration, the summation from 1 to 15 = 120
By recursion, the summation from 1 to 15 = 120
----jGRASP: operation complete.
```

Line:7 Col:17 Code:59 T

Method: Summation calculation by recursion:

The screenshot displays a Java IDE with a code editor and a console window. The code editor shows a Java program for calculating the summation of numbers from 1 to 25 using recursion. The console window shows the execution output, confirming the results for 23! and the summation from 1 to 25.

Code Editor:

```
51     }
52     return sum;
53 }
54
55 /* Method: calculate summation by recursion */
56 static int recursiveSummation(int num){
57     int j;
58     int sum = 1;
59     if(num == 1){
60         return 1;
61     }
62     else{
63         return num + recursiveSummation(num - 1);
64     }
65 }
66
67 }
```

Console Window:

Compile Messages | jGRASP Messages | Run I/O | Interactions

End
Clear
Help

```
----jGRASP exec: java Recursion
By iteration, 23! = 862453760
By recursion, 23! = 862453760
By iteration, the summation from 1 to 25 = 325
By recursion, the summation from 1 to 25 = 325

----jGRASP: operation complete.
```

Line:7 Col:17 Code:59 Tc